AURIX™ TC3xx

About this document

Scope and purpose

This User's Manual describes the Infineon AURIX™ TC3xx Platform family, a range of 32-bit multicore microcontrollers based on the Infineon TriCore™ Architecture.

This family document covers the superset functionality. It is supplemented by a separate device specific document "Appendix" that covers differences of a particular device to this family superset.

Table of Contents

27 System Timer (STM)

This chapter describes the System Timer (STM). The STM is designed for global system timing applications requiring both high precision and long period.

27.1 Feature List

The STM has the following features:

- Free-running 64-bit counter
- All 64 bits can be read synchronously
- Different 32-bit portions of the 64-bit counter can be read synchronously
- Flexible service request generation based on compare match with partial STM content
- Counting starts automatically after an Application Reset
- STM registers are reset by an Application Reset if bit ARSTDIS.STMxDIS is cleared. If bit ARSTDIS.STMxDIS is set, the STM registers are not reset by application reset, and are reset by system reset instead.

Special STM register semantics provide synchronous views of the entire 64-bit counter, or 32-bit subsets at different levels of resolution.

27.2 Overview

Figure 1 provides an overview on the STM module. It shows the options for reading parts of the STM content.

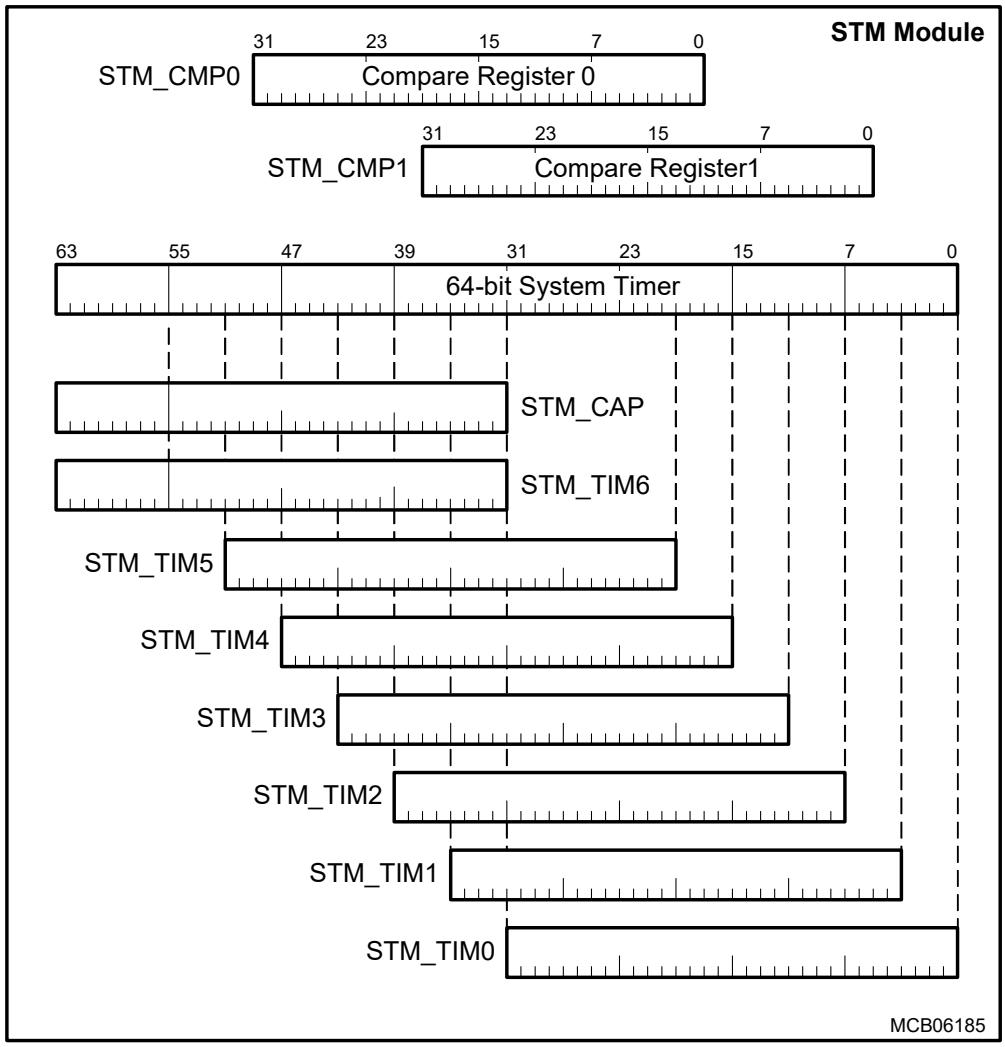


Figure 1 General Block Diagram of the STM Module

27.3 Functional Description

The STM is an upward counter, running at frequency f_{STM}. In case of an Application Reset, the STM is reset if bit SCU_ARSTDIS.STMxDIS is cleared. After reset, the STM is enabled and immediately starts counting up. It is not possible to affect the content of the timer during normal operation. The timer registers can only be read but not written to.

The STM can be optionally disabled for power-saving purposes, or suspended for debugging purposes. In suspend mode, the STM clock is stopped but all registers are still readable.

Due to the 64-bit width of the STM, it is not possible to read its entire content with one instruction. It needs to be read with two load instructions. Since the timer would continue to count between the two load operations, there is a chance that the two values read are not consistent (due to possible overflow from the low part of the timer to the high part between the two read operations). To enable a synchronous and consistent reading of the STM content, a capture register (CAP) is implemented. It latches the content of the high part of the STM each time

when one of the registers TIM0 to TIM5 is read. Thus, CAP holds the upper value of the timer at exactly the same time when the lower part is read. The second read operation would then read the content of the CAP to get the complete timer value.

The STM can also be read in sections from seven registers, TIM0 through TIM6, that select increasingly higher-order 32-bit ranges of the STM. These can be viewed as individual 32-bit timers, each with a different resolution and timing range.

The content of the 64-bit System Timer can be compared against the content of two compare values stored in the CMP0 and CMP1 registers. Service requests can be generated on a compare match of the STM with the CMP0 or CMP1 registers.

27.3.1 Compare Register Operation

The content of the 64-bit STM can be compared against the content of two compare values stored in the CMP0 and CMP1 registers. Service requests can be generated on a compare match of the STM with the CMP0 or CMP1 registers.

Two parameters are programmable for the compare operation:

1. The width of the relevant bits in registers CMP0/CMP1 (compare width MSIZEx) that is taken for the compare operation can be programmed from 0 to 31.
2. The first bit location in the 64-bit STM that is taken for the compare operation can be programmed from 0 to 31.

These programming capabilities make compare functionality very flexible. It even makes it possible to detect bit transitions of a single bit n (n = 0 to 31) within the 64-bit STM by setting MSIZE = 0 and MSTART = n.

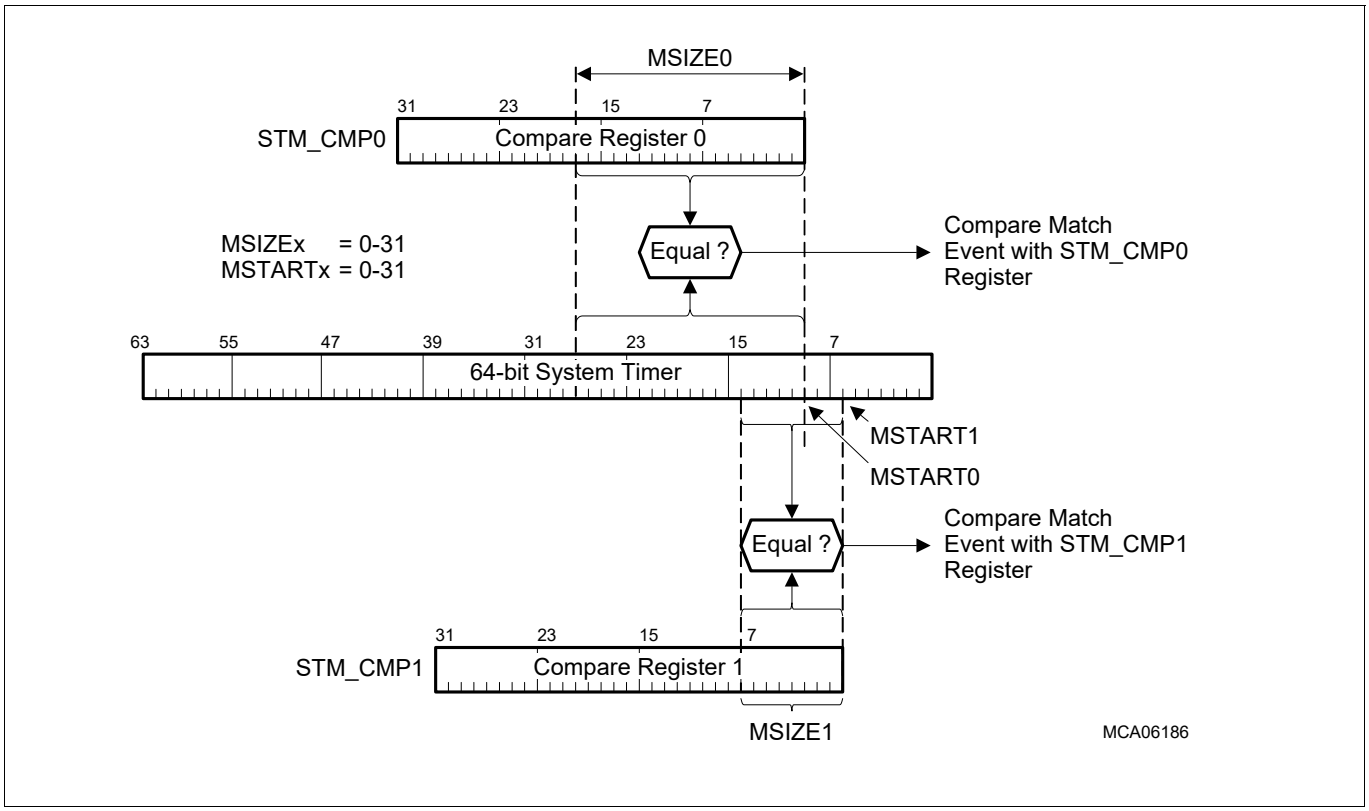


Figure 2 Compare Mode Operation

Figure 2 shows an example of the compare operation. In this example the following parameters are programmed:

- MSIZE0 = $10001_B = 17_D$; MSTART0 = $01010_B = 10_D$
- MSIZE1 = $00111_B = 7_D$; MSTART1 = $00111_B = 7_D$

A compare operation with MSIZE not equal 11111_B always implies that the compared value as stored in the CMP register is right-extended with zeros. This means that in the example of Figure 2, the compare register content CMP0[17:0] plus ten zero bits right-extended is compared with STM[27:0] with STM[9:0] = 000_H. In case of register CMP1, STM[14:0] with STM[6:0] = 00_H are compared with CMP1[9:0] plus seven zero bits right-extended.

27.3.2 Compare Match Interrupt Control

The compare match interrupt control logic is shown in **Figure 3**. Each CMPx register has its compare match interrupt request flag (ICR.CMPxIR) that is set by hardware on a compare match event. The interrupt request flags can be set (ISCR.CMPxIRS) or cleared (ISCR.CMPxIRR) by software. Note that setting ICR.CMPxIR by writing a 1 into ISSR.CMPxIRS does not generate an interrupt at STMIRx. The compare match interrupts from CMP0 and CMP1 can be further directed by ICR.CMPxOS to either output signal STMIR0 or STMIR1.

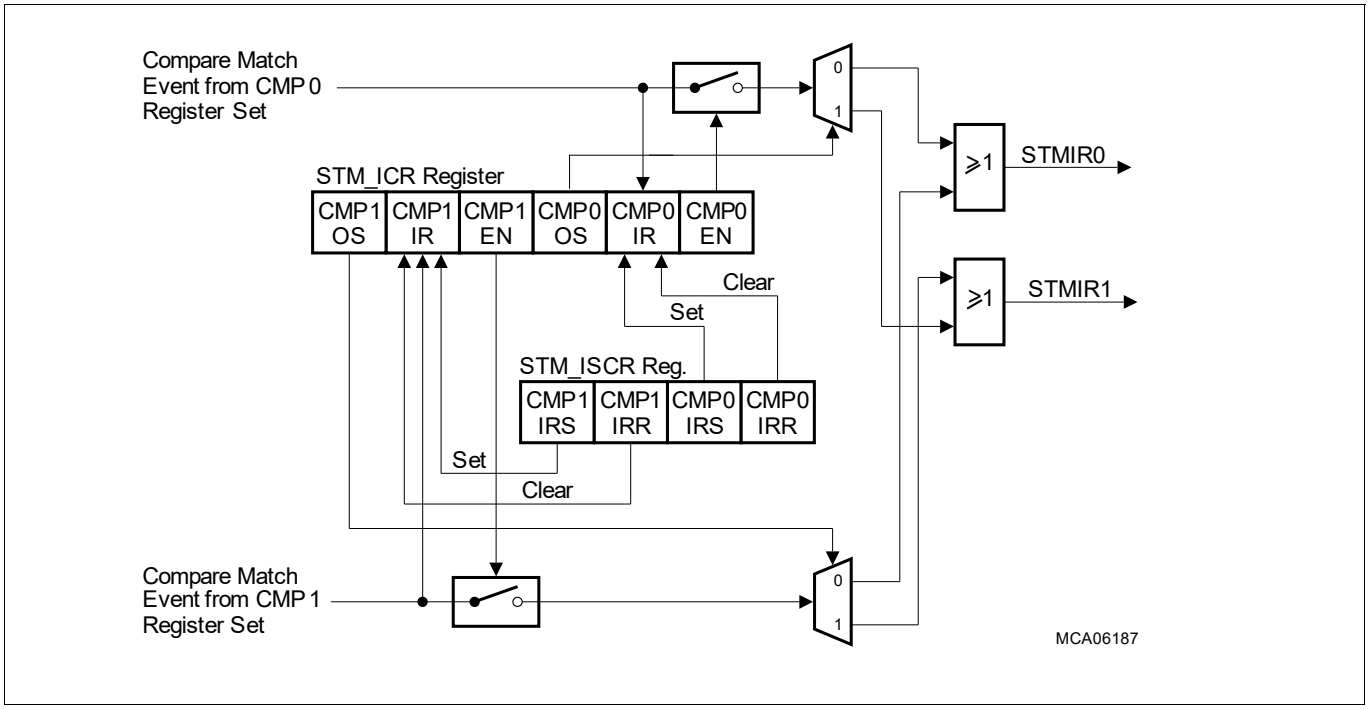


Figure 3 STM Interrupt Control

The compare match interrupt flags ICR.CMPxIR are immediately set after an STM reset operation, caused by a compare match event with the reset values of the STM and the compare registers CMPx. This does not directly generate compare match interrupts because the compare match interrupts are automatically disabled after an STM reset operation (ICR.CMPxEN = 0). Therefore, before enabling a compare match interrupt after an STM Application Reset, the software should configure the STM and modify the reset values of the compare registers. Otherwise, undesired compare match interrupt events are triggered. The CMPxIR flags which are set after an STM reset can be cleared by writing register ISCR with CMPxIRR set.

27.3.3 Using Multiple STMs

For systems that include multiple CPUs there are also multiple STMs available. Each STM is aimed to serve as time base for one individual CPU operating system main scheduler clock trigger. This is done by the usage of one compare register and the associated interrupt generating the trigger.

All STM modules are connected and controlled by f_{STM} and can therefore operate on the same frequency if desired.

27.3.4 STM as Reset Trigger

A compare match triggered by an CMP0 event can generate a reset in the system. The reset has to be enabled for each STM module individually in register SCU_RSTCON.

27.4 Registers

This section describes the registers of the STM.

Table 1 Register Overview - STM (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	0000_H	U,SV	SV,E,P	Application Reset	7
ID	Module Identification Register	0008_H	U,SV	BE	Application Reset	8
TIM0	Timer Register 0	0010_H	U,SV	BE	Application Reset	9
TIM1	Timer Register 1	0014_H	U,SV	BE	Application Reset	10
TIM2	Timer Register 2	0018_H	U,SV	BE	Application Reset	10
TIM3	Timer Register 3	$001C_H$	U,SV	BE	Application Reset	10
TIM4	Timer Register 4	0020_H	U,SV	BE	Application Reset	11
TIM5	Timer Register 5	0024_H	U,SV	BE	Application Reset	11
TIM6	Timer Register 6	0028_H	U,SV	BE	Application Reset	12
CAP	Timer Capture Register	$002C_H$	U,SV	BE	Application Reset	12
CMPx	Compare Register x	0030_H+x *4	U,SV	U,SV,P	Application Reset	13
CMCON	Compare Match Control Register	0038_H	U,SV	U,SV,P	Application Reset	14
ICR	Interrupt Control Register	$003C_H$	U,SV	U,SV,P	Application Reset	16
ISCR	Interrupt Set/Clear Register	0040_H	U,SV	U,SV,P	Application Reset	17
TIM0SV	Timer Register 0 Second View	0050_H	U,SV	BE	Application Reset	9
CAPSV	Timer Capture Register Second View	0054_H	U,SV	BE	Application Reset	13
OCS	OCDS Control and Status Register	$00E8_H$	U,SV	SV,P,OEN	Debug Reset	19
KRSTCLR	Kernel Reset Status Clear Register	$00EC_H$	U,SV	SV,E,P	Application Reset	22
KRST1	Kernel Reset Register 1	$00F0_H$	U,SV	SV,E,P	Application Reset	21

Table 1 Register Overview - STM (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
KRST0	Kernel Reset Register 0	$00F4_H$	U,SV	SV,E,P	Application Reset	21
ACCEN1	Access Enable Register 1	$00F8_H$	U,SV	SV,SE	Application Reset	20
ACCEN0	Access Enable Register 0	$00FC_H$	U,SV	SV,SE	Application Reset	20

27.4.1 Clock Control Register

Clock Control Register

The STM clock control register is used to switch the STM on or off and to control its input clock rate. After reset, the STM is always enabled and starts counting. The STM can be disabled by setting bit DISR to 1.

CLC

Clock Control Register (0000_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	rw	r	rw

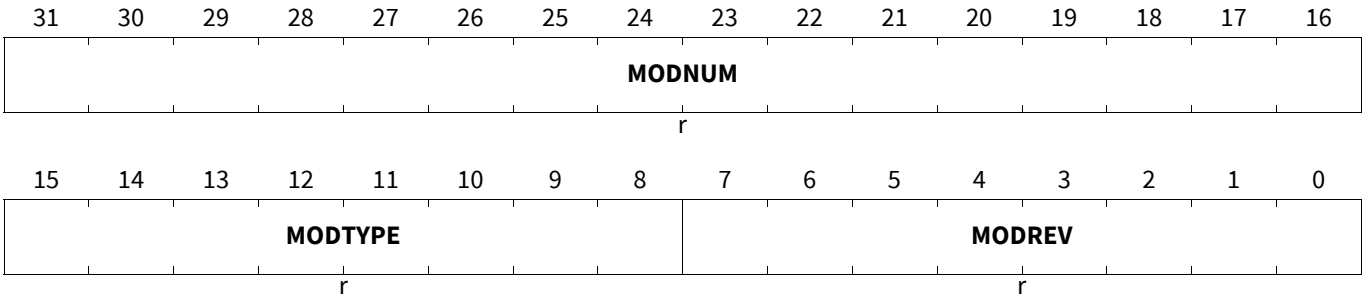
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the STM module. *Note: f_{STM} is generated by the CCU.* 0_B No disable requested 1_B Disable requested
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the STM module. 0_B STM module is enabled 1_B STM module is disabled
EDIS	3	rw	**Sleep Mode Enable Control** Used for module sleep mode control.
0	2	rw	**Reserved** Read as 0; shall be written with 0.
0	31:4	r	**Reserved** Read as 0; should be written with 0.

Module Identification Register

The STM Module Identification Register ID contains read-only information about the module version.

ID

Module Identification Register **(0008_H)** **Application Reset Value: $0000\ C007_H$**



Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** MODREV defines the module revision number. The value of a module revision starts with 01_H (first revision). Current revision is 0x7.
MODTYPE	15:8	r	**Module Type** This bit field defines the module as a 32-bit module: $C0_H$
MODNUM	31:16	r	**Module Number Value** This bit field defines the module identification number for the STM: 0068_H

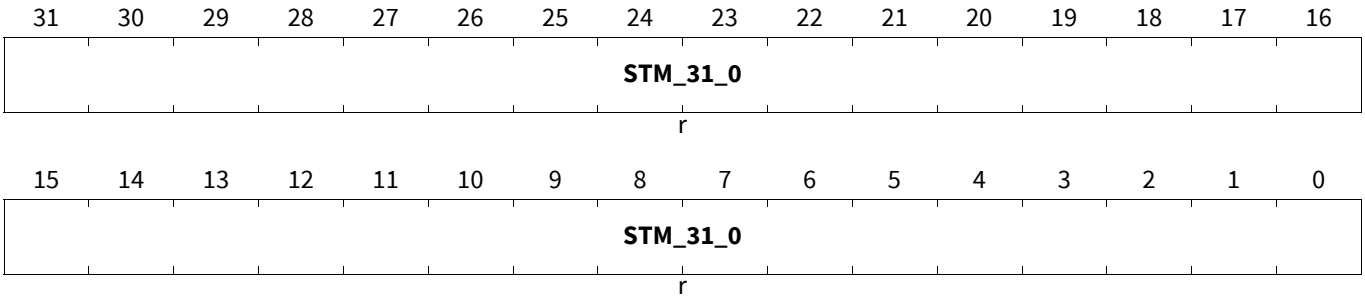
27.4.2 Timer/Capture Registers

Registers TIM0 to TIM6 provide 32-bit views at varying resolutions of the underlying STM counter.

Timer Register 0

Register TIM0SV address the STM[31:0] bits at a second optional address as TIM0.

TIM0
Timer Register 0 **(0010_H)** **Application Reset Value: $0000\ 0000_H$**

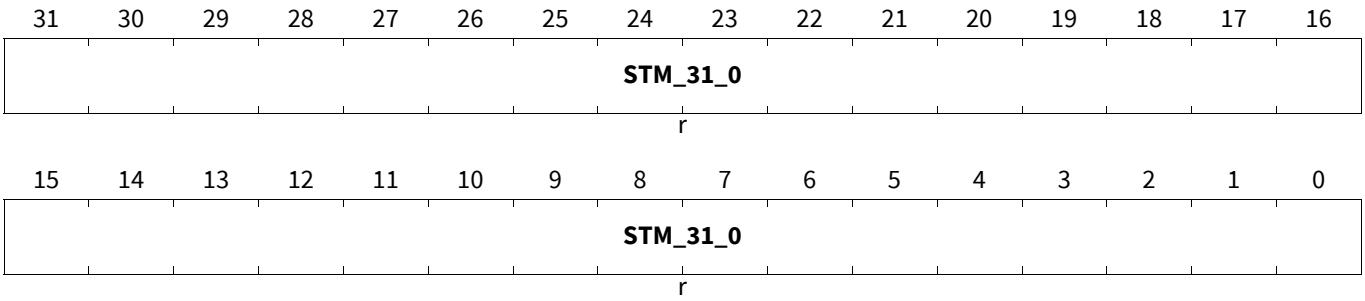


Field	Bits	Type	Description
STM_31_0	31:0	r	**System Timer Bits [31:0] - STM[31:0]** This bit field contains bits [31:0] of the 64-bit STM.

Timer Register 0 Second View

Register TIM0SV address the STM[31:0] bits at a second optional address as TIM0.

TIM0SV
Timer Register 0 Second View **(0050_H)** **Application Reset Value: $0000\ 0000_H$**

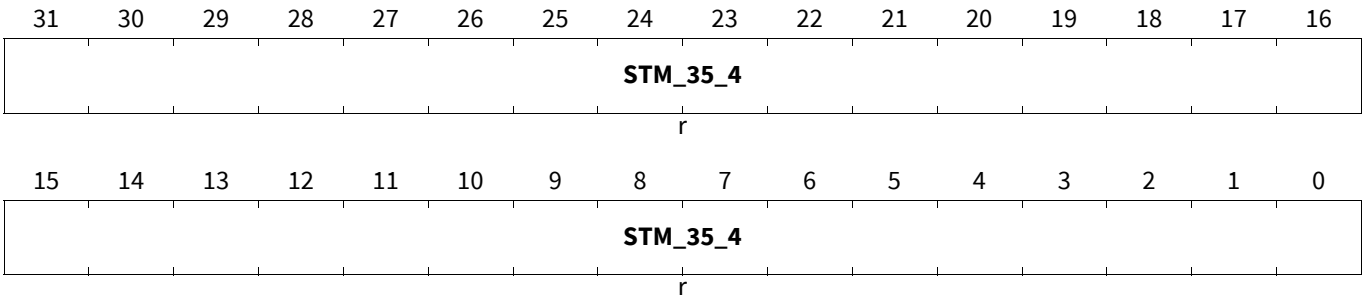


Field	Bits	Type	Description
STM_31_0	31:0	r	**System Timer Bits [31:0] - STM[31:0]** This bit field contains bits [31:0] of the 64-bit STM.

Timer Register 1

TIM1

Timer Register 1 **(0014_H)** **Application Reset Value: $0000\ 0000_H$**

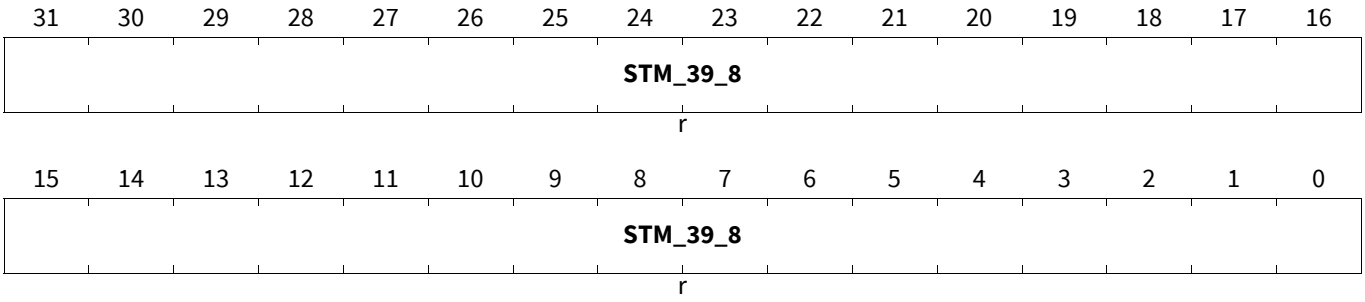


Field	Bits	Type	Description
STM_35_4	31:0	r	**System Timer Bits [35:4] - STM[35:4]** This bit field contains bits [35:4] of the 64-bit STM.

Timer Register 2

TIM2

Timer Register 2 **(0018_H)** **Application Reset Value: $0000\ 0000_H$**

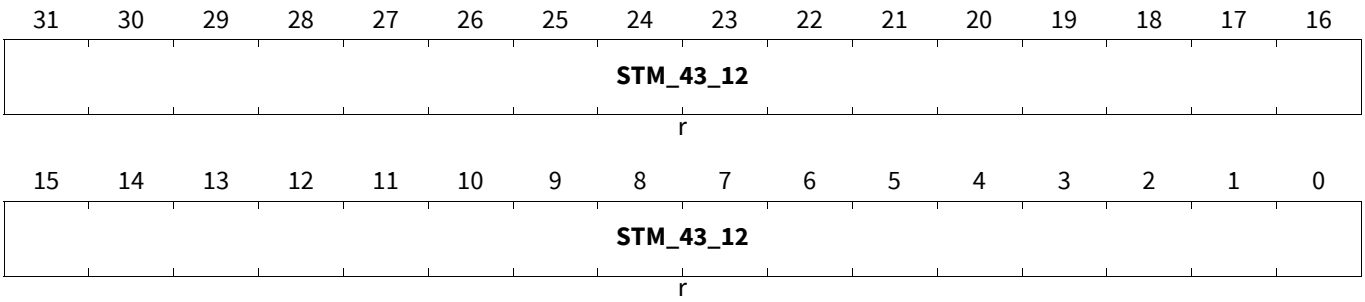


Field	Bits	Type	Description
STM_39_8	31:0	r	**System Timer Bits [39:8] - STM[39:8]** This bit field contains bits [39:8] of the 64-bit STM.

Timer Register 3

TIM3

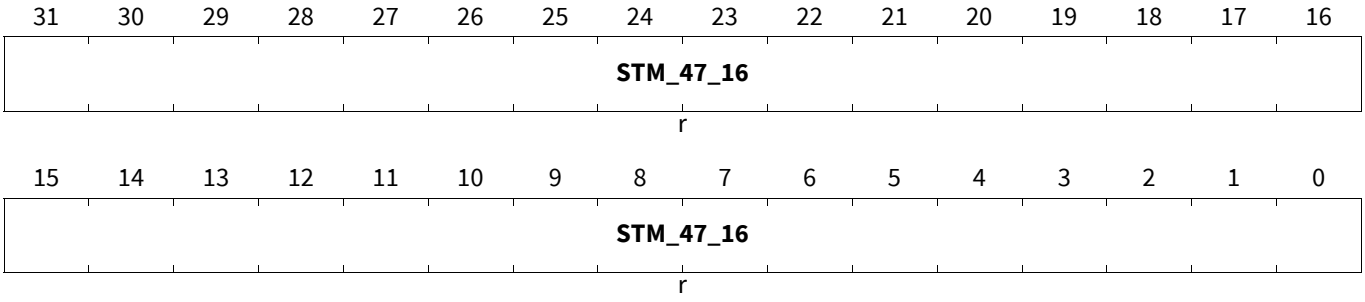
Timer Register 3 **($001C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
STM_43_12	31:0	r	**System Timer Bits [43:12] - STM[43:12]** This bit field contains bits [43:12] of the 64-bit STM.

Timer Register 4

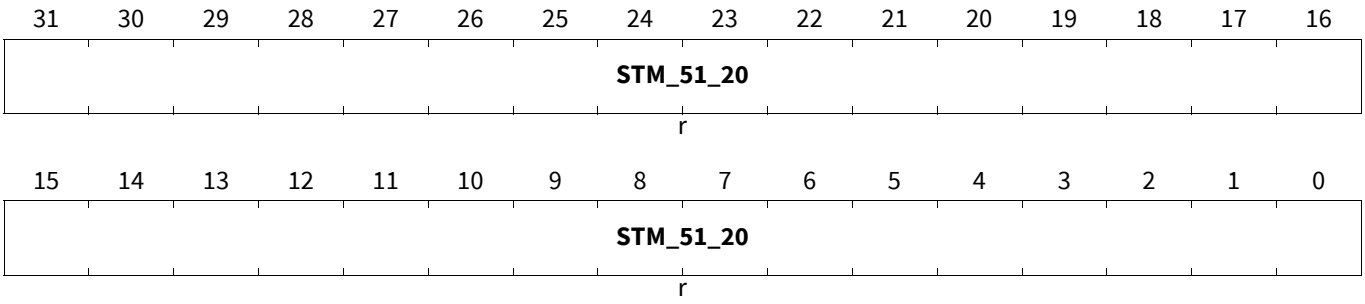
TIM4
Timer Register 4 **(0020_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
STM_47_16	31:0	r	**System Timer Bits [47:16] - STM[47:16]** This bit field contains bits [47:16] of the 64-bit STM.

Timer Register 5

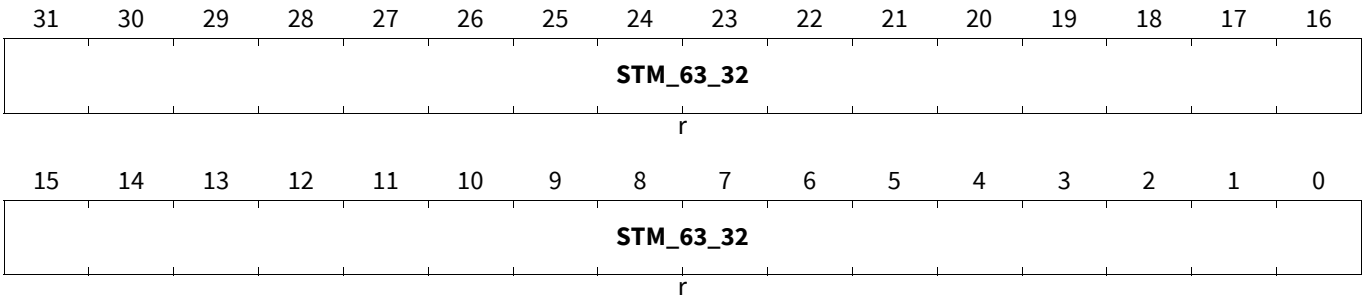
TIM5
Timer Register 5 **(0024_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
STM_51_20	31:0	r	**System Timer Bits [51:20] - STM[51:20]** This bit field contains bits [51:20] of the 64-bit STM.

Timer Register 6

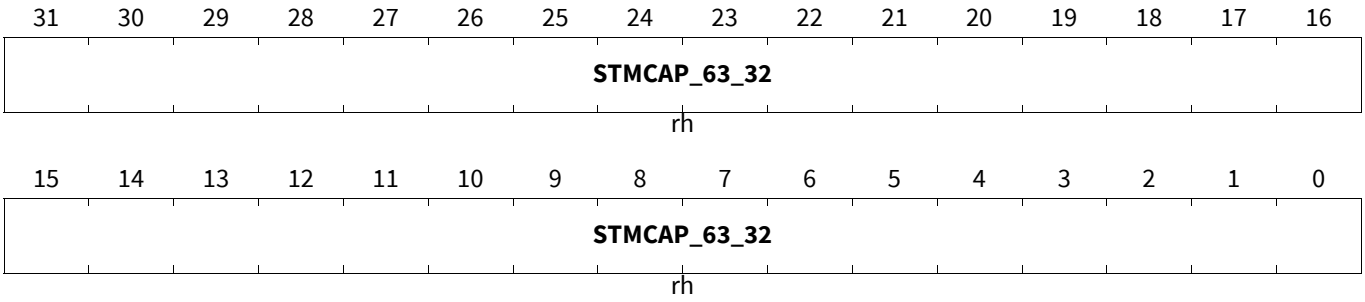
TIM6
Timer Register 6 **(0028_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
STM_63_32	31:0	r	**System Timer Bits [63:32] - STM[63:32]** This bit field contains bits [63:32] of the 64-bit STM.

Timer Capture Register

CAP
Timer Capture Register **($002C_H$)** **Application Reset Value: $0000\ 0000_H$**

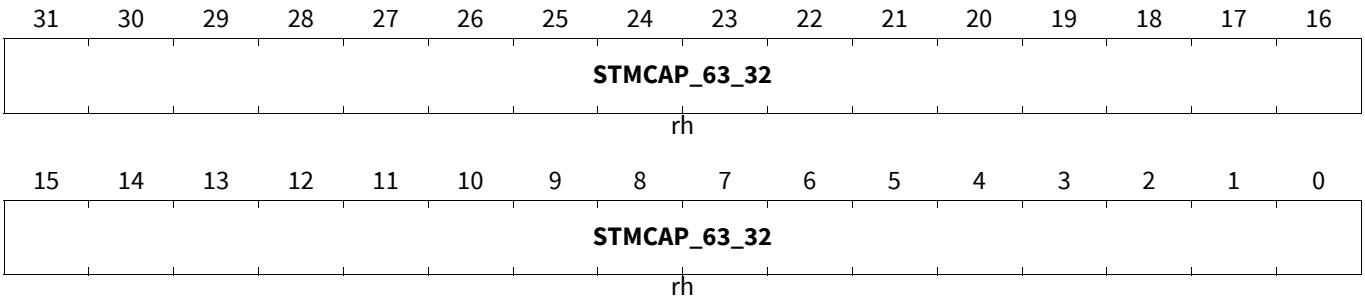


Field	Bits	Type	Description
STMCAP_63_32	31:0	rh	**Captured System Timer Bits [63:32] - STMCAP[63:32]** The capture register STMCAP always captures the STM bits [63:32] when one of the registers TIM0 to TIM6 or TIM0SV is read. This capture operation is performed in order to enable software to operate with a coherent value of all the 64 STM bits at one time stamp.This bit field contains bits [63:32] of the 64-bit STM. *Note: Reading register TIM0SV captures also the read value for register TIM6. In this way reading TIM0SV followed by CAPSV delivers the timer values for the first read request.*

Timer Capture Register Second View

CAPSV

Timer Capture Register Second View **(0054_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
STMCAP_63_32	31:0	rh	**Captured System Timer Bits [63:32] - STMCAP[63:32]** The capture register STMCAP always captures the STM bits [63:32] when one of the registers TIM0 to TIM6 or TIM0SV is read. This capture operation is performed in order to enable software to operate with a coherent value of all the 64 STM bits at one time stamp.This bit field contains bits [63:32] of the 64-bit STM. *Note: Reading register TIM0SV captures also the read value for register TIM6. In this way reading TIM0SV followed by CAPSV delivers the timer values for the first read request.*

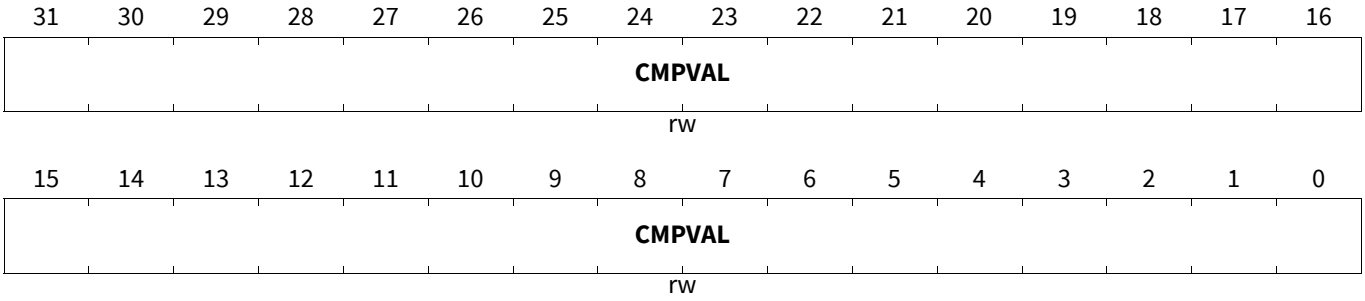
27.4.3 Compare Registers

The compare register CMPx holds up to 32-bits; its value is compared to the value of the STM.

Compare Register x

CMPx (x=0-1)

Compare Register x **(0030_H+x*4)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CMPVAL	31:0	rw	**Compare Value of Compare Register x** This bit field holds up to 32 bits of the compare value (right-adjusted).

Compare Match Control Register

The STM Compare Match Control Register controls the parameters of the compare logic.

CMCON

Compare Match Control Register **(0038_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**MSTART1**					**0**			**MSIZE1**				
r			rw					r			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**MSTART0**					**0**			**MSIZE0**				
r			rw					r			rw				

Field	Bits	Type	Description
MSIZE0	4:0	rw	**Compare Register Size for CMP0** This bit field determines the number of bits in register CMP0 (starting from bit 0) that are used for the compare operation with the System Timer. ... 00_H CMP0[0] used for compare operation 01_H CMP0[1:0] used for compare operation $1E_H$ CMP0[30:0] used for compare operation $1F_H$ CMP0[31:0] used for compare operation
MSTART0	12:8	rw	**Start Bit Location for CMP0** This bit field determines the lowest bit number of the 64-bit STM that is compared with the content of register CMP0 bit 0. The number of bits to be compared is defined by bit field MSIZE0. ... 00_H STM[0] is the lowest bit number 01_H STM[1] is the lowest bit number $1E_H$ STM[30] is the lowest bit number $1F_H$ STM[31] is the lowest bit number
MSIZE1	20:16	rw	**Compare Register Size for CMP1** This bit field determines the number of bits in register CMP1 (starting from bit 0) that are used for the compare operation with the System Timer. ... 00_H CMP1[0] used for compare operation 01_H CMP1[1:0] used for compare operation $1E_H$ CMP1[30:0] used for compare operation $1F_H$ CMP1[31:0] used for compare operation

Field	Bits	Type	Description
MSTART1	28:24	rw	**Start Bit Location for CMP1** This bit field determines the lowest bit number of the 64-bit STM that is compared with the content of register CMP1 bit 0. The number of bits to be compared is defined by bit field MSIZE1. ... 00_H STM[0] is the lowest bit number 01_H STM[1] is the lowest bit number $1E_H$ STM[30] is the lowest bit number $1F_H$ STM[31] is the lowest bit number
0	7:5, 15:13, 23:21, 31:29	r	**Reserved** Read as 0; should be written with 0.

27.4.4 Interrupt Registers

Interrupt Control Register

The two compare match interrupts of the STM are controlled by the STM Interrupt Control Register.

ICR
Interrupt Control Register **(003C$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									CMP1OS	CMP1IR	CMP1EN	0	CMP0OS	CMP0IR	CMP0EN
r									rw	rh	rw	r	rw	rh	rw

Field	Bits	Type	Description
CMP0EN	0	rw	**Compare Register CMP0 Interrupt Enable Control** This bit enables the compare match interrupt with compare register CMP0. 0_B Interrupt on compare match with CMP0 disabled 1_B Interrupt on compare match with CMP0 enabled
CMP0IR	1	rh	**Compare Register CMP0 Interrupt Request Flag** This bit indicates whether or not a compare match event of compare register CMP0 has occured. CMP0IR can be cleared by software and can be set by software, too (see ISCR register). After a STM reset operation, CMP0IR is immediately set as a result of a compare match event with the reset values of the STM and the compare registers CMP0. *Note: This flag does not gate the interrupt generation. i.e., even if this flag is not cleared, compare match event interrupts are forwarded if CMPxEN is set.* 0_B A compare match event has not been detected since the bit was last cleared. 1_B A compare match event has been detected.
CMP0OS	2	rw	**Compare Register CMP0 Interrupt Output Selection** This bit determines the interrupt output that is activated on a compare match event of compare register CMP0. 0_B Interrupt output STMIR0 selected 1_B Interrupt output STMIR1 selected

Field	Bits	Type	Description
CMP1EN	4	rw	**Compare Register CMP1 Interrupt Enable Control** This bit enables the compare match interrupt with compare register CMP1. 0_B Interrupt on compare match with CMP1 disabled 1_B Interrupt on compare match with CMP1 enabled
CMP1IR	5	rh	**Compare Register CMP1 Interrupt Request Flag** This bit indicates whether or not a compare match event of compare register CMP1 has occured. CMP1IR can be cleared by software and can be set by software, too (see ISCR register). After a STM reset operation, CMP1IR is immediately set as a result of a compare match event with the reset values of the STM and the compare register CMP1. *Note: This flag does not gate the interrupt generation. i.e., even if this flag is not cleared, compare match event interrupts are forwarded if CMPxEN is set.* 0_B A compare match event has not been detected since the bit was last cleared. 1_B A compare match event has been detected.
CMP1OS	6	rw	**Compare Register CMP1 Interrupt Output Selection** This bit determines the interrupt output that is activated on a compare match event of compare register CMP1. 0_B Interrupt output STMIR0 selected 1_B Interrupt output STMIR1 selected
0	3, 31:7	r	**Reserved** Read as 0; should be written with 0.

Interrupt Set/Clear Register

The bits in the STM Interrupt Set/Clear Register make it possible to set or cleared the compare match interrupt request status flags of register ICR.

Note: Reading register ISCR always returns 0000 0000$_H$.

ISCR

Interrupt Set/Clear Register (0040$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												CMP1IRS	CMP1IRR	CMP0IRS	CMP0IRR
r												w	w	w	w

Field	Bits	Type	Description
CMP0IRR	0	w	**Reset Compare Register CMP0 Interrupt Flag** 0_B Bit ICR.CMP0IR is not changed. 1_B Bit ICR.CMP0IR is cleared.
CMP0IRS	1	w	**Set Compare Register CMP0 Interrupt Flag** 0_B Bit ICR.CMP0IR is not changed. 1_B Bit ICR.CMP0IR is set. The state of bit CMP0IRR is “don’t care” in this case.
CMP1IRR	2	w	**Reset Compare Register CMP1 Interrupt Flag** 0_B Bit ICR.CMP1IR is not changed. 1_B Bit ICR.CMP1IR is cleared.
CMP1IRS	3	w	**Set Compare Register CMP1 Interrupt Flag** 0_B Bit ICR.CMP1IR is not changed. 1_B Bit ICR.CMP1IR is set. The state of bit CMP1IRR is “don’t care” in this case.
0	31:4	r	**Reserved** Read as 0; should be written with 0.

27.4.5 Interface Registers

OCDS Control and Status Register

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The OCS register can only be written when the OCDS is enabled.

If OCDS is being disabled, the OCS register value will not change.When OCDS is disabled the OCS suspend control is ineffective.

OCS

OCDS Control and Status Register (00E8$_H$) Debug Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		SUSSTA	SUS_P	SUS				0							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														0	
r														rw	

Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity of STMx to the suspend signal coming from CPUx (CPUxSUSOUT) 0_H Will not suspend 1_H Reserved, do not use this combination 2_H 64-bit counter will be stopped **others**, Reserved, do not use this combination
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	2:0	rw	**Reserved** Read as 0; must be written with 0.
0	23:3, 31:30	r	**Reserved** Read as 0; must be written with 0.

Table 2 Access Mode Restrictions of OCS sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **SUS_P**	rw	SUS	
(default)	r	SUS	

Access Enable Register 0

The Access Enable Register 0 controls write access for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B , …,EN31 -> TAG ID 011111B.

ACCEN0

Access Enable Register 0 **($00FC_H$)** **Application Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID master peripheral mapping).

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, …, EN31 -> TAG ID 111111B.

ACCEN1

Access Enable Register 1 **($00F8_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers related to the module kernel that should be reset (kernel 0 or kernel 1). In order support modules with two kernel the BPI_FPI provides two set of kernel reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing to it with ´1´.

KRST0

Kernel Reset Register 0 **(00F4$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RSTSTAT	RST
r														rh	rwh

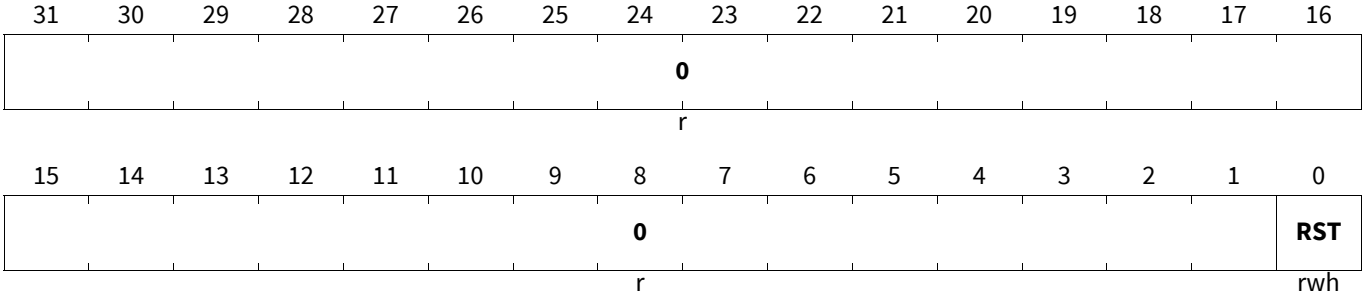
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the STM kernel. STM kernel registers related to the Debug Reset (Class 1) are not influenced. To reset the STM kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be cleared with the end of the BPI kernel reset sequence.

KRST1

Kernel Reset Register 1 **(00F0$_H$)** **Application Reset Value: 0000 0000$_H$**



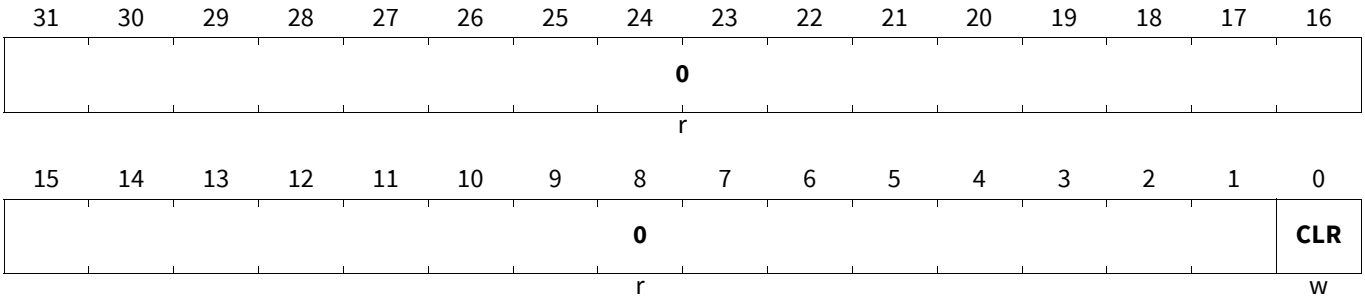
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Register Clear register is used to clear the Kernel Reset Status bit (<>_KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **(00EC$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

27.5 IO Interfaces

The table below list of interfaces of the STM to other modules.

Table 3 List of STM Interface Signals

Interface Signals	I/O	Description
sx_fpi		**FPI slave interface**
sx_irq_stm		**STM Interrupt Socket**
SR0_INT	out	**System Timer Service Request 0**
SR1_INT	out	**System Timer Service Request 1**

27.6 Revision History

Table 4 Revision History

Reference	Change to Previous Version	Comment
V9.2.3		
Page 23	Previous versions removed from revision history.	
V9.2.4		
Page 19	Description of bit field “SUS” of register “OCDS Control and Status Register” updated.	

28 Generic Timer Module (GTM)

GTM general Architecture

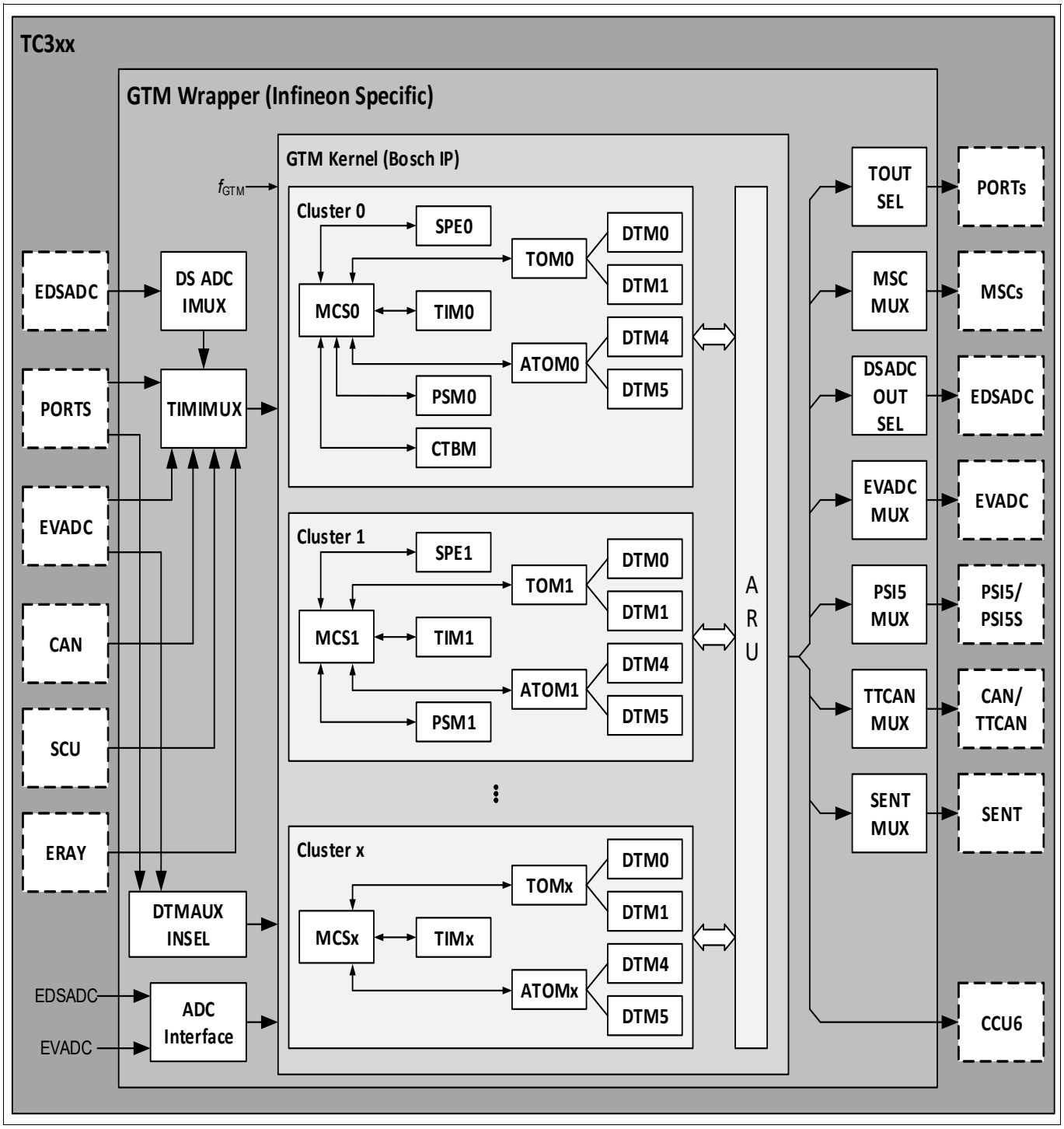


Figure 4 GTM Architecture block diagram

Note: The GTM system runs with the GTM global clock f_{GTM} (referred in this chapter as SYS_CLK)

28.1 Feature List

FEATURE LIST

The GTM module is comprised of two main parts:

- the GTM IP v3.1.5.1, designed by Bosch
  - The GTM IP consists of many different sub-modules, offering a wide-variety of functionality to address all the most common applications that are relevant for a timer module
- the GTM Wrapper[1] designed by Infineon

The following is a list of some of the most important GTM IP features found in the sub-modules:

- Central/Edge aligned PWM Generation (TOM/ATOM)
- Dedicated module for asymmetric dead time generation (DTM)
- Dedicated support for DC-DC phase shift operation (DTM)
- Logical combination of digital signals (DTM)
- Provide common time base for system (TBU)
- Multiple capture/compare of external signals and combination with time stamps (TIM/ATOM/TBU)
- Complex digital signals generation (PSM, ATOM, MCS)
- Digital signals filtering and characterization (TIM)
- Input signal measurement and timeout detection (TIM)
- BLDC motor control using Block Commutation Mode (SPE, TIM, TOM)
- Engine angle clock for engine management applications (DPLL, TIM, MAP, TBU)
- Injection/Ignition pulses generation with dedicated HW support (DPLL, ATOM, MCS)
- Programmable RISC-like cores to offload the CPU (MCS)
- Automatic data sharing between GTM modules without CPU/DMA (ARU,BRC)
- Configurable operating frequency up to 200MHz (CCM,CMU)

28.1.1 Delta to AURIX

Backward compatibility

- The GTM IP v3.1.5.1 is fully backward compatible with previous generations
- The GTM wrapper is not fully backward compatible
  - Some parts of the wrapper has been changed to increase the GTM flexibility and to address the higher number of connectivities required by the AURIX second generation

GTM IP differences

The main differences between GTM2.x and GTM3.1.x are the following:

- Maximum operation frequency extended up to 200MHz[2]
- GTM architecture divided in clusters
- Enhanced the MCS RISC-like processor enhanced with new instructions (i.e. MUL,DIV)

1) For the GTM Wrapper details, please refer to the GTM Implementation chapter
2) Only for the first five clusters

- Every MCS can access all the other GTM modules' registers inside the same cluster
- Every MCS can get interrupts from all the other GTM modules inside the same cluster
- MCS memory write protection against CPU/DMA accesses and other MCS
- Added new configurable scheduling schemes to the ARU
- Added the emergency switch off feature to the DTM modules
- Improved the DPLL ticks generation and the bi-directional sensors support
- Added new ADC Interface to the MCS
- Added new TBU modulo counter (engine angle clock)
- Added new up-down counter mode to (A)TOM modules
- Added new scheduling schemes to the MCS

GTM Wrapper differences

With the TC3xx devices, there are significant changes with the Wrapper:

- Increased the number of selectable GTM output resources connected to each Port (from 4 to 12)
- Added three more triggers towards ADC (from 2 to 5)
- Implemented a new multiplexer to connect Port and ADC signals to DTM
- Created a new interface to connect all the EVADC and EDSADC result registers to the MCS

28.2 Overview

GTM Kernel Architecture

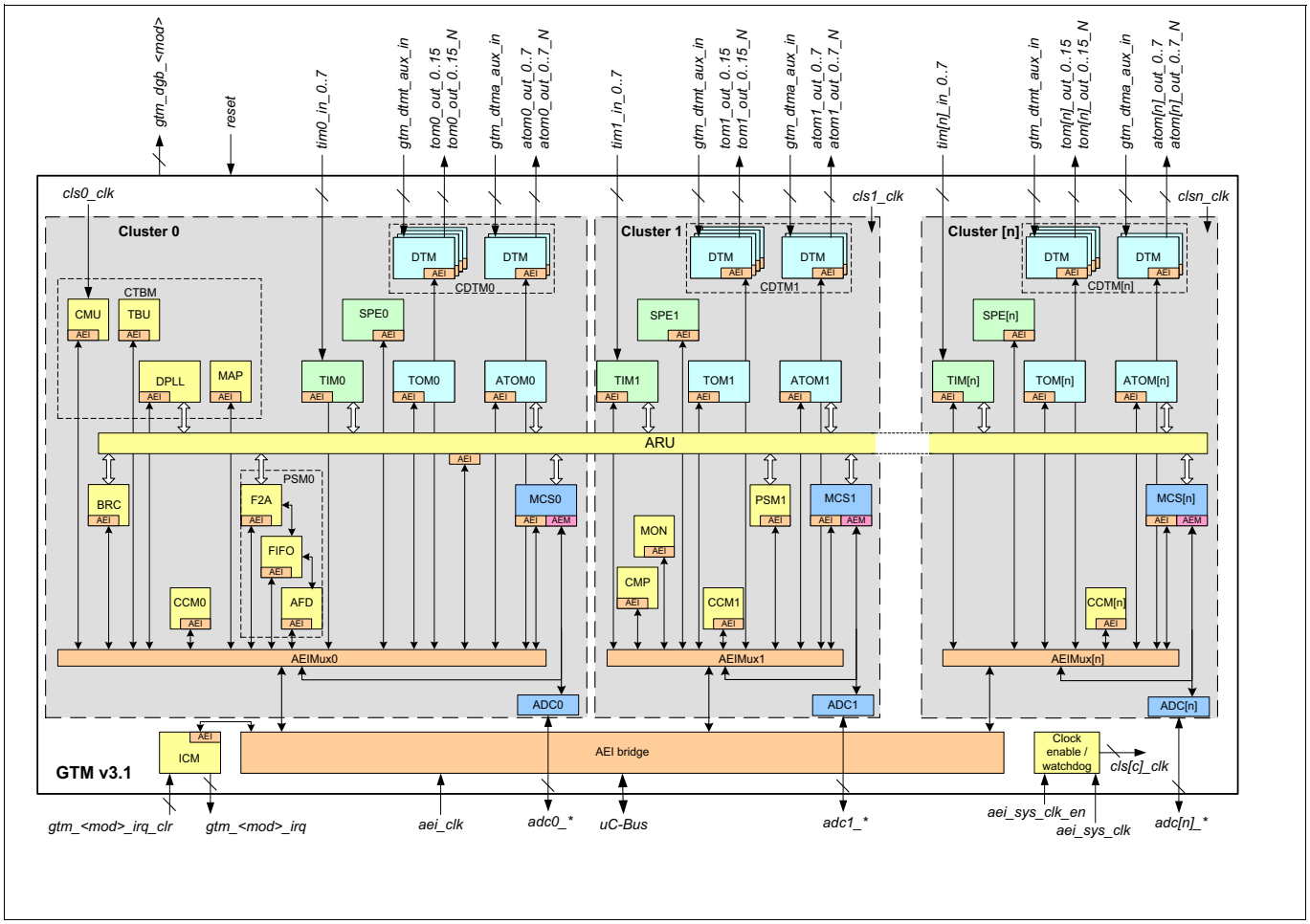


Figure 5 GTM Architecture Block Diagram

28.3 Generic Timer Module (GTM)

This document is based on the following GTM specification of the Robert Bosch GmbH:

- Version: release-v3.1.5.1-2016.03.24
- Date: 24 March 2016

28.3.1 Overview

This document is the specification for the Generic Timer Module (GTM). It contains a module framework with sub-modules of different functionality. These sub-modules can be combined in a configurable manner to form a complex timer module that serves different application domains and different classes within one application domain. Because of this scalability and configurability the timer is called generic.

The scalability and configurability is reached with an architecture philosophy where dedicated hardware sub-modules are located around a central routing unit (called Advanced Routing Unit (ARU)). The ARU can connect the sub-modules in a flexible manner. The connectivity is software programmable and can be configured during runtime.

Nevertheless, the GTM is designed to unload the CPU or a peripheral core from a high interrupt load. Most of the tasks inside the GTM can run -once setup by an external CPU- independent and in parallel to the software. There

may be special situations, where the CPU has to take action but the goal of the GTM design was to reduce these situations to a minimum.

The hardware sub-modules have dedicated functionality's, e.g. there are timer input modules where incoming signals can be captured and characterized together with a notion of time. By combination of several sub-modules through the ARU complex functions can be established. E.g. the signals characterized at an input module can be routed to a signal processing unit where an intermediate value about the incoming signal frequency can be calculated.

The modules that help to implement such complex functions are called *infrastructure components* further on. These components are present in all GTM variants. However, the number of these components may vary from device to device.

Other sub-modules have a more general architecture and can fulfill typical timer functions, e.g. there are PWM generation units. The third classes of sub-modules are those fulfilling a dedicated functionality for a certain application domain, e.g. the DPLL serves engine management applications. A fourth group of sub-modules is responsible for supporting the implementation of safety functions to fulfill a defined safety level. The module ICM is responsible for interrupt services and defines the fifth group.

Each GTM is build up therefore with sub-modules coming from those four groups. The application class is defined by the amount of components of those sub-modules integrated into the implemented GTM.

28.3.2 Document Structure

The structure of this document is motivated out of the aforementioned sub-module classes. **Section 28.4** describes the dedicated GTM implementation this specification is written for. It gives an overview about the implemented sub-modules.

The following sections **Section 28.5** up to **Section 28.12** deals with the so called infrastructure components for routing, clock management and common time base functions. Sections **Section 28.13** to **Section 28.16** describe the signal input and output modules while the following **Section 28.17** explains the signal processing and generation sub-module with **Section 28.18** its memory configuration. The next sections **Section 28.19** to **Section 28.21** provides a detailed description of application specific modules like the MAP, DPLL and SPE. The last sections **Section 28.23** to **Section 28.24** provide to safety related modules (not part of the Infineon safety manual) like CMP and MON sub-modules. **Section 28.22** describes a module that bundles several interrupts coming from the other sub-modules and connect them to the outside world.

Note: Infineon: CMP and MON are not part of the safety manual. As safety measure the IOM is provided in combination with GTP1/2 or CCU6.

Table 5 Sub-module groups

Chapter	Sub-module	Group
Section 28.5	Advanced Routing Unit (ARU)	Infrastructure components
Section 28.6	Broadcast Module (BRC)	Infrastructure components
Section 28.7	First In First Out Module (FIFO)	Infrastructure components
Section 28.8	AEI-to-FIFO Data Interface (AFD)	Infrastructure components
Section 28.9	FIFO-to-ARU Interface (F2A)	Infrastructure components
Section 28.10	Clock Management Unit (CMU)	Infrastructure components
Section 28.11	Cluster Configuration Module (CCM)	Infrastructure components
Section 28.12	Time Base Unit (TBU)	Infrastructure components
Section 28.13	Timer Input Module (TIM)	IO Modules
Section 28.14	Timer Output Module (TOM)	IO Modules
Section 28.15	ARU-connected Timer Output Module (ATOM)	IO Modules
Section 28.16	Dead Time Module (DTM)	IO Modules
Section 28.17	Multi Channel Sequencer (MCS)	Signal generation and processing
Section 28.18	Memory Configuration (MCFG)	Memory for signal generation and processing
Section 28.19	TIM0 Input Mapping Module (MAP)	Dedicated

Table 5 Sub-module groups (cont'd)

Chapter	Sub-module	Group
Section 28.20	Digital PLL (DPLL)	Dedicated
Section 28.21	Sensor Pattern Evaluation Module (SPE)	BLDC support
Section 28.22	Interrupt Concentrator Module (ICM)	Interrupt services
Section 28.23	Output Compare Unit (CMP)	Safety features (not part of the Infineon safety manual)
Section 28.24	Monitoring Unit (MON)	Safety features (not part of the Infineon safety manual)

28.4 GTM Architecture

28.4.1 Overview

As already mentioned in the Introduction the GTM forms a generic timer platform that serves different application domains and different classes within these application domains. Depending on these multiple requirements of application domains multiple device configurations with different number of sub-modules (i.e. ATOM, BRC, MCS, PSM, SPE, TIM, TOM, DTM) and different number of channel per sub-module (if applicable) are possible.The device dependent configuration (i.e. the number of sub-modules) is listed in the device specific appendix. The Parameter Storage Module (PSM) is only a virtual hierarchy and consists of the sub-module F2A, FIFO and AFD.The Cluster Dead Time Module (CDTM) is also a virtual hierarchy and consists of up to six DTM modules. It depends on the GTM device configuration which of the six DTM instances are available. Please refer to device specific appendix for list of available DTM instances. In general, the first four DTM modules inside a CDTM[n] hierarchy are connected to the outputs of the TOM instance [n] of the cluster [n], the other two DTM instances are connected to the outputs of the ATOM instance [n] of this cluster [n].

The cluster view of a GTM_IP architecture is depicted in **Figure 6**. This is a generic figure which shows a possible GTM-IP device configuration.

The device dependent configuration (i.e. the count of sub-modules and channels per sub-module) is listed in the device specific appendix.

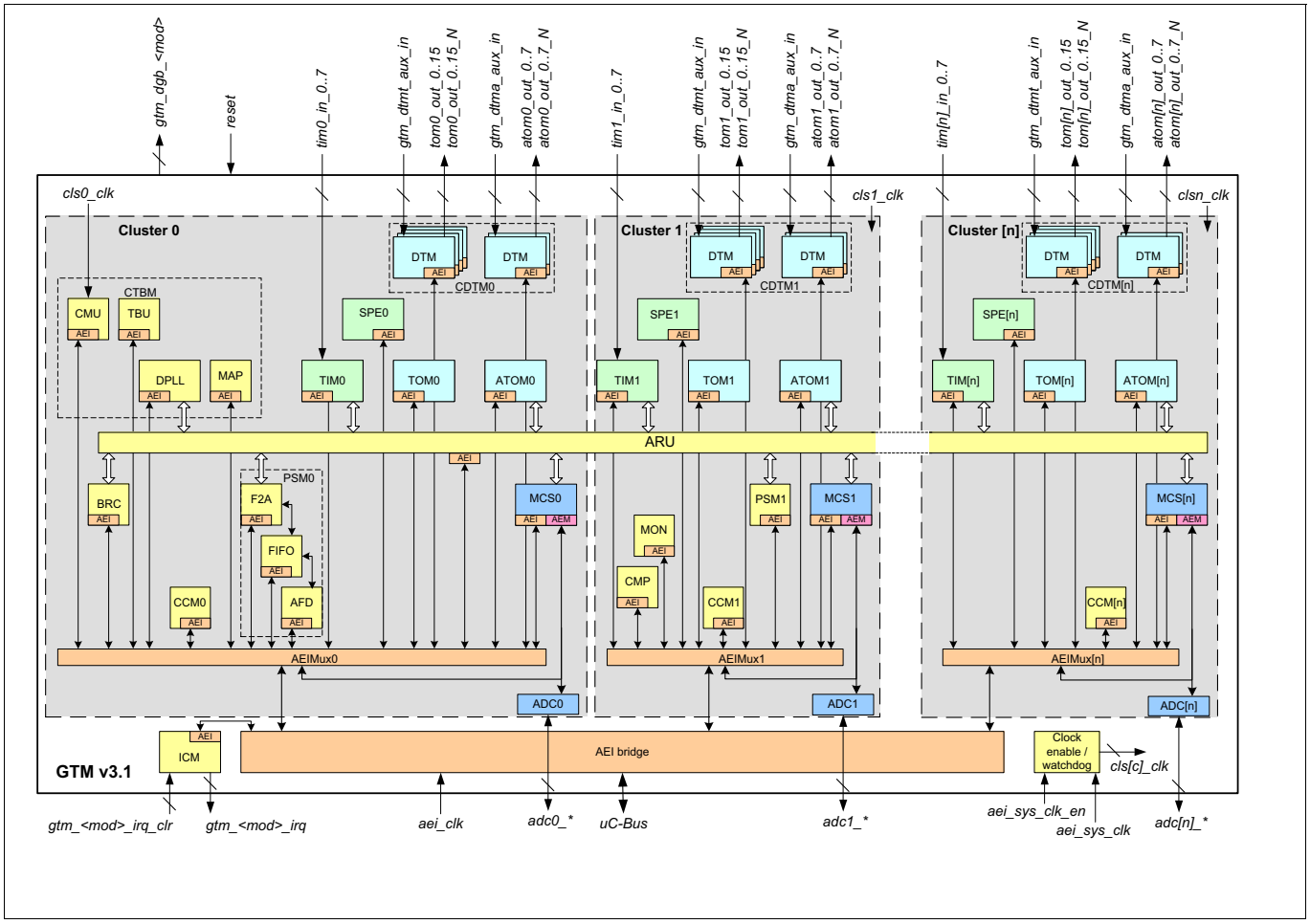


Figure 6 GTM Architecture Block Diagram

The GTM is divided in multiple clusters 0…n. A certain amount of modules exist in each cluster. The operating frequency of a cluster can be configured to OFF, aei_sys_clk or aei_sys_clk/2. The clock enable generation can be implemented internal to the GTM_IP or external. In case of an external enable generation aei_sys_clk_en is used to generate the internal clocks. In addition an enable watchdog is implemented to monitor the correctness of the external applied enable signals aei_sys_clk_en.

The central component of the GTM is the Advanced Routing Unit (ARU) where most of the sub-modules are located around and connected to. This ARU forms together with the Broadcast (BRC) and the Parameter Storage Module (PSM) the infrastructure part of the GTM. The ARU is able to route data from a connected source sub-module to a connected destination sub-module. The routing is done in a deterministic manner with a round-robin scheduling scheme of connected channels which receive data from ARU and with a worst case round-trip time.

The routed data word size of the ARU is 53 bit. The data word can logically be split into three parts. These parts are shown in **Figure 7**. Bits 0 to 23 and bits 24 to 47 typically hold data for the operation registers of the GTM. This can be, for example, the duty cycle and period duration of a measured PWM input signal or the output characteristic of an output PWM to be generated. Another possible content of Data0 and Data1 can be two 24 bit values of the GTM time bases TBU_TS0, TBU_TS1 and TBU_TS2. Bits 48 to 52 can contain control bits to send control information from one sub-module to another. These ARU Control Bits (ACB) can have a different meaning for different sub-modules.It is also possible to route data from a source to a destination and the destination can act later on as source for another destination. These routes through the GTM are further on called *data streams*. For a detailed description of the ARU sub-module please refer to the ARU chapter.

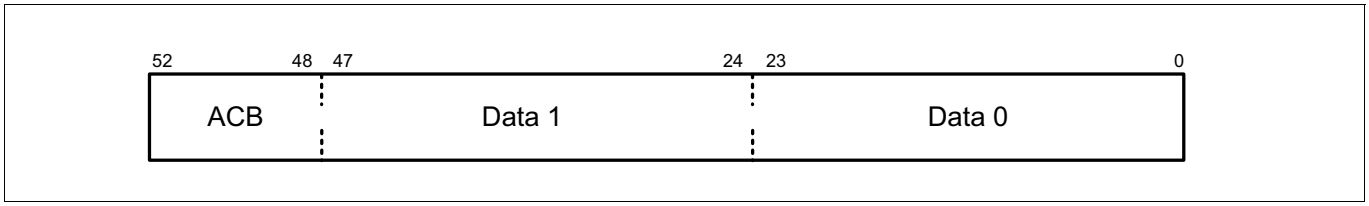


Figure 7 ARU Data Word Description

The BRC is able to distribute data from one source module to more than one destination modules connected to the ARU. The PSM sub-module consists of three sub-units, the AEI-to-FIFO Data Interface (AFD), FIFO-to-ARU Interface (F2A) and the FIFO itself. The PSM can serve as a data storage for incoming data characteristics or as parameter storage for outgoing data. This data is stored in a RAM that is logically located inside the FIFO sub-unit, but physically the RAM is implemented and integrated by the silicon vendor with his RAM implementation technology. Therefore, the GTM provides the interface to the RAM at its module boundary. The AFD sub-unit is the interface between the FIFO and the GTM SoC system bus interface AEI (see **Section 28.4.2.1** for detailed discussion). The F2A sub-unit is the interface between the FIFO sub-unit and the ARU.

Signals are transferred into the GTM at the Timer Input Modules (TIM). These modules are able to filter the input signals and annotate additional information. Each channel is for example able to measure pulse high or low times and the period of a PWM signal in parallel and route the values to ARU for further processing. The internal operation registers of the TIM sub-module are 24 bits wide.

The Clock Management Unit (CMU) serves up to 13 different clocks for the GTM and up to three external clock pins *GTM_ECLK0…2*. It acts as a clock divider for the system clock. The counters implemented inside other sub-modules are typically driven from this sub-module. Please note, that the CMU clocks are implemented as enable signals for the counters while the whole system runs with the GTM global clock *SYS_CLK*.[1)] This global clock typically corresponds to the micro controller bus clock the GTM-IP is connected to and should not exceed 100MHz because of the power dissipation of the used transistors where the GTM is implemented with.

The TBU provides up to three independent common time bases for the GTM. In general, the number of time bases depends on the implemented device. If three time bases are implemented, two of these time bases can also be clocked with the digital PLL (DPLL) *sub_inc1*c and *sub_inc2c* outputs. The DPLL generates the higher frequent clock signals *sub_inc1*, *sub_inc2*, *sub_inc1c* and *sub_inc2c* on behalf of the frequencies of up to two input signals. These two input signals can be selected out of six incoming signals from the TIM0 sub-module. In this sub-module the incoming signals are filtered and transferred to the MAP sub-module where two of these six signals are selected for further processing inside the DPLL.

Signal outputs are generated with the Dead Time Module (DTM), Timer Output Modules (TOM) and the ARU-connected TOMs (ATOM). Each TOM channel is able to generate a PWM signal at its output. Because of the integrated shadow register even the generation of complex PWM outputs is possible with the TOM channels by serving the parameters with the CPU. It is possible to trigger TOM channels for a successor TOM sub-module through a trigger line between TOM(x)_CH(15) and TOM(x+1)_CH(0). But to avoid long trigger paths the GTM integrator can configure after which TOM sub-module instance a register is placed into the trigger signal chain. Each register results in one SYS_CLK cycle delay of the trigger signal. Please refer to device specification of silicon vendor for unregistered trigger chain length.

In addition, each TOM sub-module can integrate functions to drive one BLDC engine. This BLDC support is established together with the TIM and Sensor Pattern Evaluation (SPE) sub-module.

The ATOMs offer the additional functionality to generate complex output signals without CPU interaction by serving these complex waveform characteristics by other sub-modules that are connected to the ARU like the PSM or Multi Channel Sequencer (MCS). While the internal operation and shadow registers of the TOM channels

1) SYS_CLK = f_{GTM} clock provided by the GTM wrapper. f_{GTM} max value is $2xf_{SPB}$ = 200MHz

are 16 bit wide, the operation and shadow registers of the ATOM channels are 24 bit wide to have a higher resolution and to have the opportunity to compare against time base values coming from the TBU.

It is possible to trigger ATOM channels for a successor ATOM sub-module through a trigger line between ATOM(x)_CH(7) and ATOM(x+1)_CH(0). But to avoid long trigger paths the GTM integrator can configure after which ATOM sub-module instance a register is placed into the trigger signal chain. Each register results in one SYS_CLK cycle delay of the trigger signal. Please refer to device specification of silicon vendor for unregistered trigger chain length.

Together with the MCS the ATOM is able to generate an arbitrary predefined output sequence at the GTM output pins. The output sequence is defined by instructions located in RAM connected to the MCS sub-module. The instructions define the points were an output signal should change or to react on other signal inputs. The output points can be one or two time stamps (or even angle stamp in case of an engine management system) provided by the TBU. Since the MCS is able to read data from the ARU it is also able to operate on incoming data routed from the TIM. Additionally, the MCS can process data that is located in its connected RAMs. The MCS RAM is located logically inside the MCS while the silicon vendor has to implement its own RAM technology there.

The two modules Compare Module (CMP) and Monitor Module (MON) implement safety related features. The CMP compares two output channels of the DTM and sends the result to the MON sub-module were the error is signaled to the CPU. The MON module is also able to monitor the ARU and CMU activities.

In the described implementation the sub-modules of the GTM have a huge amount of different interrupt sources. These interrupt sources are grouped and concentrated by the Interrupt Concentrator Module (ICM) to form a much easier manageable bunch of interrupts that are visible outside of the GTM.

On the GTM top level there are some configurable signal connections from the signal output of the DTM modules to the input signals of the TIM modules.

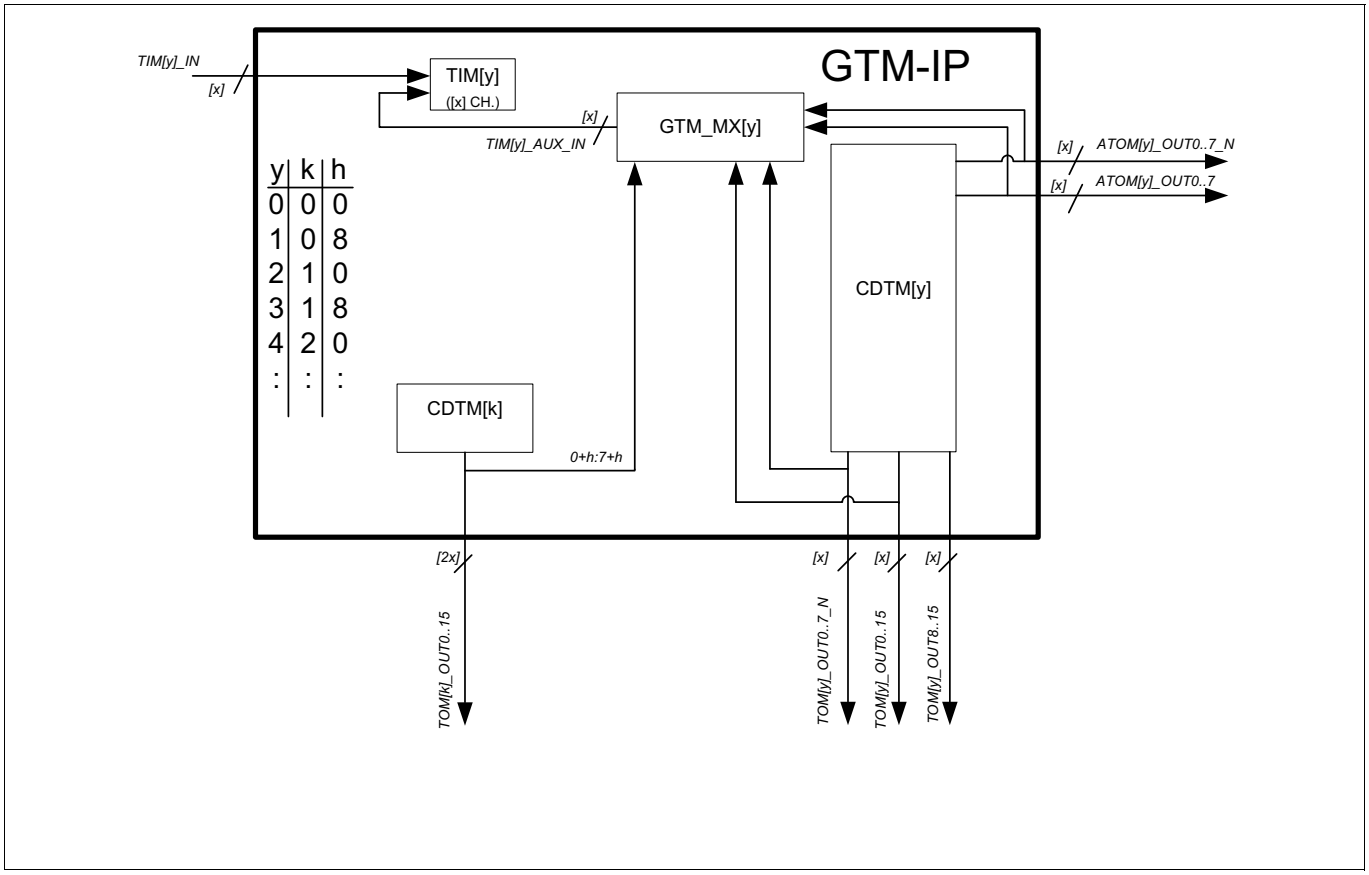


Figure 8 GTM signal multiplex

The next diagram gives an overview of the connectivity for different configuration of GTM global bit **SRC_IN_MUX** of register **GTM_CFG** and the cluster configuration register **CCM[y]_TIM_AUX_IN_SRC**. The source selection is defined per channel with the bit **SRC_CH[x]** and **SEL_OUT_N_CH[x]** in the register **CCM[y]_TIM_AUX_IN_SRC**.

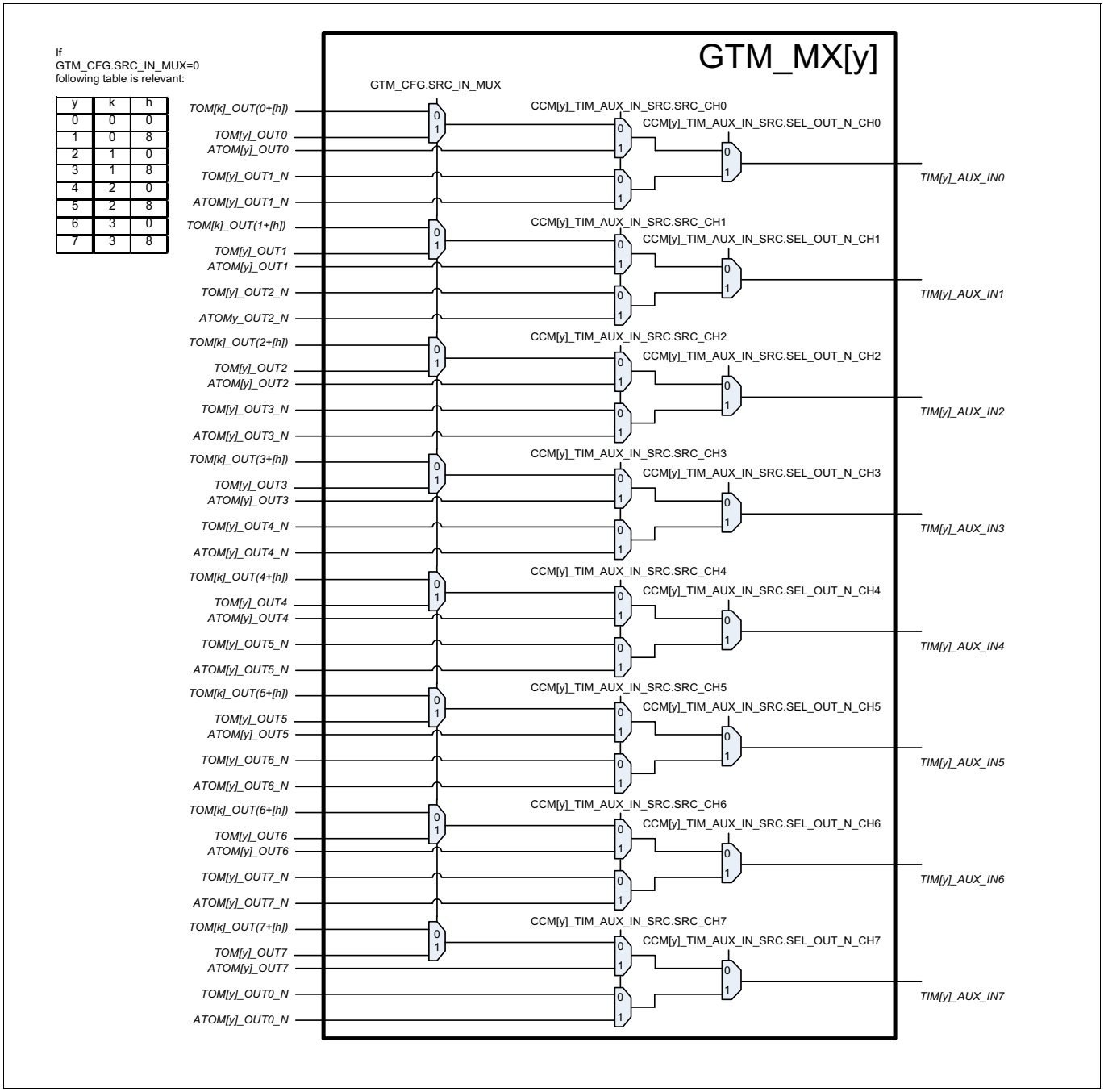


Figure 9 TIM auxiliary input multiplexing

The trigger out of TIM (i.e. the signals TIM[i]_EXT_CAPTURE(7:0) of each TIM instance i) are routed to ATOM instance [i] and TOM instance [i] with i=0…cITIM-1 (cITIM defines the number of available TIM instances, please refer to device specific device specific appendix).This TIM trigger can be used to trigger inside the ATOM or TOM instance either a channel or the global control register of AGC or TGC0/TGC1 unit.

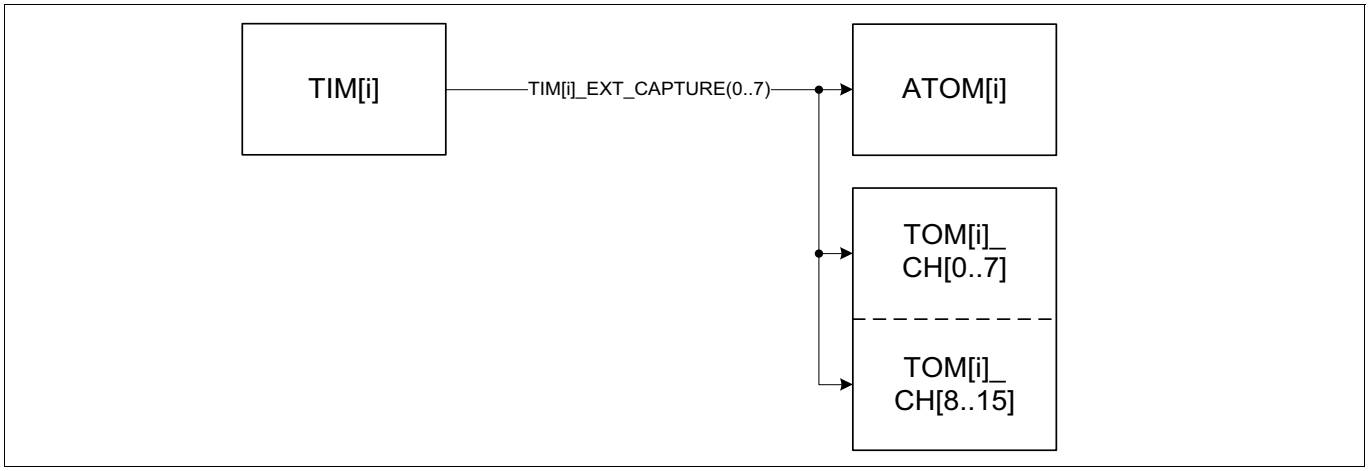


Figure 10 TIM external capture forwarding to TOM and ATOM

The trigger out of TIM (i.e. the signals TIM[i]_EXT_CAPTURE(7:0) of each TIM instance i) are additionally routed to the MCS instance [i]. This trigger forwarding can be enabled by register CCM[i]_EXT_CAP_EN.

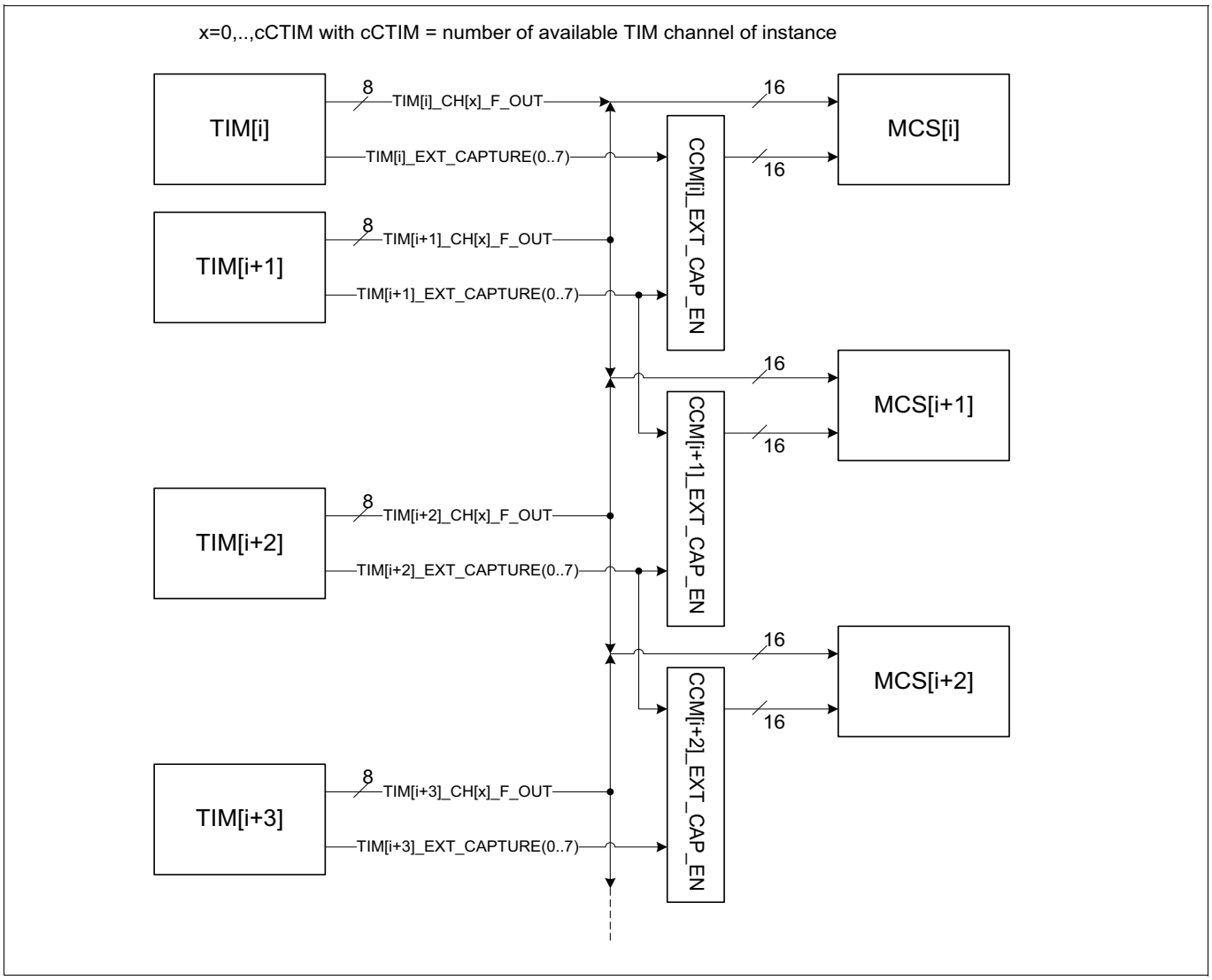


Figure 11 TIM to MCS signal forwarding

28.4.2 GTM Interfaces

In general the GTM can be divided into four interface groups. Two interface groups represent the ports of the GTM where incoming signals are assembled and outgoing signals are created. These interfaces are therefore connected to the GTM input sub-module TIM and to the GTM output sub-modules DTM.

Another interface is the bus interface where the GTM can be connected to the SoC system bus. This generic bus interface is described in more detail in **Section 28.4.2.1**. The last interface is the interrupt controller interface. The GTM provides several interrupt lines coming from the various sub-modules. These interrupt lines are concentrated inside the ICM and have to be adapted to the dedicated micro controller environment where each interrupt handling can look different. The interrupt concept is described in more detail in **Section 28.4.5**.

28.4.2.1 GTM Generic Bus Interface (AEI)

The GTM is equipped with a generic bus interface that can be widely adapted to different SoC bus systems. This generic bus interface is called AE-Interface (AEI). The adaptation of the AEI to SoC buses is typically done with a bridge module translating the AEI signals to the SoC bus signals of the silicon vendor. The AEI bus signals are depicted in the following table:

Table 6 AEI bus signals

Signal name	I/O	Description	Bit width
AEI_SEL	I	GTM select line	1
AEI_ADDR	I	GTM address	32
AEI_PIPE	I	AEI Address phase signal	1
AEI_W1R0	I	Read/Write access	1
AEI_WDATA	I	Write data bus	32
AEI_RDATA	O	Read data bus	32
AEI_READY	O	Data ready signal	1
AEI_STATUS	O	AEI Access status	2

The AEI Status Signal may drive one of the following values:

Table 7 AEI Status Signal

AEI_STATUS	Description
0b00	No Error
0b01	Illegal Byte Addressing
0b10	Illegal Address Access
0b11	Unsupported Address

The signal value 0b00 is returned if no error occurred during AEI access.

The signal value 0b01 is returned if the bus address is not an integer multiple of 4 (byte addressing).

The signal value 0b11 is returned if the address is not handled in the GTM.

The signal value 0b10 is returned if the written register is a protected register (e.g. protected by bit RF_PROT) or if the register is temporarily not writable because of sub-module internal state or the clock of the relevant cluster is disabled.

In case of an illegal write access signaled by status 0b10 the register will not be modified.

Reading registers will never return status 0b10.

Note: Exception for register CMU_CLK_CTRL. In case of write access signalled by aei_status 0b10 the register will be modified each completely disabled bit.

The detailed list of register addresses with return status 0b10 can be found in the appendix.

28.4.2.2 GTM Multi-master and multitasking support

To support multi-master and multi-task access to the registers of the GTM a dedicated write-access scheme is used for critical control bits inside the IP that need such a mechanism. This can be for example a shared register where more than one channel can be controlled globally by one register write access. Such register bits are implemented inside the GTM with a double bit mechanism, where the writing of 0b00 and 0b11 has no effect on the register bit and where 0b01 sets the bit and 0b10 resets the bit. If the CPU wants to read the status of the bit it always gets a 0b00 if the bit is reset and it gets an 0b11 if the bit is set.

28.4.3 ARU Routing Concept

One central concept of the GTM is the routing mechanism of the ARU sub-module for data streams. Each data word transferred between the ARU and its connected sub-module is 53 bit wide. It is important to understand this concept in order to use the resources of the GTM effectively. Each module that is connected to the ARU may provide an arbitrary number of ARU write channels and an arbitrary number of ARU read channels. In the following, the ARU write channels are named data sources and the ARU read channels are named data destinations.

The concept of the ARU intends to provide a flexible and resource efficient way for connecting any data source to an arbitrary data destination. In order to save resource costs, the ARU does not implement a switch matrix, but it implements a data router with serialized connectivity providing the same interconnection flexibility. **Figure 12** shows the ARU data routing principle. Data sources are marked with a green rectangle and the data destinations are marked with yellow rectangles. The dashed lines in the ARU depict the configurable connections between data sources and data destinations. A connection between a data source and a data destination is also called a data stream.

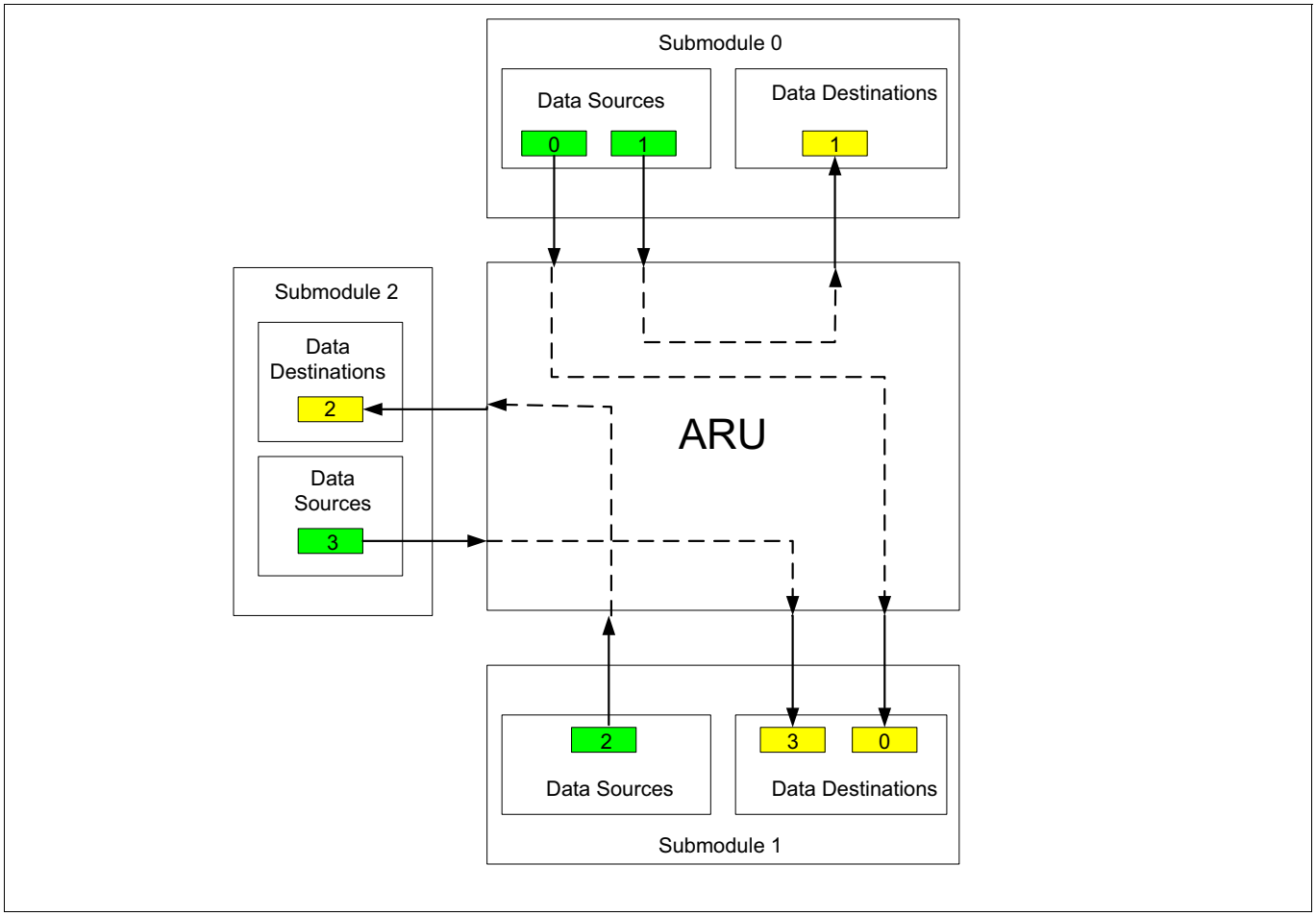


Figure 12 Principle of data routing using ARU

The configuration of the data streams is realized according to the following manner: Each data source has its fixed and unique source address: The ARU read ID.The fixed address of each data source is pointed out by the numbers in the green boxes of **Figure 12**. The address definitions of all available data sources in the GTM can be obtained from the product specific appendix. The connection from a specific data source to a specific data destination is defined by configuring the corresponding address of a data source (i.e. the ARU read ID) in the desired data destination. The configured address of each data destination is pointed out by the numbers in the yellow boxes of **Figure 12**.

Normally, the destination is idle and waits for data from the source. If the source offers new data, the destination does a destructive read, processes the data and goes idle again. The same data is never read twice.

There is one sub-module for which this destructive read access does not hold. This is the BRC sub-module configured in Maximum Throughput Mode (MTM). For a detailed description of this module please refer to chapter “Broadcast Module (BRC)”.

The functionality of the ARU is as follows: The ARU sequentially polls the data destinations of the connected modules in a round-robin order. If a data destination requests new data from its configured data source and the data source has data available, the ARU delivers the data to the destination and it informs both, the data source and destination that the data is transferred. The data source marks the delivered ARU data as invalid which means that the destination consumed the data.

It should be noted that each data source should only be connected to a single data destination. This is because the destinations consume the data. If two destinations would reference the same source one destination would consume the data before the other destination could consume it. Since the data transfers are blocking, the second destination would block until it receives new data from the source. If a data source should be connected

to more than one data destination the sub-module Broadcast (BRC) has to be used. On the other hand, the transfer from a data source to the ARU is also blocking, which means that the source channel can only provide new data to the ARU when an old data word is consumed by a destination. In order to speed up the process of data transfers, the ARU handles two different data destinations in parallel.

Following table gives an overview about the number of data destinations and data sources of each GTM instance type.

Table 8 ARU source and destination address count per instance

Sub-module	Number of data sources per instance	Number of data destinations per instance
ARU	1	0
DPLL	24	24
TIM	8	0
MCS	24	8
BRC	22	12
TOM	0	0
ATOM	8	8
DTM	0	0
PSM	8	8
ICM	0	0
CMP	0	0
MON	0	0
CCM	0	0

28.4.3.1 ARU Round Trip Time

The ARU uses a round-robin arbitration scheme with a fixed round trip time for all connected data destinations. This means that the time between two adjacent read requests resulting from a data destination channel always takes the round trip time, independently if the read request succeeds or fails.

28.4.3.2 ARU Blocking Mechanism

Another important concept of the ARU is its blocking mechanism that is implemented for transferring data from a data source to a data destination. This mechanism is used by ARU connected sub-modules to synchronize the sub-modules to the routed data streams. **Figure 13** explains the blocking mechanism.

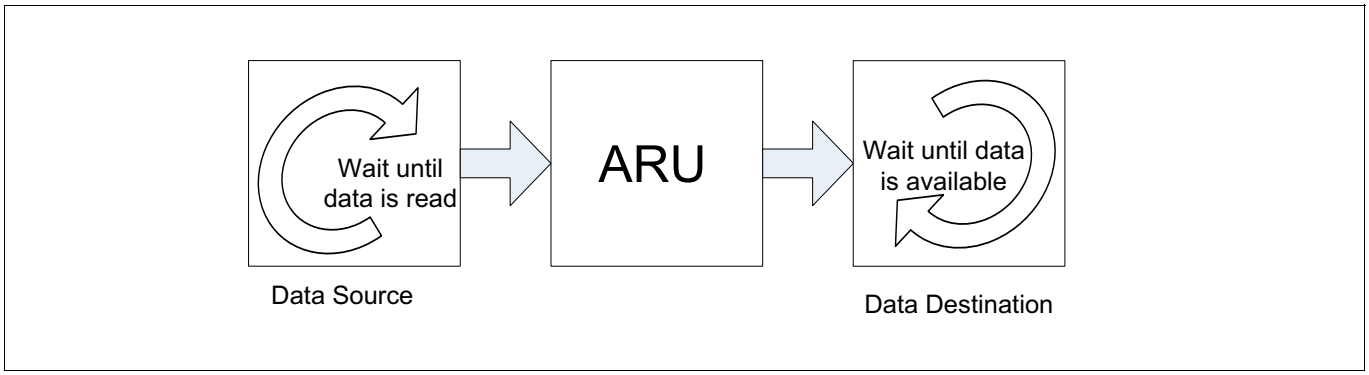


Figure 13 Graphical representation of ARU blocking mechanism

If a data destination requests data from a data source over the ARU but the data source does not have any data yet, it has to wait until the data source provides new data. In this case the sub-module that owns the data destination may perform other tasks. When a data source produces new data faster than a data destination can consume the data the source raises an error interrupt and signals that the data could not be delivered in time. The new data is marked as valid for further transfers and the old data is overwritten.

In any case the round trip time for the ARU has a fixed reset value for a specific device configuration. The end value of the round-trip counter can be changed with a configuration register **ARU_CADDR_END** inside the ARU. For more details see the ARU specific chapter.

It is possible to reset the ARU round-trip counter **ARU_CADDR** manually synchronous to CMU clock enable from configuration register inside CMU module. Please refer to CMU specific chapter for more details.

One exception is the BRC sub-module when configured in Maximal Throughput Mode. Please refer to Broadcast Module chapter for a detailed description.

28.4.4 GTM Clock and Time Base Management (CTBM)

Inside the GTM several sub-units are involved in the clock and time base management of the whole GTM. **Section 28.4.4.1** shows the connections and sub blocks involved in these tasks. The sub blocks involved are called Clock and Time Base Management (CTBM) modules further on.

28.4.4.1 GTM Clock and time base management architecture

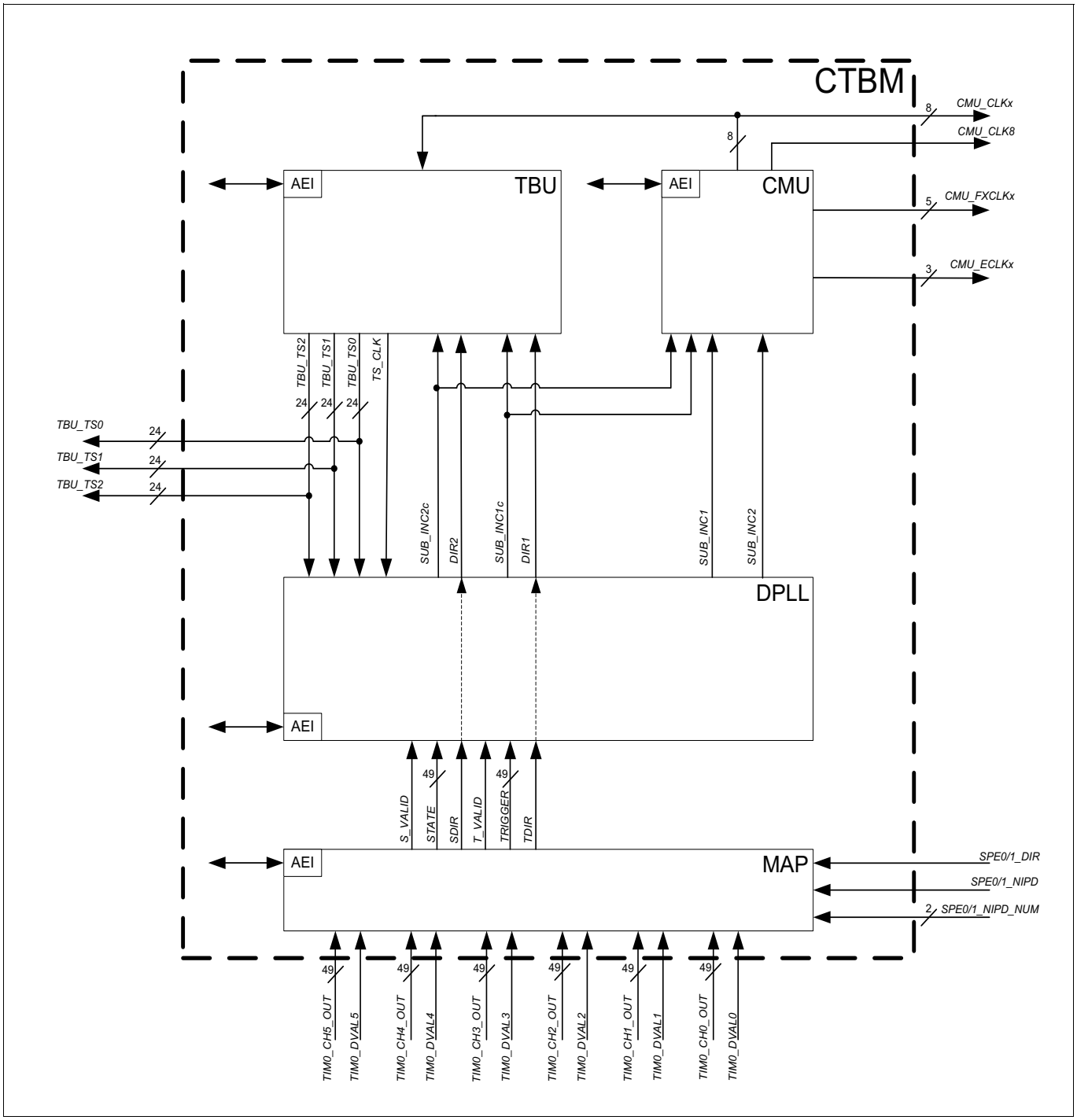


Figure 14 CTBM architecture

One important module of the CTBM is the Clock Management Unit (CMU) which generates up to 14 clocks for the sub-modules of the GTM and up to three GTM external clocks *CMU_ECLK[z]* (z: 0…2). For a detailed description of the CMU functionality and clocks please refer to Clock Management Unit chapter.

The five (5) *CMU_FXCLK[y]* (y: 0…4) clocks are used by the TOM sub-module for PWM generation.

A maximum of nine (9) *CMU_CLK[x]* (x: 0…8) clocks are used by other sub-modules of the GTM for signal generation.

Inside the Time Base Unit (TBU) one of *CMU_CLK[x]* (x: 0…7) clocks is used per channel to generate a common time base for the GTM. Besides the *CMU_CLK[x]* signals, the TBU can use the compensated *SUB_INC[i]c* (i: 1,2) signals coming from the DPLL sub-module for time base generation. This time base then typically represents an angle clock for an engine management system. For the meaning of compensated (*SUB_INC[i]c*) and uncompensated (*SUB_INC[i]*) DPLL signals please refer to the DPLL chapter. The *SUB_INC[i]c* signals in combination with the two direction signal lines *DIR[i]* the TBU time base can be controlled to run forwards or backwards. The TBU functionality is described in chapter “Time Base Unit (TBU)”.

The TBU sub-module generates the three time base signals *TBU_TS0*, *TBU_TS1* and *TBU_TS2* which are widely used inside the GTM as common time bases for signal characterization and generation.

Besides the time base 1 and 2 which may represent a relative angle clock for an engine management system it is helpful to have an absolute angle clock for CPU/MCS internal angle algorithm calculations. This absolute angle clock is represented by the TBU base 3. The TBU channel 0 up to 2 are widely used inside the GTM as common time (channel 0, 1 and/or 2) or angle (channel 1 and/or 2) bases for signal characterization and generation. The TBU channel 3 is only configurable and readable by MCS0 or CPU.

As stated before, the DPLL sub-module provides the four clock signals *SUB_INC[i]* and *SUB_INC[i]c* which can be seen as a clock multiplier generated out of the two input signal vectors *TRIGGER* and *STATE* coming from the MAP sub-module. For a detailed description of the DPLL functionality please refer to chapter “Digital PLL Module (DPLL)”.

The MAP sub-module is used to select the *TRIGGER* and *STATE* signals for the DPLL out of six input signals coming from TIM0 sub-module. Besides this, the MAP sub-module is able to generate a *TDIR* (TRIGGER Direction) and *SDIR* (STATE Direction) signal for the DPLL and TBU coming from the SPE0 and SPE1 signal lines. The direction signals are generated out of a defined input pattern. For a detailed description of the MAP sub-module please refer to chapter “TIM0 Input Mapping Module (MAP)”.

28.4.4.2 Cyclic Event Compare

With the time base module (TBU) the GTM provides three counters, where the counter of TBU_CH0 represents a time and the counter TBU_CH1 and TBU_CH2 may represent a time (if clock source is CMU_CLK generated inside CMU) or an angle (if clock source is a DPLL sub_inc signal provided via CMU).

From application point of view it is necessary to divide the cyclic event counter representing time or angle into two parts, the past and the future.The border of past/future is a moving border depending on current time or angle value.The cyclic event counting and the moving border of past/future is depicted in the figure below.

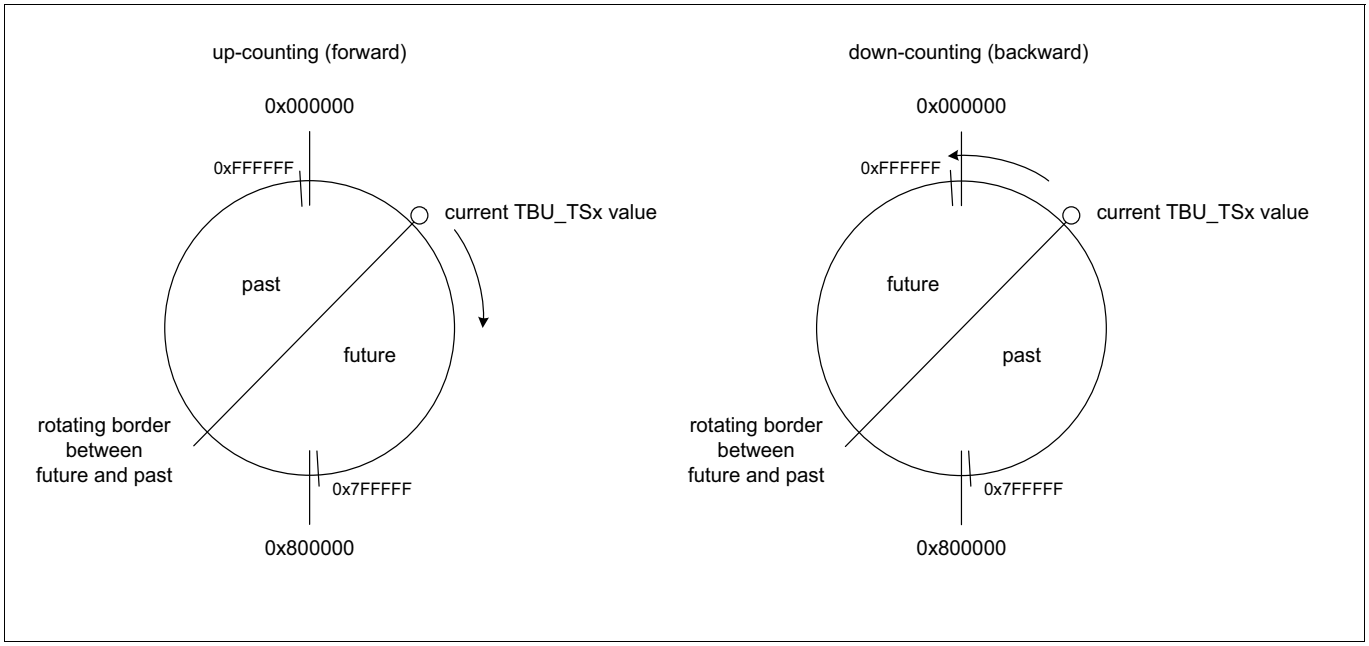


Figure 15 Cyclic event counter representing time or angle

Inside different submodules of GTM a grater-equal compare (in case of up-counting) or a less-equal compare (in case of down-counting) against a TBU base value (representing time or angle) always means that it is checked if the reference value is in relation to the current TBU value in the future or in the past.

28.4.5 GTM Interrupt Concept

The sub-modules of the GTM can generate thousands of interrupts on behalf of internal events. This high amount of interrupts is combined inside the Interrupt Concentrator Module (ICM) into interrupt groups. In these interrupt groups the GTM sub-module interrupt signals are bundled to a smaller set of interrupts. From these interrupt sets, a smaller amount of interrupt signals is created and signaled outside of the GTM as a signal *GTM_<MOD>_IRQ,* where <MOD> identifies the name of the corresponding GTM sub-module.

Moreover, each output signal *GTM_<MOD>_IRQ* has a corresponding input signal *GTM_<MOD>_IRQ_CLR* that can be used for clearing the interrupts. These input signals can be used by the surrounding micro controller system as:

- acknowledge signal from a DMA controller
- validation signal from ADC
- clear signal from a GTM-external interrupt controller to do an atomic clear while entering an ISR routine

The controlling of the individual interrupts is done inside the sub-modules. If a sub-module consists of several sub-module channels that are most likely to work independent from each other (like TIM, PSM, MCS, TOM, and ATOM), each sub-module channel has its own interrupt control and status register set, named as interrupt set in the following. Other sub-modules (SPE, ARU, DPLL, BRC, CMP and global GTM functionality) have a common interrupt set for the whole sub-module.

The interrupt set consists of four registers: The **IRQ_EN** register, the **IRQ_NOTIFY** register, the **IRQ_FORCINT** register, and the **IRQ_MODE** register. While the registers **IRQ_EN**, **IRQ_NOTIFY,** and **IRQ_FORCINT** signalize the status and allow controlling of each individual interrupt source within an interrupt set, the register **IRQ_MODE** configures the interrupt mode that is applied to all interrupts that belong to the same interrupt set.

In order to support a wide variety of micro controller architectures and interrupt systems with different interrupt signal output characteristics and internal interrupt handling the following four modes can be configured:

- Level mode,

- Pulse mode,
- Pulse-Notify mode,
- Single-Pulse mode.

These interrupt modes are described in more details in the following subsections.

The register **IRQ_EN** allows the enabling and disabling of an individual interrupt within an interrupt set. Independent of the configured mode, only enabled interrupts can signalize an interrupts on its signal *GTM_<MOD>_IRQ*.

The register **IRQ_NOTIFY** collects the occurrence of interrupt events. The behavior for setting a bit in this register depends on the configured mode and thus it is described later on in the mode descriptions.

Independent of the configured mode any write access with value '1' to a bit in the register **IRQ_NOTIFY** always clears the corresponding **IRQ_NOTIFY** bit.

Moreover, the enabling of a disabled interrupt source with a write access to the register **IRQ_EN** also clears the corresponding bit in the **IRQ_NOTIFY** register but only if the error interrupt source **EIRQ_EN** is disabled. However, if the enabling of a disabled interrupt is simultaneous to an incoming interrupt event, the interrupt event is dominant and the register **IRQ_NOTIFY** is not cleared.

Additionally, each write access to the register **IRQ_MODE**, clears all bits in the **IRQ_NOTIFY** register. It should be notified that the clearing of **IRQ_NOTIFY** is applied independently of the written data (e.g. no mode change).

Thus, a secure way for reconfiguring the interrupt mode of an interrupt set, is to disable all interrupts of the interrupt set with the register **IRQ_EN**, define the new interrupt mode by writing register **IRQ_MODE**, followed by enabling the desired interrupts with the register **IRQ_EN**.

Thus, a secure way for reconfiguring the interrupt mode of an error interrupt set, is to disable all error interrupts of the error interrupt set with the register **EIRQ_EN**, define the new interrupt mode by writing register **IRQ_MODE**, followed by enabling the desired error interrupts with the register **EIRQ_EN**.

The register **IRQ_FORCINT** is used by software for triggering individual interrupts with a write access with value '1'. Since a write access to **IRQ_FORCINT** only generates a single pulse, **IRQ_FORCINT** is not implemented as a true register and thus any read access to **IRQ_FORCINT** always results with a value of '0'.

The mechanism for triggering interrupts with **IRQ_FORCINT** is globally disabled after reset. It has to be explicitly enabled by clearing the bit **RF_PROT** in the register **GTM_CTRL** (see **Chapter 28.4.9.3**)

For the modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE and CMP each interrupt may configured to raise instead of the normal interrupt an error interrupt if enabled by the corresponding error interrupt enable bit in register **EIRQ_EN**.It is possible for one source to enable the normal interrupt and the error interrupt in parallel. Because both interrupt clear signals could reset the notify bit this is expected to cause problems in a system and therefore it is strongly recommended to not enable both interrupt types at the same point in time.

Similar to enabling an interrupt, the enabling of a disabled error interrupt source with a write access to the register **EIRQ_EN** also clears the corresponding bit in the **IRQ_NOTIFY** register only if the interrupt source **IRQ_EN** is disabled. However, if the enabling of a disabled error interrupt is simultaneous to an incoming interrupt event, the interrupt event is dominant and the register **IRQ_NOTIFY** is not cleared.

All enabled error interrupts are OR-combined inside the ICM and assigned to the dedicated GTM port *gtm_err_irq*. A corresponding input *gtm_err_irq_clr* allows the reset of this error interrupt from outside the GTM (hardware clear).

To be able to detect the module source of the error interrupt the ICM provides the register **ICM_IRQG_MEI**.

The error interrupt causing channel can be determined for the module FIFO by evaluating the ICM register **ICM_IRQG_CEI0**.

The error interrupt causing channel can be determined for the modules TIM by evaluating the ICM register **ICM_IRQG_CEI1...2**.

The error interrupt causing channel can be determined for the modules MCS with all possible channel by evaluating the ICM register **ICM_IRQG_MCS[i]_CEI**. In case of usage only the first 8 channels of each MCS the error interrupt causing channel can be determined by evaluating the ICM register **ICM_IRQG_CEI3...4**.

28.4.5.1 Level interrupt mode

The default interrupt mode is the Level Interrupt Mode. In this mode each occurred interrupt event is collected in the register **IRQ_NOTIFY**, independent of the corresponding enable bit of register **IRQ_EN** and **EIRQ_EN**.

An interrupt event, which is defined as a pulse on the signal *Int_out* of **Figure 16**, may be triggered by the interrupt source of the sub-module or by software performing a write access to the corresponding register **IRQ_FORCINT**, with a disabled bit **RF_PROT** in register **GTM_CTRL**.

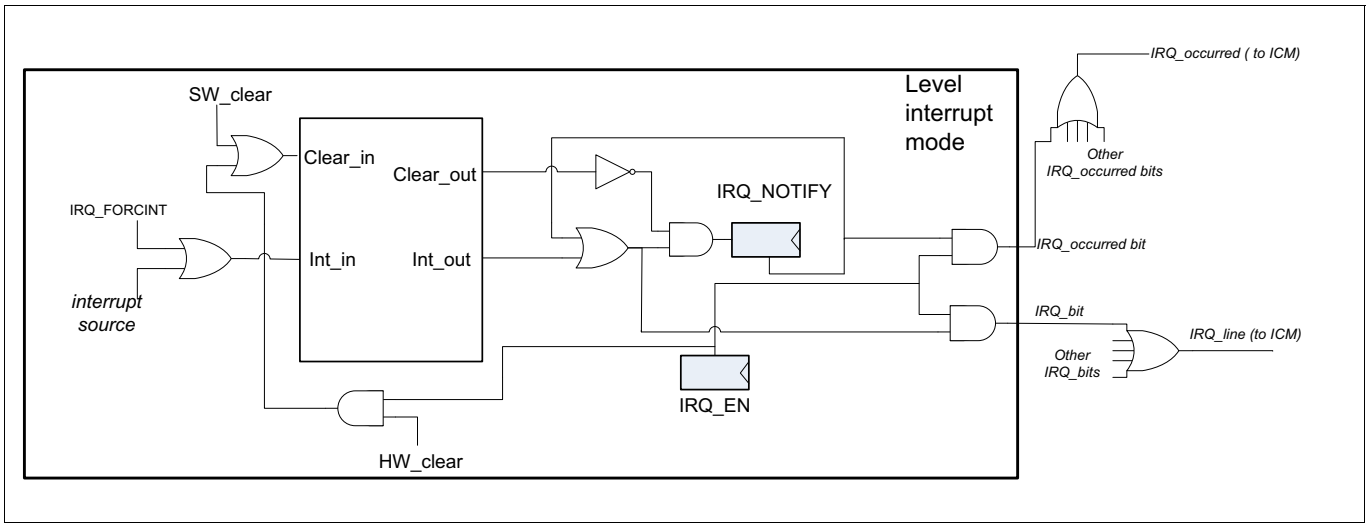


Figure 16 Level interrupt mode scheme

A collected interrupt bit in register **IRQ_NOTIFY** may be cleared by a clear event, which is defined as a pulse on signal *Clear_out* of **Figure 16**. A clear event can be performed by writing '1' to the corresponding bit in the register **IRQ_NOTIFY** leading to a pulse on signals *SW_clear*. A clear event may also result from an externally connected signal *GTM_<MOD>_IRQ_CLR*, which is routed to the signal *HW_clear* of **Figure 16**. However, the hardware clear mechanism is only possible, if the corresponding interrupt is enabled by register **IRQ_EN**.

As **Table 9** shows, interrupt events are dominant in the case of a simultaneous interrupt event and clear event.

Table 9 Priority of Interrupt Events and Clear Events

Int_in	Clear_in	Int_out	Clear_out
0	0	0	0
0	1	0	1
1	0	1	0
1	1	1	0

As shown in **Figure 16** an occurred interrupt event is signaled as a constant signal level with value 1 to the signal *IRQ_bit*, if the corresponding interrupt is enabled in register **IRQ_EN**.

With exception of the sub-modules ARU and DPLL, the signal *IRQ_bit* is OR-combined with the neighboring *IRQ_bit* signals of the same interrupt set and they are routed as a signal *IRQ_line* to the interrupt concentrator module (ICM). The interrupt signals *IRQ_bit* of the sub-modules DPLL and ARU are routed directly as a signal *IRQ_line* to the sub-module ICM. In some cases (sub-modules TOM and ATOM) the ICM may further OR-combine

several *IRQ_line* signals to an outgoing interrupt signal *GTM_<MOD>_IRQ*. In the other cases the *IRQ_line* signals are directly connected to the outgoing signals *GTM_<MOD>_IRQ,* within the sub-module ICM.

The signal *IRQ_occurred* is connected in a similar way as the signal *IRQ_line*, however this signal is used for monitoring the interrupt state of the register **IRQ_NOTIFY** in the registers of the ICM.

The additional error interrupt enable mechanism for level interrupt is shown below.

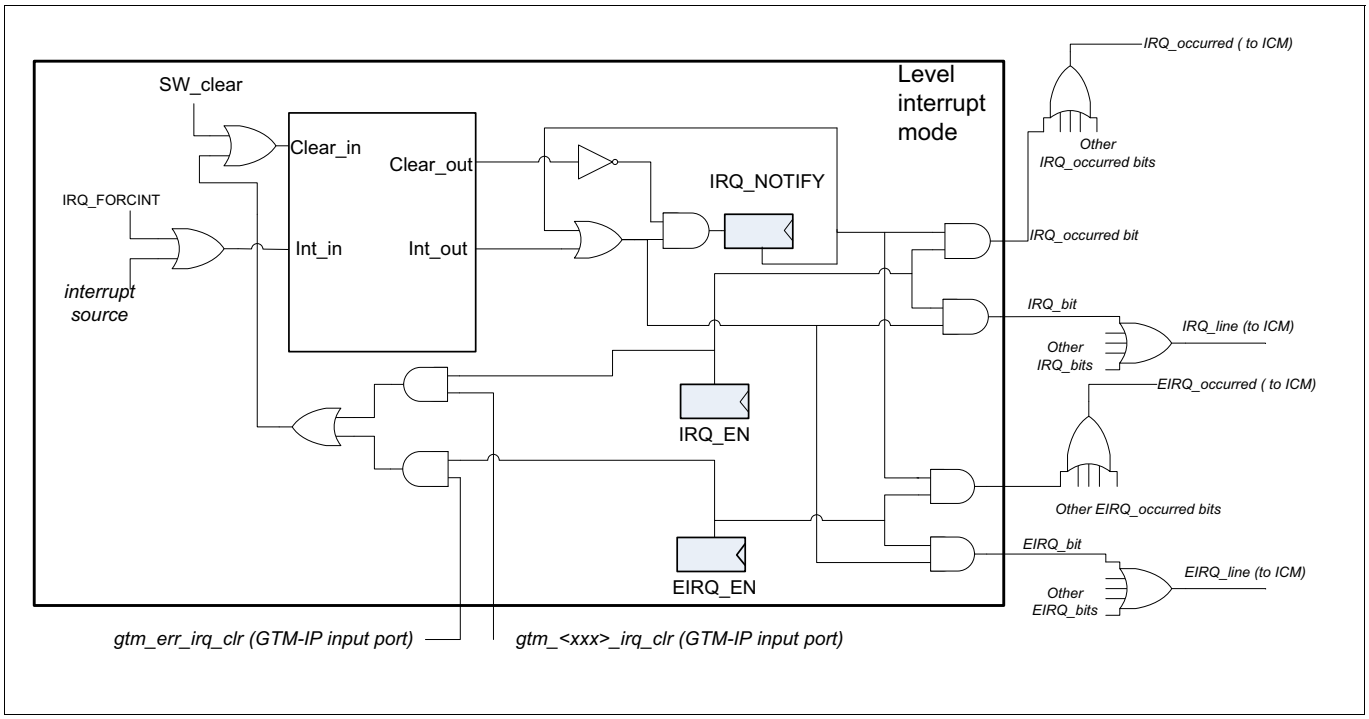


Figure 17 Level interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP

A collected interrupt bit in register **IRQ_NOTIFY** may be cleared by a clear event, which is defined as a pulse on signal *Clear_out* of **Figure 17**. A clear event can be performed by writing '1' to the corresponding bit in the register **IRQ_NOTIFY** leading to a pulse on signals *SW_clear*. A clear event may also result from the externally connected signal *gtm_<MOD>_irq_clr* or *gtm_err_irq_clr*, which is routed as an *HW_clear* to *Clear_in* of **Figure 17**. However, the hardware clear mechanism is only possible, if the corresponding interrupt or error interrupt is enabled by register **IRQ_EN** or **EIRQ_EN**.

As it can be seen from the **Figure 17** an occurred interrupt event is signaled as a constant signal level with value 1 to the signal *IRQ_bit*, if the corresponding interrupt is enabled in register **IRQ_EN**.

28.4.5.2 Pulse interrupt mode

The Pulse interrupt mode behavior can be observed from **Figure 18**.

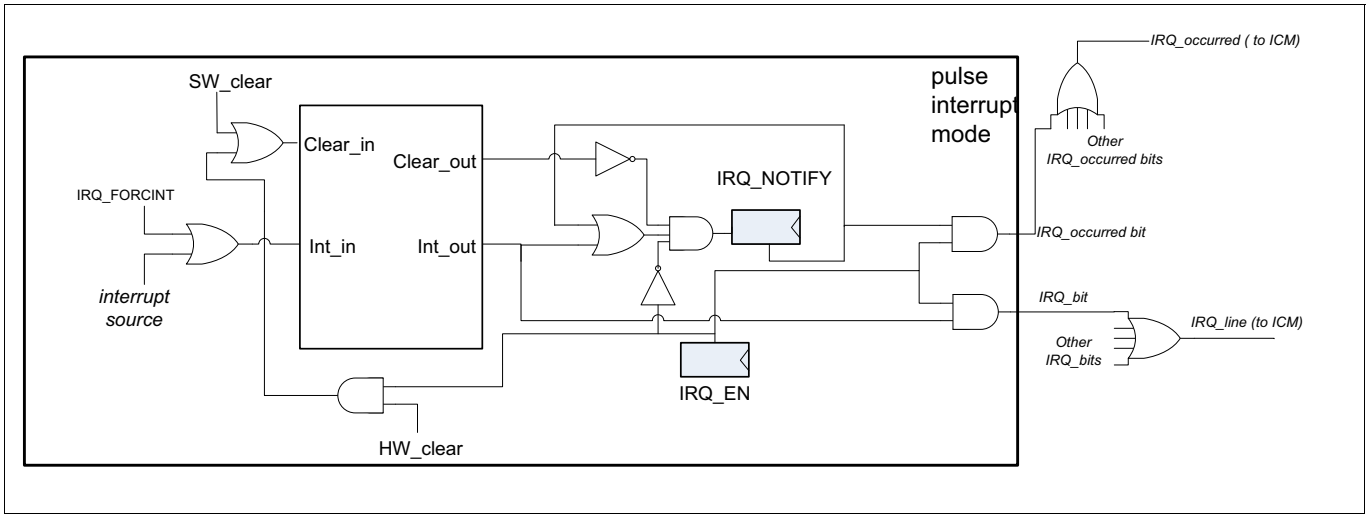


Figure 18 Pulse interrupt mode scheme

In Pulse Interrupt Mode each Interrupt Event will generate a pulse on the *IRQ_bit* signal if **IRQ_EN** is enabled.

As it can be seen from the figure, the interrupt bit in **IRQ_NOTIFY** register is always cleared if **IRQ_EN** is enabled.

However, if an interrupt is disabled in the register **IRQ_EN**, an occurred interrupt event is captured in the register **IRQ_NOTIFY**, in order to allow polling for disabled interrupts by software.

Disabled interrupts may be cleared by an interrupt clear event.

In Pulse interrupt mode, the signal IRQ_occurred is always 0.

The additional error interrupt enable mechanism for pulse interrupt is shown below.

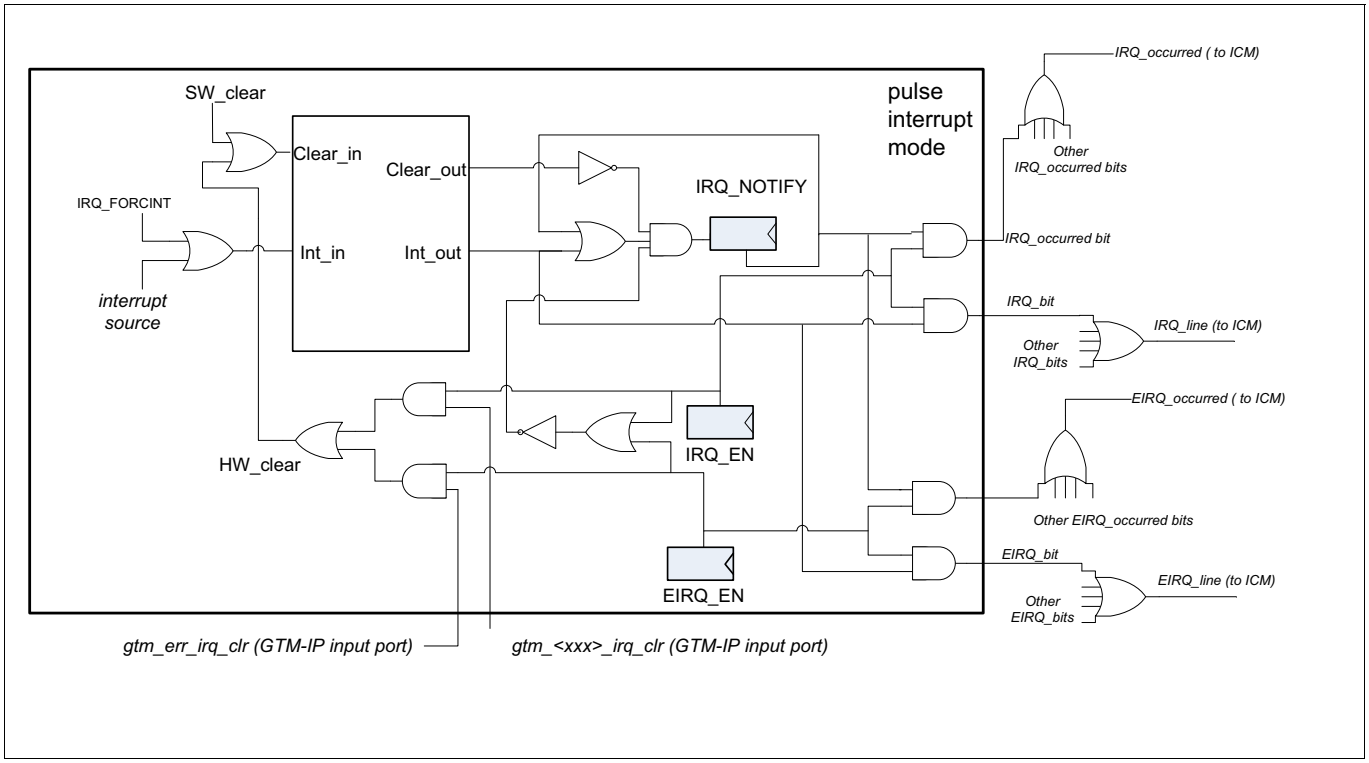


Figure 19 Pulse interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP

In Pulse Interrupt Mode each Interrupt Event will generate a pulse on the *EIRQ_bit* signal if **EIRQ_EN** is enabled.

As it can be seen from the figure, the interrupt bit in **IRQ_NOTIFY** register is always cleared if **EIRQ_EN** or **IRQ_EN** are enabled.

However, if an error interrupt is disabled in the register **EIRQ_EN**, an occurred error interrupt event is captured in the register **IRQ_NOTIFY**, in order to allow polling for disabled error interrupts by software.

Disabled error interrupts may be cleared by an error interrupt clear event.

In Pulse interrupt mode, the signal EIRQ_occurred is always 0.

28.4.5.3 Pulse-notify interrupt mode

In Pulse-notify Interrupt mode, all interrupt events are captured in the register **IRQ_NOTIFY**. If an interrupt is enabled by the register **IRQ_EN**, each interrupt event will also generate a pulse on the *IRQ_bit* signal. The signal *IRQ_occurred* will be high if interrupt is enabled in register **IRQ_EN** and the corresponding bit of register **IRQ_NOTIFY** is set. The Pulse-notify interrupt mode is shown in **Figure 20**.

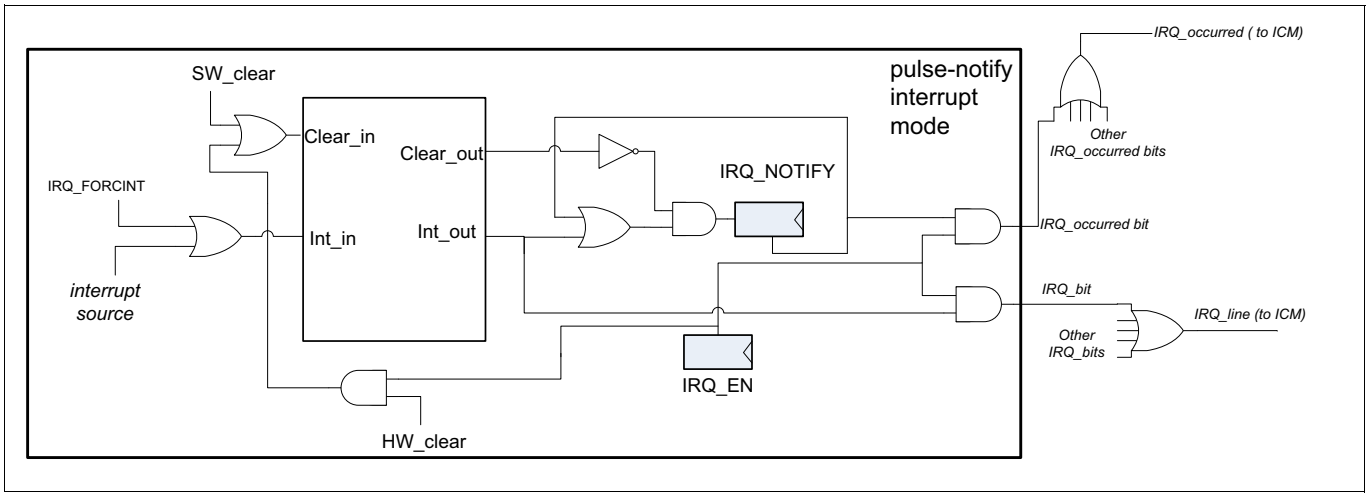


Figure 20 Pulse-notify interrupt mode scheme

The additional error interrupt enable mechanism for pulse-notify interrupt is shown below

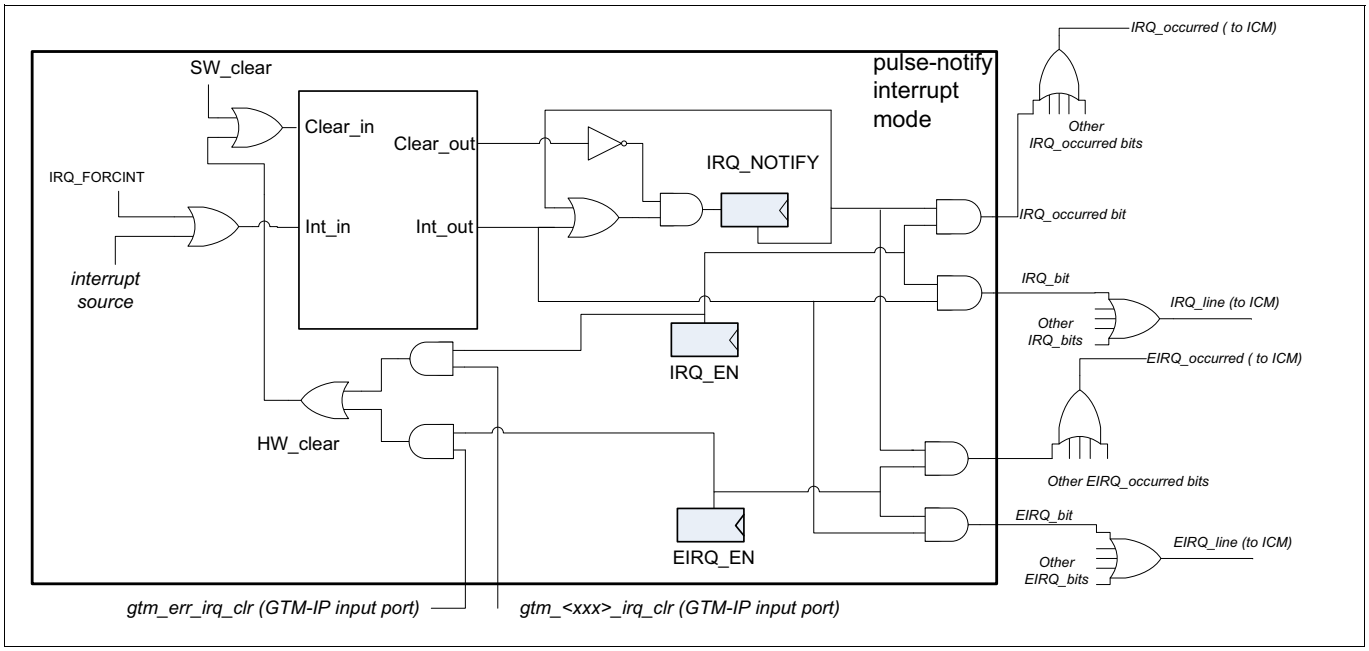


Figure 21 Pulse-notify interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP

In Pulse-notify Interrupt mode, all error interrupt events are captured in the register **IRQ_NOTIFY**. If an error interrupt is enabled by the register **EIRQ_EN**, each error interrupt event will also generate a pulse on the *EIRQ_bit* signal. The signal *EIRQ_occurred* will be high if error interrupt is enabled in register **EIRQ_EN** and the corresponding bit of register **IRQ_NOTIFY** is set. The Pulse-notify interrupt mode for error interrupts is shown in **Figure 21**.

28.4.5.4 Single-pulse interrupt mode

In Single-pulse Interrupt Mode, an interrupt event is always captured in the register **IRQ_NOTIFY**, independent of the state of **IRQ_EN**. However, only the first interrupt event of an enabled interrupt within a common interrupt set is forwarded to signal *IRQ_line.* Additional interrupt events of the same interrupt set cannot generate pulses on the signal *IRQ_line*, until the corresponding bits in register **IRQ_NOTIFY** of enabled interrupts are cleared by a clear event. The *IRQ_occurred* signal line will be high, if the **IRQ_EN** and the **IRQ_NOTIFY** register bits are set. The Single-pulse interrupt mode is shown in **Figure 22**.

The only exceptions are the modules ARU and DPLL. In these modules the *IRQ_occurred* bit of each interrupt is directly connected (without OR-conjunction of neighboring *IRQ_occurred* bits) to the inverter for suppressing further interrupt pulses.

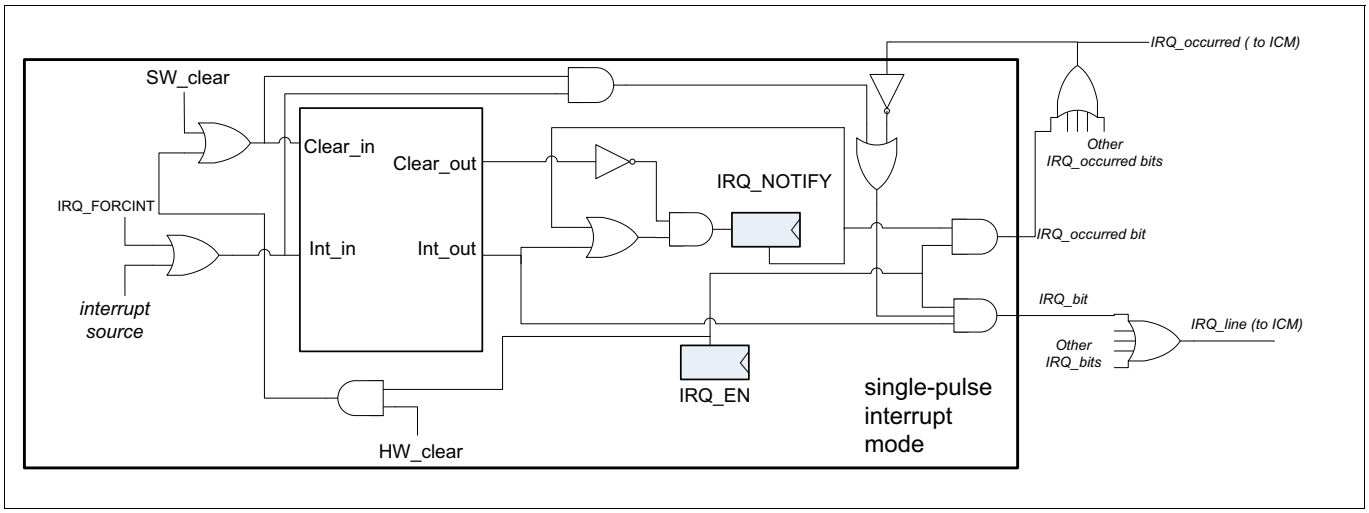


Figure 22 Single-pulse interrupt mode scheme

To avoid unexpected IRQ behavior in the single pulse mode, all desired interrupt sources should be enabled by a single write access to **IRQ_EN** and the notification bits should be cleared by a single write access to the register **IRQ_NOTIFY**.

The additional error interrupt enable mechanism for single-pulse interrupt is shown below.

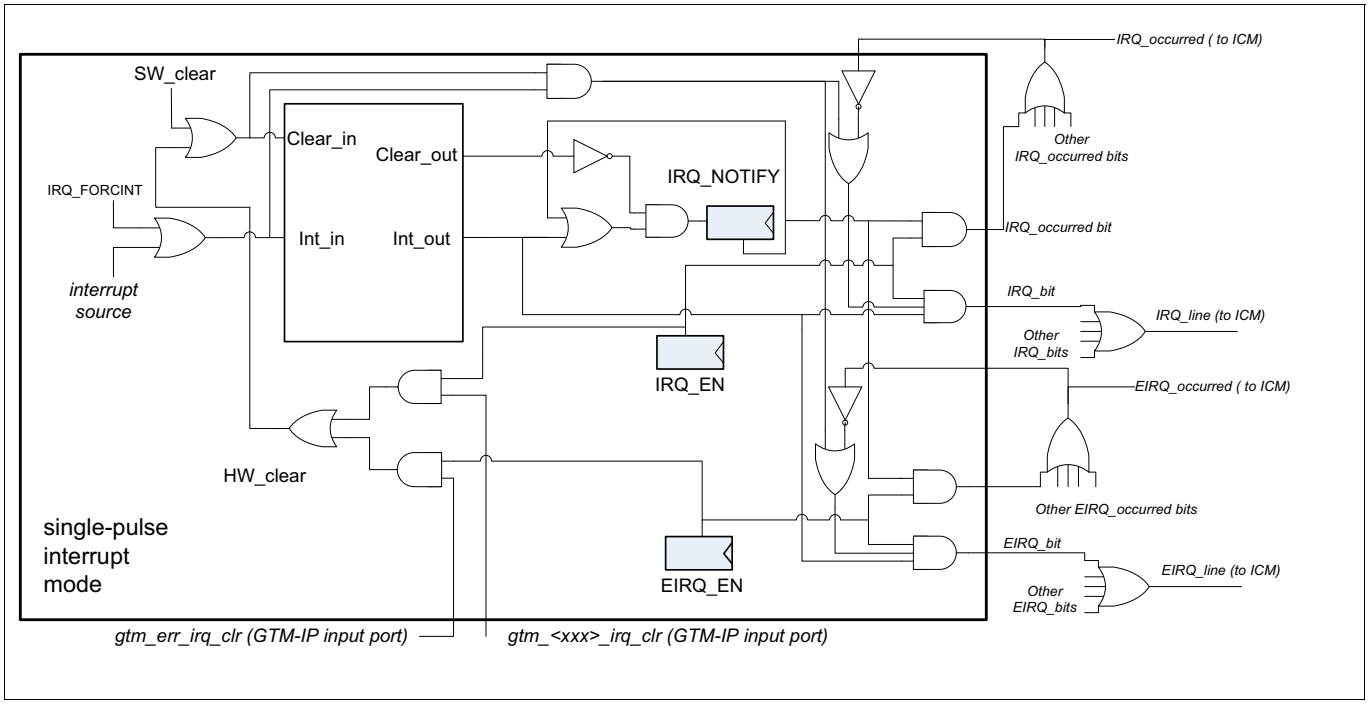


Figure 23 Single-pulse interrupt scheme for modules AEI-bridge, BRC, FIFO, TIM, MCS, DPLL, SPE, CMP

In Single-pulse Interrupt Mode, an error interrupt event is always captured in the register **IRQ_NOTIFY**, independent of the state of **EIRQ_EN**. However, only the first error interrupt event of an enabled error interrupt within a common error interrupt set is forwarded to signal *EIRQ_line.* Additional error interrupt events of the same error interrupt set cannot generate pulses on the signal *EIRQ_line*, until the corresponding bits in register **IRQ_NOTIFY** of enabled error interrupts are cleared by a clear event. The *EIRQ_occurred* signal line will be high, if the **EIRQ_EN** and the **IRQ_NOTIFY** register bits are set. The Single-pulse interrupt mode for error interrupts is shown in **Figure 23**.

To avoid unexpected EIRQ behavior in the single pulse mode, all desired error interrupt sources should be enabled by a single write access to **EIRQ_EN** and the notification bits should be cleared by a single write access to the register **IRQ_NOTIFY**.

The only exceptions are the modules ARU and DPLL. In these modules the *EIRQ_occurred* bit of each error interrupt is directly connected (without OR-conjunction of neighboring *EIRQ_occurred* bits) to the inverter for suppressing further error interrupt pulses.

28.4.5.5 GTM Interrupt concentration method

Because of the grouping of interrupts inside the ICM, it can be necessary for the software to access the ICM sub-module first to determine the interrupt set that is responsible for an interrupt. A second access to the responsible register **IRQ_NOTIFY** is then necessary to identify the interrupt source, serve it and to reset the interrupt flag in register **IRQ_NOTIFY** afterwards. The interrupt flags are never reset by an access to the ICM. For a detailed description of the ICM sub-module please refer to chapter “Interrupt Concentrator Module (ICM)”.

28.4.6 GTM Software Debugger Support

For software debugger support the GTM comes with several features. E.g. status register bits must not be altered by a read access from a software debugger. To avoid this behavior to reset a status register bit by software, the CPU has to write a '1' explicitly to the register bit to reset its content.

Table 10 describes the behavior of some GTM registers with special functionality on behalf of read accesses from the AEI bus interface.

Table 10 Register behavior in case of Software Debugger accesses

Module	Register	Description
AFD	AFD[i]_CH[x]_BUFFACC	The FIFO read access pointers are not altered on behalf of a Debugger read access to this register.
TIM	TIM[i]_CH[x]_GPR0/1	The overflow bit is not altered in case of a Debugger read access to this registers.
ATOM	ATOM[i]_CH[x]_SR0/1	In SOMC mode a read access to this register by the Debugger does not release the channel for a new compare/match event.

Further on, some important states inside the GTM sub-module have to be signaled to the outside world, when reached and should for example trigger the software debugger to stop program execution. For this internal state signaling please refer to the GTM module integration guide.

The GTM provides an external signal *gtm_halt*, which disables clock signal *SYS_CLK* for debugging purposes. If *SYS_CLK* is disabled, a connected debugger can read any GTM related register and the GTM internal RAMs using AEI. Moreover, the debugger can also perform write accesses to the internal RAMs and to all GTM related registers in order to enable advanced debugging features (e.g. modifications of register contents in single step mode).

28.4.7 GTM Programming conventions

To serve different application domains the GTM is a highly configurable module with many configuration modes. In principle the sub-modules of the GTM are intended to be configured at system startup to fulfill certain functionality for the application domain the micro controller runs in.

For example, a TIM input channel can be used to monitor an application specific external signal, and this signal has to be filtered. Therefore, the configuration of the TIM channel filter mode will be specific to the external signal characteristic. While it can be necessary to adapt the filter thresholds during runtime an adaptation of the filter mode during runtime is not reasonable. Thus, the change of the filter mode during runtime can lead to an unexpected behavior.

In general, the programmer has to be careful when reprogramming configuration registers of the GTM sub-modules during runtime. It is recommended to disable the channels before reconfiguration takes place to avoid unexpected behavior of the GTM.

28.4.8 GTM TOP-Level Configuration Register Overview

Table 11 GTM TOP-Level Configuration Register Overview

Register name	Description	see Page
GTM_REV	GTM Version control register	30
GTM_RST	GTM Global reset register	31
GTM_CTRL	GTM Global control register	31
GTM_AEI_ADDR_XPT	GTM AEI Timeout exception address register	32
GTM_AEI_STA_XPT	GTM AEI Non zero status register	33
GTM_IRQ_NOTIFY	GTM Interrupt notification register	34
GTM_IRQ_EN	GTM Interrupt enable register	36
GTM_EIRQ_EN	GTM Error interrupt enable register	44

Table 11 GTM TOP-Level Configuration Register Overview (cont'd)

Register name	Description	see Page
GTM_IRQ_FORCINT	GTM Software interrupt generation register	37
GTM_IRQ_MODE	GTM top level interrupts mode selection	39
GTM_BRIDGE_MODE	GTM AEI bridge mode register	39
GTM_BRIDGE_PTR1	GTM AEI bridge pointer 1 register	41
GTM_BRIDGE_PTR2	GTM AEI bridge pointer 2 register	42
GTM_MCS_AEM_DIS	GTM MCS master port disable register	43
GTM_CLS_CLK_CFG	GTM Cluster Clock Configuration	45
GTM_CFG	GTM Configuration register	46

28.4.9 GTM TOP-Level Configuration Registers Description

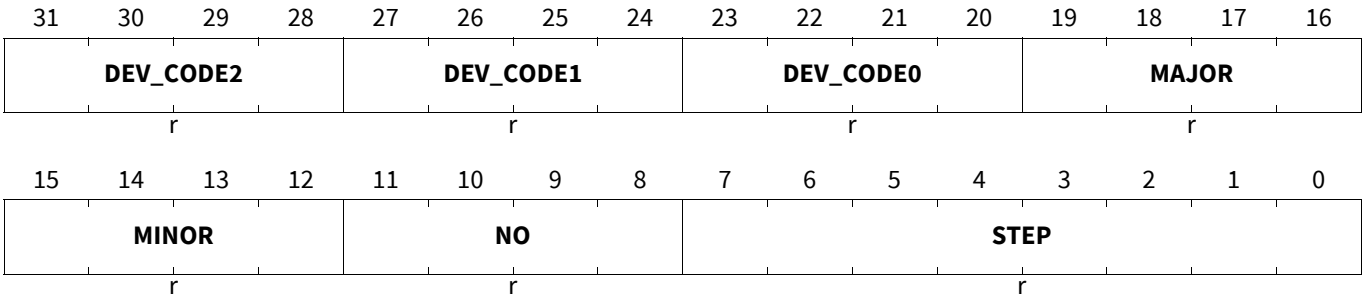
28.4.9.1 Register GTM_REV

Please keep in mind, that the actual Revision number is the reset value. This reset value is dependent on the delivery done by Bosch AE for the actual device. In case of Infineon's decision to change via metal fix for a different version, the reset value contains the initial version.

GTM Version Control Register

Note: The numbers are encoded in BCD. Values "A" - "F" are characters.

REV
GTM Version Control Register **(000000_H)** **Application Reset Value: $3153\ 15B6_H$**



Field	Bits	Type	Description
STEP	7:0	r	**Release step** GTM Release step
NO	11:8	r	**Delivery number** Define delivery number of GTM specification.
MINOR	15:12	r	**Minor version number** Define minor version number of GTM specification.
MAJOR	19:16	r	**Major version number** Define major version number of GTM specification.
DEV_CODE0	23:20	r	**Device encoding digit 0** Device encoding digit 0.
DEV_CODE1	27:24	r	**Device encoding digit 1** Device encoding digit 1.
DEV_CODE2	31:28	r	**Device encoding digit 2** Device encoding digit 2.

28.4.9.2 Register GTM_RST

GTM Global Reset Register

RST

GTM Global Reset Register **(000004$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				BRIDGE_MODE_WRDIS	0										
r				rw	r										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															0
r															r

Field	Bits	Type	Description
BRIDGE_MODE_WRDIS	27	rw	**GTM_BRIDGE_MODE write disable** This bit is write protected by bit RF_PROT 0_B Writing of GTM_BRIDGE_MODE register is enabled 1_B Writing of GTM_BRIDGE_MODE register is disabled
0	0, 26:1, 31:28	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.3 Register GTM_CTRL

GTM Global Control Register

CTRL

GTM Global Control Register **(000008$_H$)** **Application Reset Value: 0000 0001$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEIM_CLUSTER				0			TO_VAL					0		TO_MODE	RF_PROT
r				r			rw					r		rw	rw

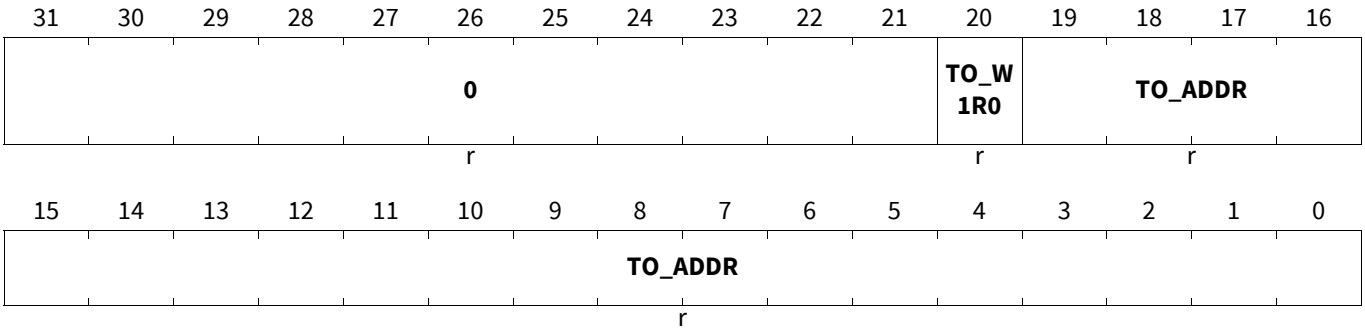
Field	Bits	Type	Description
RF_PROT	0	rw	**RST and FORCINT protection** 0_B SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is enabled 1_B SW RST (global), SW interrupt FORCINT, and SW RAM reset functionality is disabled
TO_MODE	1	rw	**AEI timeout mode** 0_B Observe: If timeout_counter=0 the address and rw signal in addition with timeout flag will be stored to the GTM_AEI_ADDR_XPT register. Following timeout_counter=0 accesses will not overwrite the first entry in the aei_addr_timeout register. Clearing the timeout flag/aei_status error_code will reenable the storing of a next faulty access. 1_B Abort: In addition to observe mode, the pending access will be aborted by signaling an illegal module access on aei_status and sending ready. In case of a read, deliver as data 0 by serving of next AEI accesses.
TO_VAL	8:4	rw	**AEI timeout value** These bits define the number of cycles after which a timeout event occurs. When TO_VAL equals zero (0) the AEI timeout functionality is disabled.
AEIM_CLUSTER	15:12	r	**AEIM cluster number** These bits show the number of the AEI master port cluster which throws the interrupts *AEIM_USP_ADDR, AEIM_IM_ADDR* and *AEIM_USP_BE* depending on the AEI master port access status.Note: If one of the corresponding irq notify bits (6:4) is set, this bit field will be frozen until the interrupt notify bits (6:4) are cleared.
0	3:2, 11:9, 31:16	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.4 Register GTM_AEI_ADDR_XPT

GTM AEI Timeout Exception Address Register

AEI_ADDR_XPT

GTM AEI Timeout Exception Address Register (00000C$_H$) **Application Reset Value: 0000 0000$_H$**

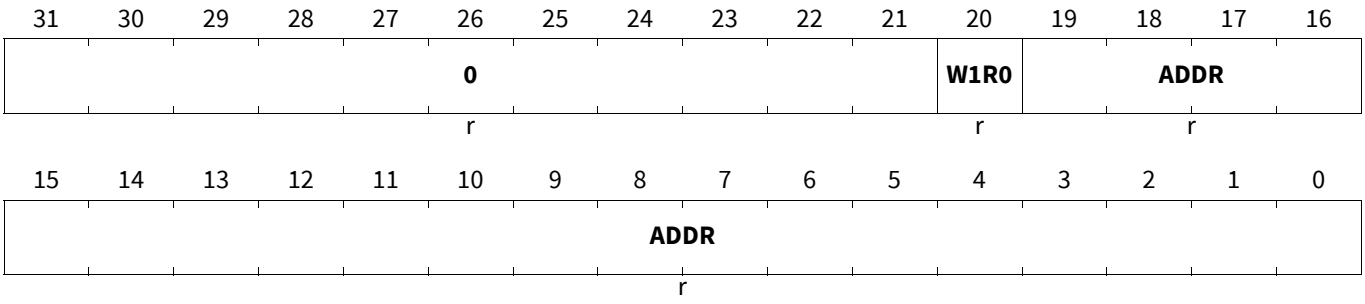


Field	Bits	Type	Description
TO_ADDR	19:0	r	**AEI timeout address** This bit field defines the AEI address for which the AEI timeout event occurred.
TO_W1R0	20	r	**AEI timeout Read/Write flag** This bit defines the AEI Read/Write flag for which the AEI timeout event occurred.
0	31:21	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.5 Register GTM_AEI_STA_XPT

GTM AEI Non Zero Status Register

AEI_STA_XPT
GTM AEI Non Zero Status Register **(00002C$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ADDR	19:0	r	**AEI exception address** This bit field captures the address of the first AEI access resulting with a non-zero AEI status signal. The bit field can be cleared by clearing the interrupt flags AEI_USP_ADDR,AEI_USP_BE, and AEI_IM_ADDR in the register GTM_IRQ_NOTIFY.
W1R0	20	r	**AEI exception Read/Write flag** This bit defines the AEI Read/Write flag for which the AEI non-zero event occurred. This bit field captures the address of the first AEI access resulting with a non-zero AEI status signal. The bit field can be cleared by clearing the interrupt flags AEI_USP_ADDR, AEI_USP_BE, and AEI_IM_ADDR in the register GTM_IRQ_NOTIFY.
0	31:21	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.6 Register GTM_IRQ_NOTIFY

GTM Interrupt Notification Register

IRQ_NOTIFY

GTM Interrupt Notification Register (000010$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		CLK_EN_EXP_STATE1	CLK_EN_EXP_STATE0	0		CLK_EN_ERR_STATE1	CLK_EN_ERR_STATE0	0							
r		r	r	r		r	r	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							CLK_PER_ERR	CLK_EN_ERR	AEIM_USP_BE	AEIM_IM_ADDR	AEIM_USP_ADDR	AEI_USP_BE	AEI_IM_ADDR	AEI_USP_ADDR	AEI_TO_XPT
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
AEI_TO_XPT	0	rw	**AEI timeout exception occurred** This bit will be cleared on a CPU write access of value '1'. (As the bit is rw, otherwise no clear.) A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEI_TO_XPT interrupt was raised by the AEI Timeout detection unit
AEI_USP_ADDR	1	rw	**AEI unsupported address interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEI_USP_ADDR interrupt was raised by the AEI interface
AEI_IM_ADDR	2	rw	**AEI illegal Module address interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEI_IM_ADDR interrupt was raised by the AEI interface
AEI_USP_BE	3	rw	**AEI unsupported byte enable interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEI_USP_BE interrupt was raised by the AEI interface
AEIM_USP_ADDR	4	rw	**AEI master port unsupported address interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEIM_USP_ADDR interrupt was raised by the AEI master port interface

Field	Bits	Type	Description
AEIM_IM_ADDR	5	rw	**AEI master port illegal Module address interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEIM_IM_ADDR interrupt was raised by the AEI master port interface
AEIM_USP_BE	6	rw	**AEI master port unsupported byte enable interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B AEIM_USP_BE interrupt was raised by the AEI master port interface
CLK_EN_ERR	7	rw	**Clock enable error interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B No interrupt occurred 1_B CLK_EN_ERR interrupt was raised by clock enable watchdog
CLK_PER_ERR	8	rw	**Clock period error interrupt** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B No interrupt occurred 1_B CLK_PER_ERR interrupt was raised by clock enable watchdog
CLK_EN_ERR_STATE0	24	r	**Erroneous clock enable state** This bit field defines the GTM external clk enable state for internal clock aei_sys_clkat occurrence of the CLK_EN_ERR event. **Function only available with INT_CLK_EN_GEN = 0b0:** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Internal clock aei_sys_clk disabled 1_B Internal clock aei_sys_clk enabled
CLK_EN_ERR_STATE1	25	r	**Erroneous clock enable state** This bit field defines the GTM external clk enable state for internal clock aei_sys_clk / 2 at occurrence of the CLK_EN_ERR event. **Function only available with INT_CLK_EN_GEN = 0b0:** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Internal clock aei_sys_clk / 2 disabled 1_B Internal clock aei_sys_clk / 2 enabled
CLK_EN_EXP_STATE0	28	r	**Expected clock enable state** This bit field defines the GTM expected clk enable state for internal clock aei_sys_clk at occurrence of the CLK_EN_ERR event. **Function only available with INT_CLK_EN_GEN = 0b0:** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Internal clock aei_sys_clk disabled 1_B Internal clock aei_sys_clk enabled

Field	Bits	Type	Description
CLK_EN_EXP_STATE1	29	r	**Expected clock enable state** This bit field defines the GTM expected clk enable state for internal clock aei_sys_clk / 2 at occurrence of the CLK_EN_ERR event. **Function only available with INT_CLK_EN_GEN = 0b0:** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Internal clock aei_sys_clk / 20 disabled 1_B Internal clock aei_sys_clk / 20 enabled
0	23:9, 27:26, 31:30	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.7 Register GTM_IRQ_EN

GTM Interrupt Enable Register

IRQ_EN
GTM Interrupt Enable Register **(000014_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							CLK_PER_ERR_IRQ_EN	CLK_EN_ERR_IRQ_EN	AEIM_USP_BE_IRQ_EN	AEIM_IM_ADDR_IRQ_EN	AEIM_USP_ADDR_IRQ_E	AEI_USP_BE_IRQ_EN	AEI_IM_ADDR_IRQ_EN	AEI_USP_ADDR_IRQ_EN	AEI_TO_XPT_IRQ_EN
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
AEI_TO_XPT_IRQ_EN	0	rw	**AEI_TO_XPT_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEI_USP_ADDR_IRQ_EN	1	rw	**AEI_USP_ADDR_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEI_IM_ADDR_IRQ_EN	2	rw	**AEI_IM_ADDR_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEI_USP_BE_IRQ_EN	3	rw	**AEI_USP_BE_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEIM_USP_ADDR_IRQ_EN	4	rw	**AEI_MUSP_ADDR_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM

Field	Bits	Type	Description
AEIM_IM_ADDR_IRQ_EN	5	rw	**AEIM_IM_ADDR_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
AEIM_USP_BE_IRQ_EN	6	rw	**AEIM_USP_BE_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
CLK_EN_ERR_IRQ_EN	7	rw	**CLK_EN_ERR_IRQ interrupt enable** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
CLK_PER_ERR_IRQ_EN	8	rw	**CLK_PER_ERR_IRQ interrupt enable** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.8 Register GTM_IRQ_FORCINT

GTM Software Interrupt Generation Register

IRQ_FORCINT
GTM Software Interrupt Generation Register (000018$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							TRG_CLK_PER_ERR	TRG_CLK_EN_ERR	TRG_AEIM_USP_BE	TRG_AEIM_IM_ADDR	TRG_AEIM_USP_ADDR	TRG_AEI_USP_BE	TRG_AEI_IM_ADDR	TRG_AEI_USP_ADDR	TRG_AEI_TO_XPT
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TRG_AEI_TO_XPT	0	rw	**Trigger AEI_TO_XPT_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEI_TO_XPT_IRQ interrupt for one clock cycle
TRG_AEI_USP_ADDR	1	rw	**Trigger AEI_USP_ADDR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEI_USP_ADDR_IRQ interrupt for one clock cycle

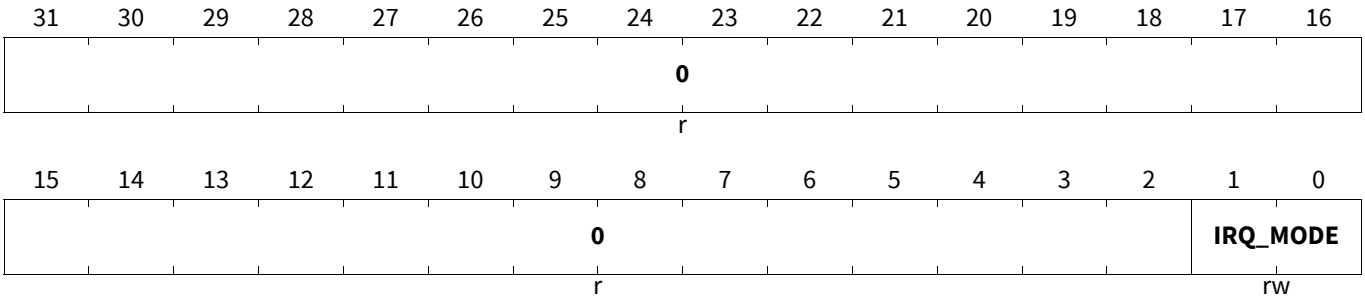
Field	Bits	Type	Description
TRG_AEI_IM_ADDR	2	rw	**Trigger AEI_IM_ADDR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEI_IM_ADDR_IRQ interrupt for one clock cycle
TRG_AEI_USP_BE	3	rw	**Trigger AEI_USP_BE_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEI_USP_BE_IRQ interrupt for one clock cycle
TRG_AEIM_USP_ADDR	4	rw	**Trigger AEIM_USP_ADDR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEIM_USP_ADDR_IRQ interrupt for one clock cycle
TRG_AEIM_IM_ADDR	5	rw	**Trigger AEIM_IM_ADDR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEIM_IM_ADDR_IRQ interrupt for one clock cycle
TRG_AEIM_USP_BE	6	rw	**Trigger AEIM_USP_BE_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** 0_B No interrupt triggering 1_B Assert AEIM_USP_BE_IRQ interrupt for one clock cycle
TRG_CLK_EN_ERR	7	rw	**Trigger CLK_EN_ERR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B No interrupt triggering 1_B Assert CLK_EN_ERR_IRQ interrupt for one clock cycle
TRG_CLK_PER_ERR	8	rw	**Trigger CLK_PER_ERR_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of **CTRL** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B No interrupt triggering 1_B Assert CLK_PER_ERR_IRQ interrupt for one clock cycle
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.9 Register GTM_IRQ_MODE

GTM Top Level Interrupts Mode Selection Register

IRQ_MODE

GTM Top Level Interrupts Mode Selection Register(00001C$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**Interrupt strategy mode selection for the AEI timeout and address monitoring interrupts** The interrupt modes are described in **Section 28.4.5**.Note: This mode selection is only valid for the six interrupts described in section **Register GTM_IRQ_NOTIFY** 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

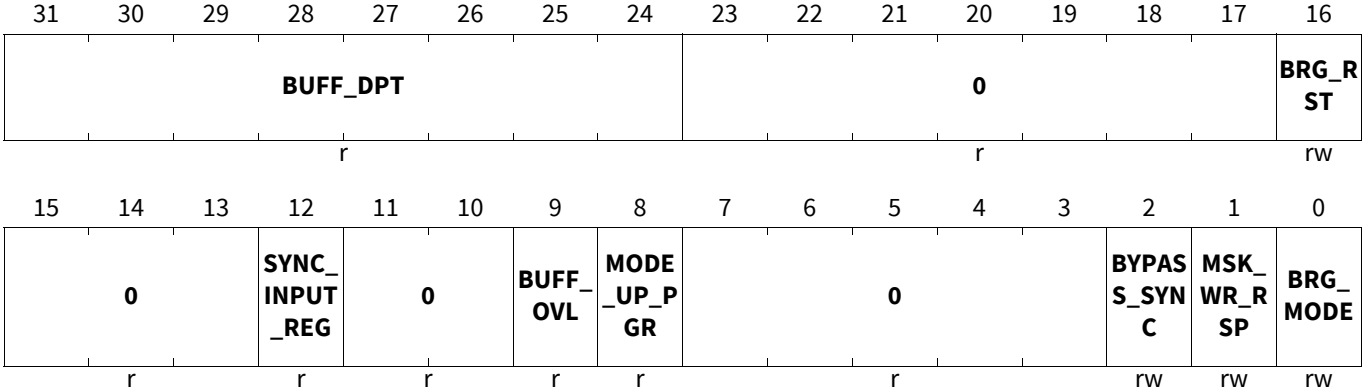
28.4.9.10 Register GTM_BRIDGE_MODE

GTM AEI Bridge Mode Register

Note: *All writable bits are write protected by bit BRIDGE_MODE_WRDIS*

BRIDGE_MODE

GTM AEI Bridge Mode Register **(000030$_H$)** **Application Reset Value: 0200 1001$_H$**



Field	Bits	Type	Description
BRG_MODE	0	rw	**Defines the operation mode for the AEI bridge** Reset value depends on the hardware configuration chosen by silicon vendor. BRG_MODE shall not be written with 0. 0_B AEI bridge operates in sync_bridge mode 1_B AEI bridge operates in async_bridge mode
MSK_WR_RSP	1	rw	**Mask write response** With active write buffer MSK_WR_RSP=1, execution of actions can be delayed due to previous inserted write actions in the transaction buffer which wait to be serviced. This can lead to the fact that an access on the bus to a different peripheral than the GTM might be executed earlier in time than the write access buffered in the GTM. Applications must be setup up with this in mind otherwise unexpected operation can happen. 0_B Do not mask the write response. Depending on the selected address the latency for execution can vary due to GTM internal arbitration. After this time the status of the access will be signaled by the signal AEI_STATUS to the bus interface. 1_B Mask write response. The write buffer of the bridge is activated, the actual access will be stored to the write buffer, and without latency on the bus interface; the acceptance of the access is signaled. AEI_STATUS=0b00 will be signaled. In case of a full write buffer, the actual access will be postponed until the next write buffer entry becomes free. Note: The status of the executed write accesses can be observed by using the notify bits AEI_USP_ADDR ,AEI_IM_ADDR, AEI_USP_BE in the register GTM_IRQ_NOTIFY.
BYPASS_SYNC	2	rw	**Bypass synchronizer flipflops** Function only available with BRG_MODE=1 0_B Synchronizer flip-flops in use, latency increase due to synchronization (aei_clk -> aei_sys_clk and back aei_sys_clk -> aei_clk). This setting must be used if aei_clk and aei_sys_clk operate fully asynchronous by independent clock sources. 1_B Synchronizer flip-flops are bypassed. No additional latency due to synchronization. This setting can be used if aei_clk and aei_sys_clk are generated by clock gating or clock division out of a common clock source. Clock edges on aei_clk and aei_sys_clk generated out of the same clock edge of the common clock source must have zero skew.
MODE_UP_PGR	8	r	**Mode update in progress** 0_B No update in progress 1_B Update in progress
BUFF_OVL	9	r	**Buffer overflow register** A buffer overflow can occur while multiple aborts are issued by the external bus or a pipelined instruction is started while FBC = 0 (see GTM_BRIDGE_PTR1 register). 0_B No buffer overflow occurred 1_B Buffer overflow occurred

Field	Bits	Type	Description
SYNC_INPUT_REG	12	r	**Additional pipelined stage in synchronous bridge mode** Reset value depends on the hardware configuration chosen by silicon vendor. 0_B No additional pipelined stage implemented 1_B Additional pipelined stage implemented. All accesses in synchronous mode will be increased by one clock cycle.
BRG_RST	16	rw	**Bridge software reset** This bit is cleared automatically after write. 0_B No bridge reset request 1_B Bridge reset request
BUFF_DPT	31:24	r	**Buffer depth of AEI bridge** Signals the buffer depth of the GTM AEI bridge implementation. Reset value depends on the hardware configuration chosen by silicon vendor.
0	7:3, 11:10, 15:13, 23:17	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.11 Register GTM_BRIDGE_PTR1

GTM AEI Bridge Pointer 1 Register

Note: This register operates on the AEI_CLK domain.

Note: This register holds diagnosis information about the AEI bus bridge. Each access to the GTM_IP will update the defined pointer bit fields. Depending on the mode of GTM_MODE_BRIDGE (BRG_MODE, MSK_WR_RESP), the AEI protocol and operating frequency which is use, the 4 pointer bit fields will change at different clock cycles relative to the start of the transaction. This leads to the fact that reading the register can show values not equal to the defined Initial Value, even directly after a write to GTM_BRIDGE_MODE with BRG_RST=1 was done.

BRIDGE_PTR1
GTM AEI Bridge Pointer 1 Register **(000034_H)** **Application Reset Value: $0020\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSP_TRAN_RDY						FBC						ABT_TRAN_PGR			
r						r						r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABT_TRAN_PGR	TRAN_IN_PGR					FIRST_RSP_PTR					NEW_TRAN_PTR				
r	r					r					r				

Field	Bits	Type	Description
NEW_TRAN_PTR	4:0	r	**New transaction pointer** Signals the actual value of the new transaction pointer.

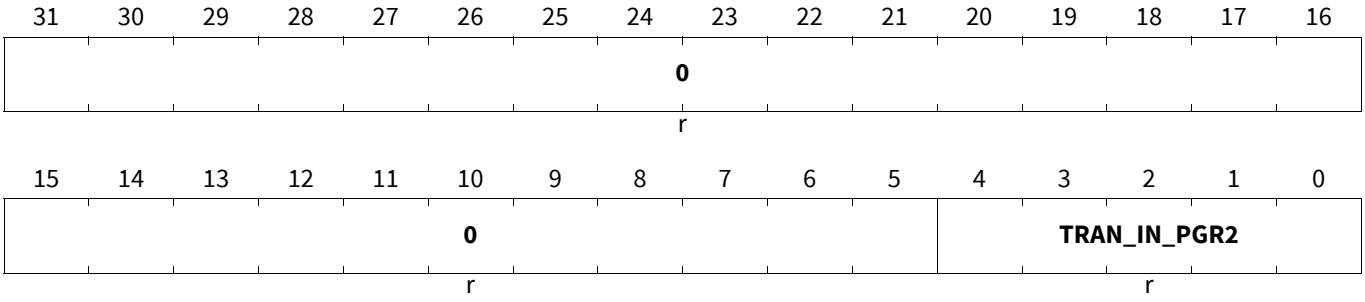
Field	Bits	Type	Description
FIRST_RSP_PTR	9:5	r	**First response pointer** Signals the actual value of first response pointer.
TRAN_IN_PGR	14:10	r	**Transaction in progress pointer (acquire)** Transaction in progress pointer.
ABT_TRAN_PGR	19:15	r	**Aborted transaction in progress pointer** Aborted transaction in progress pointer.
FBC	25:20	r	**Free buffer count** Number of free buffer entries. Initial value depends on the hardware configuration chosen by silicon vendor. (see BUFF_DPT in GTM_BRIDGE_MODE register).
RSP_TRAN_RDY	31:26	r	**Response transactions ready.** Amount of ready response transactions.

28.4.9.12 Register GTM_BRIDGE_PTR2

GTM AEI Bridge Pointer 2 Register

Note: This register operates on the GTM_CLK domain.

BRIDGE_PTR2
GTM AEI Bridge Pointer 2 Register **(000038_H)** **Application Reset Value: $0000\ 0000_H$**

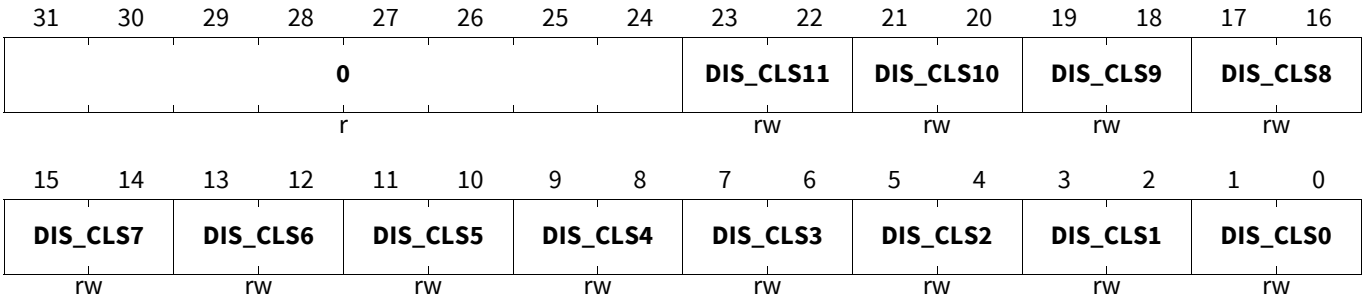


Field	Bits	Type	Description
TRAN_IN_PGR2	4:0	r	**Transaction in progress pointer (aquire2)** Transaction in progress pointer 2.
0	31:5	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.13 Register GTM_MCS_AEM_DIS

GTM MCS Master Port Disable Register

MCS_AEM_DIS
GTM MCS Master Port Disable Register (00003C$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
DIS_CLS0	1:0	rw	**Disable MCS AEIM access in cluster 0** Multicore encoding in use (**DIS_CLSx(1)** defines the state of the signal) Any read access to a **DIS_CLSx** bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored. 00_B State is 0; MCS AEM access in cluster x enabled (ignore write access) 01_B Change state to 0 10_B Change state to 1 11_B State is 1; MCS AEM access in cluster x disabled (ignore write access)
DIS_CLS1	3:2	rw	**Disable MCS AEIM access in cluster 1, see bit DIS_CLS0**
DIS_CLS2	5:4	rw	**Disable MCS AEIM access in cluster 2, see bit DIS_CLS0**
DIS_CLS3	7:6	rw	**Disable MCS AEIM access in cluster 3, see bit DIS_CLS0**
DIS_CLS4	9:8	rw	**Disable MCS AEIM access in cluster 4, see bit DIS_CLS0**
DIS_CLS5	11:10	rw	**Disable MCS AEIM access in cluster 5, see bit DIS_CLS0**
DIS_CLS6	13:12	rw	**Disable MCS AEIM access in cluster 6, see bit DIS_CLS0**
DIS_CLS7	15:14	rw	**Disable MCS AEIM access in cluster 7, see bit DIS_CLS0**
DIS_CLS8	17:16	rw	**Disable MCS AEIM access in cluster 8, see bit DIS_CLS0**
DIS_CLS9	19:18	rw	**Disable MCS AEIM access in cluster 9, see bit DIS_CLS0**
DIS_CLS10	21:20	rw	**Disable MCS AEIM access in cluster 10, see bit DIS_CLS0**
DIS_CLS11	23:22	rw	**Disable MCS AEIM access in cluster 11, see bit DIS_CLS0**
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.14 Register GTM_EIRQ_EN

GTM Error Interrupt Enable Register

EIRQ_EN
GTM Error Interrupt Enable Register **(000020_H)** **Application Reset Value: $0000\ 0180_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							CLK_PER_ERR_EIRQ_EN	CLK_EN_ERR_EIRQ_EN	AEIM_USP_BE_EIRQ_EN	AEIM_IM_ADDR_EIRQ_E	AEIM_USP_ADDR_EIRQ_E	AEI_USP_BE_EIRQ_EN	AEI_IM_ADDR_EIRQ_E	AEI_USP_ADDR_EIRQ_E	AEI_TO_XPT_EIRQ_EN
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
AEI_TO_XPT_EIRQ_EN	0	rw	**AEI_TO_XPT_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
AEI_USP_ADDR_EIRQ_EN	1	rw	**AEI_USP_ADDR_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
AEI_IM_ADDR_EIRQ_EN	2	rw	**AEI_IM_ADDR_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
AEI_USP_BE_EIRQ_EN	3	rw	**AEI_USP_BE_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
AEIM_USP_ADDR_EIRQ_EN	4	rw	**AEIM_USP_ADDR_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
AEIM_IM_ADDR_EIRQ_EN	5	rw	**AEIM_IM_ADDR_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
AEIM_USP_BE_EIRQ_EN	6	rw	**AEIM_USP_BE_EIRQ error interrupt enable** 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
CLK_EN_ERR_EIRQ_EN	7	rw	**CLK_EN_ERR_EIRQ interrupt enable** Read as zero in case of INT_CLK_EN_GEN = 0b1. 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM

Field	Bits	Type	Description
CLK_PER_ERR_EIRQ_EN	8	rw	**CLK_PER_ERR_EIRQ interrupt enable** Read as zero in case of INT_CLK_EN_GEN = 0b1. Read as zero, shall be written as zero. 0_B Disable error interrupt, interrupt is not visible outside GTM 1_B Enable error interrupt, interrupt is visible outside GTM
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.15 Register GTM_CLS_CLK_CFG

GTM Cluster Clock Configuration

Note: For clusters greater than 4 (only MAX 100 MHz capable), the allowed setting for the CLS_CLK_DIV are 00_B and 10_B (clock divider 2). For clusters < 5, 200 MHz is available. In case a device has a single 100 MHz cluster, the ARU will run with 100 MHz.

Note: Writing a value to a bit field CLS[c]_CLK_DIV that is not available in the device, an AEI status 10_B is returned.

Note: The availability of configuration bits is indicated by value of bit CFG_CLOCK_RATE in register CCM[c]_HW_CFG. If CFG_CLOCK_RATE=0, only the values 00_B and 01_B are valid for bit fields CLS[c]_CLK_DIV.

CLS_CLK_CFG
GTM Cluster Clock Configuration **($0000B0_H$)** **Application Reset Value: $00AA\ AAAA_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								CLS11_CLK_DIV		CLS10_CLK_DIV		CLS9_CLK_DIV		CLS8_CLK_DIV	
r								rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLS7_CLK_DIV		CLS6_CLK_DIV		CLS5_CLK_DIV		CLS4_CLK_DIV		CLS3_CLK_DIV		CLS2_CLK_DIV		CLS1_CLK_DIV		CLS0_CLK_DIV	
rw		rw		rw		rw		rw		rw		rw		rw	

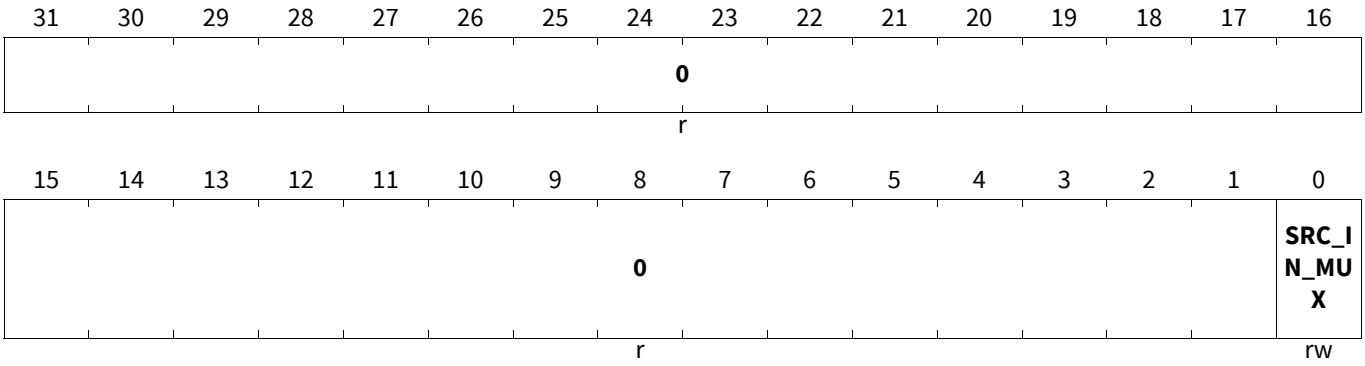
Field	Bits	Type	Description
CLSc_CLK_DIV (c=0-11)	2*c+1:2*c	rw	**Cluster c Clock Divider** This bit is only writable if bit field **RF_PROT** of register **GTM_CTRL** is cleared. 00_B Cluster c is disabled 01_B Cluster c is enabled without clock divider 10_B Cluster c is enabled with clock divider 11_B Reserved, do not use.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.4.9.16 Register GTM_CFG

GTM Configuration Register

CFG

GTM Configuration Register **(000028_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SRC_IN_MUX	0	rw	**GTM_TIM[i]_AUX_IN input source selection** See **Figure 9** for details. 0_B Use for TIM[i] output of TOM[n] 1_B Use for TIM[i] output of TOM[i] (same cluster)
0	31:1	r	**Reserved** Read as zero, shall be written as zero.

28.5 Advanced Routing Unit (ARU)

28.5.1 Overview

The Advanced Routing Unit (ARU) is a flexible infrastructure component for transferring 53 bit wide data (five control bits and two 24 bit values) between several sub-modules of the GTM core in a configurable manner.

Since the concept of the ARU has already been described in the paragraph “ARU routing concept”, this section only describes additional ARU features that can be used by the software for configuring and debugging ARU related data streams.Also the definition of 'streams' and 'channels' in the ARU context is done in “ARU routing concept”.

The principle of ARU data routing is described in “ARU Write Address Overview”. In the real GTM implementation the ARU serves in parallel per clock period two individual data destinations, one destination at port ARU-0 and at port ARU-1. Both ARU ports ARU-0 and ARU-1 are running by default in parallel but can be configured in dynamic routing mode (see below) to run in an individual mode.

As already defined in the “ARU routing concept”, the ARU read ID is the address of the data source that is configured in the data destination module. These ARU read ID's are selected by the individual counter of ARU ports ARU-0 and ARU-1.

Via the ARU ports ARU-0 and ARU-1 with each ARU read ID two independent GTM sub-modules are addressed and served. The combination of ARU port (ARU-0 or ARU-1) and the ARU read ID addresses one ARU wdata source (i.e. the ARU write port of a GTM sub-module).

The assignment of ARU write ports of GTM sub-modules to the ARU ports ARU-0 and ARU-1 and the ARU read ID's is device depending and can be found in the corresponding sub-chapter.

28.5.2 Special Data Sources

Besides the addresses of the sub-module related data sources as described in product specific appendix, the ARU provides two special data sources that can be used for the configuration of data streams. These data sources are defined as follows:

Address 0x1FF: Data source that provides always a 53 bit data word with zeros. A read access to this memory location will never block a requesting data destination.

Address 0x1FE: Data source that never provides a data word. A read access to this memory location will always block a requesting data destination. This is the reset value of the read registers inside the data destinations.

Address 0x000: This address is reserved and can be used to bring data through the ARU registers **ARU_DATA_H** and **ARU_DATA_L** into the system by writing the write address 0x000 into the **ARU_ACCESS** register. This means that software test data can be brought into the GTM by the CPU.

28.5.3 ARU Access via AEI

Besides the data transfer between the connected sub-modules, there are two possibilities to access ARU data via the AEI.

28.5.3.1 Default ARU Access

The default ARU access incorporates the registers **ARU_ACCESS**, which is used for initiation of a read or write request and the registers **ARU_DATA_H** and **ARU_DATA_L** that provide the ARU data word to be transferred.

The status of a read or write transfer can be determined by polling specific bits in register **ARU_ACCESS**. Furthermore the *acc_ack* bit in the interrupt notify register is set after the read or write access is performed to avoid data loss e.g. on access cancelation.

A pending read or write request may also be canceled by clearing the associated bit.

In the case of a read request, the AEI access behaves as a read request initiated by a data destination of a module. The read request is served by the ARU immediately when no other destination has a pending read request. This means, that an AEI read access does not take part in the scheduling of the destination channels and that the time between two consecutive read accesses is not limited by the round trip time.

On the other hand, the AEI access has the lowest priority behind the ARU scheduler that serves the destination channels. Thus, in worst case, the read request is served after one round trip of the ARU, when all destination channels would request data at the same point in time.

In the case of the write request, the ARU provides the write data at the address defined by the ADDR bit field inside the **ARU_ACCESS** register.

To avoid data loss, the reserved ARU address 0x0 has to be used to bring data into the system. Otherwise, in case the address specified inside the ADDR bit field is defined for another sub-module that acts as a source at the ARU data loss may occur and no deterministic behavior is guaranteed.

This is because the regular source sub-module is not aware that its address is used by the ARU itself to provide data to a destination.

It is guaranteed that the ARU write data is send to the destination in case of both modules want to provide data at the same time.

Configuring both read and write request bits results in a read request, if the write request bit inside the register isn't already set. The read request bit will be set but not the write request bit. The following table describes the important cases of the bit 12 (RREQ) and bit 13 (WREQ) of the **ARU_ACCESS** register:

Table 12 WREQ and RREQ in ARU_ACCESS register

AEI write access: aei_wdata (13:12)	actual value of ARU_ACCESS(13:12)	next value of ARU_ACCESS(13:12)	comment
0 0	0 1	0 0	cancel read request
0 0	1 0	0 0	cancel write request
0 1	1 0	1 0	unchanged register
1 0	0 1	0 1	unchanged register
1 1	0 0	0 1	both read and write request results in a read request
1 1	1 0	1 0	as before but WREQ bit is already set -> unchanged register

28.5.3.2 Debug Access

The debug access mode enables to inspect routed data of configured data streams during runtime.

The ARU provides two independent debug channels, whereas each is configured by a dedicated ARU read address in register **ARU_DBG_ACCESS0** and **ARU_DBG_ACCESS1** respectively.

The registers **ARU_DBG_DATA0_H** and **ARU_DBG_DATA0_L** (**ARU_DBG_DATA1_H** and **ARU_DBG_DATA1_L**) provide read access to the latest data word that the corresponding data source sent through the ARU.

Any time when data is transferred through the ARU from a data source to the destination requesting the data the interrupt signal *ARU_NEW_DATA0*_IRQ (*ARU_NEW_DATA1*_IRQ) is raised.

For advanced debugging purposes, the interrupt signal can also be triggered by software using the register **ARU_IRQ_FORCINT**.

Please note, that the debug mechanism should not be used by the application, when a HW-Debugger is used to trace the ARU communication. In that case, the debug registers are used by the HW-Debugger to specify the ARU streams that should be traced.

28.5.4 ARU dynamic routing

A dynamic routing feature of the ARU is implemented and can be configured using the additional AEI registers**:**,

- **ARU_CTRL**
- **ARU_[x]_DYN_CTRL**
- **ARU_[x]_DYN_RDADDR**
- **ARU_[x]_DYN_ROUTE_LOW**
- **ARU_[x]_DYN_ROUTE_HIGH**
- **ARU_[x]_DYN_ROUTE_SR_LOW**
- **ARU_[x]_DYN_ROUTE_SR_HIGH**

For further information see the register part of this chapter.

28.5.4.1 Dynamic routing - CPU controlled

The dynamic routing feature can be enabled separately for ARU-0 and ARU-1 by setting the corresponding bit fields of the register **ARU_CTRL**.

The enabling of the dynamic routing feature is synchronized to the normal routing scheme if ARU master ID-0 is addressed. The dynamic route will started with additional ARU master DYN_READ_ID0.

With the dynamic routing feature it is possible to insert additional ARU master ID's, DYN_READ_IDy (y:0-5), in a defined manner into the normal ARU routing scheme.

While inserting additional ARU master ID's the normal ARU routing scheme is paused. Therefore please consider that inserting additional ARU master ID's will lengthen the normal routing scheme.

It is possible to configure 6 additional ARU master ID's in the **ARU_[x]_DYN_ROUTE_LOW/_HIGH** registers for both ARU-0 and ARU-1.

In the bit field **DYN_CLK_WAIT** of **ARU_[x]_DYN_ROUTE_HIGH** register the number of clock cycles has to be configured, after which one of the additional ARU master ID's will be inserted.

After each configured number of clock cycles the defined ARU master ID's will be inserted cyclic one after each other in the following manner:
… -> DYN_READ_ID0 -> DYN_READ_ID1 -> DYN_READ_ID2 -> DYN_READ_ID3 -> DYN_READ_ID4 -> DYN_READ_ID5 -> DYN_READ_ID0 -> ….

In the shadow registers **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** further 6 ARU master, DYN_READ_IDy (y:6-11), can be configured.

The bit DYN_UPDATE_EN in the **ARU_[x]_DYN_ROUTE_SR_HIGH** register controls whether the **ARU_[x]_DYN_ROUTE_LOW/_HIGH** registers are updated from its shadow registers except DYN_UPDATE_EN, it is not updated. The update is executed once after writing **ARU_[x]_DYN_ROUTE_SR_HIGH**. If update started DYN_UPDATE_EN is reset.

With the DYN_ROUTE_SWAP option in the **ARU_[x]_DYN_CTRL** register it is possible to swap the registers **ARU_[x]_DYN_ROUTE_LOW/HIGH** with its shadow registers **ARU_[x]_DYN_ROUTE_SR_LOW/HIGH**. The swapping is executed always after the 6 ARU master DYN_READ_ID's are inserted. So it is possible to insert a maximum of 12 ARU master DYN_READ_ID's cyclic after each configured number of clock cycles. If swap started DYN_UPDATE_EN is reset.

Setting the bit field **DYN_CLK_WAIT** of **ARU_[x]_DYN_ROUTE_HIGH** register to zero, only the defined ARU master DYN_READ_ID's will be executed. The normal ARU routing scheme is stopped.

Setting the bit field **DYN_CLK_WAIT** of **ARU_[x]_DYN_ROUTE_HIGH** register to 15, only the normal ARU routing scheme is executed. Inserting of additional ID's is stopped.

To reset the ARU caddr counter and ARU dynamic route counter set bit ARU_ADDR_RSTGLB of **CMU_GLB_CTRL** following by a write access to register **CMU_CLK_EN**.

28.5.4.1.1 Dynamic routing ring mode

In dynamic routing ring mode it is possible to use all 24 DYN_READ_ID's from both ARU-0 and ARU-1 by setting bit field ARU_DYN_RING_MODE in **ARU_CTRL** register to 1. In this mode all 4 registers **ARU_[x]_DYN_ROUTE_LOW/_HIGH** and **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** are connected as a ring, so all 24 DYN_READ_ID's can be used from both ARU's. The ring structure is shown in **Figure 24**. The data register shift direction is shown by the arrows in the ring.

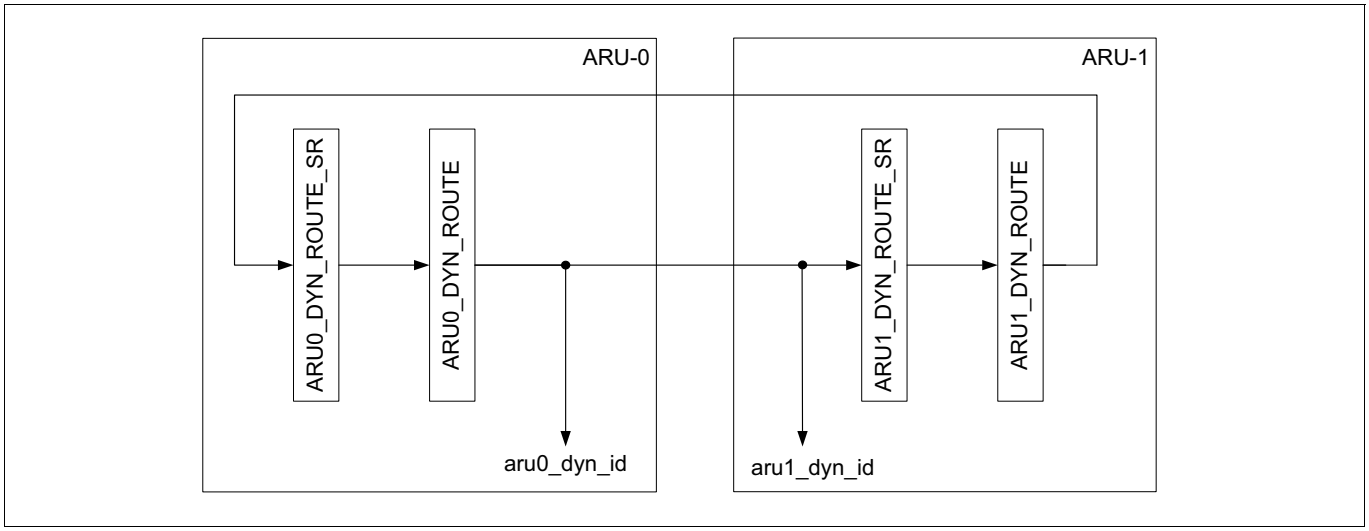


Figure 24 ARU dynamic routing - ring mode

Enabling the dynamic routing ring mode will automatically reset the caddr counter of both ARU-0 and ARU-1. This is necessary to synchronize both ARU's in this mode.

Enabling the dynamic routing ring mode will ignored DYN_ROUTE_SWAP and DYN_UPDATE_EN.

Note: DYN_ARU_UPDATE_EN should be disabled in dynamic routing ring mode.

It is possible to enable the dynamic routing ring mode for both ARU-0 and ARU-1 or only for one of the ARU's by setting the corresponding bit field ARU_0_DYN_EN/ARU_1_DYN_EN of the register **ARU_CTRL.**

Because of the fact that to each ARU port ARU-0 and ARU-1 with the same ARU read ID two different GTM sub-modules are served it may make sense to enable ARU dynamic routing only for one port ARU-0 or ARU-1 if configured to ring-mode. The other port is then served in the default round robin manner.

In dynamic routing ring mode **ARU_[x]_DYN_ROUTE_LOW/_HIGH** and **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** are not write-protected. NOTE: Avoid modification of **ARU_[x]_DYN_ROUTE_LOW/_HIGH** and **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** in active dynamic routing ring mode.

28.5.4.2 Dynamic routing - ARU controlled

Furthermore it is possible to reload the **ARU_[x]_DYN_ROUTE_SR_LOW/_HIGH** registers by ARU itself.

Therefore the ARU has its own master port which will be served in the normal ARU routing scheme. The ARU read address for this master port has to be configured in the register **ARU_[x]_DYN_RDADDR.**

This feature can be enabled by setting bit DYN_ARU_UPDATE_EN of **ARU_[x]_DYN_CTRL** register.

The following mapping of the ARU word to the **ARU_[x]_DYN_ROUTE_LOW/_HIGH** registers is implemented:

- **ARU_[x]_DYN_ROUTE_SR_LOW(23:0) = aru_data(23:0)**
- **ARU_[x]_DYN_ROUTE_SR_HIGH(28:0) = aru_data(52:24)**

The bit field aru_data(51:48) controls the configuration bits DYN_CLK_WAIT and the bit aru_data(52) controls the configuration bit DYN_UPDATE_EN. Both functions are described in **Section 28.5.4.1**.

In opposite to the dynamic routing scheme controlled from CPU/AEI (only the 6 additional ARU master DYN_REA_ID's are inserted) two additional ID's are served. One is the ARU master ID itself for reloading and the other is the default ID-0. The ID-0 is only added to the inserted routing scheme if bit field **DYN_CLK_WAIT** of **ARU_[x]_DYN_ROUTE_HIGH** is set to zero (only the inserted routing scheme is executed). This ensures that a debug access can take place even if only the inserted routing scheme is executed.

The following dynamic routing scheme is executed for 15 > **DYN_CLK_WAIT** > 0:
… -> ARU-master_ID -> DYN_READ_ID0 ->DYN_READ_ID1 -> DYN_READ_ID2 -> DYN_READ_ID3 -> DYN_READ_ID4 -> DYN_READ_ID5 -> ARU-master_ID -> …

The following dynamic routing scheme is executed for **DYN_CLK_WAIT** = 0:
… -> ARU-master_ID -> DYN_READ_ID0 -> DYN_READ_ID1 -> DYN_READ_ID2 -> DYN_READ_ID3 -> DYN_READ_ID4 -> DYN_READ_ID5 -> default_ID0 -> ARU-master_ID -> …

With the possibility of reloading the dynamic routing scheme over ARU, a FIFO or MCS is able to deliver the dynamic routing scheme data.

28.5.5 ARU Interrupt Signals

Table 13 ARU Interrupt Signals

Signal	Description
ARU_NEW_DATA0_IRQ	Indicates that data is transferred through the ARU using debug channel **ARU_DBG_ACCESS0**.
ARU_NEW_DATA1_IRQ	Indicates that data is transferred through the ARU using debug channel **ARU_DBG_ACCESS1**.
ARU_ACC_ACK_IRQ	ARU access acknowledge IRQ.

28.5.6 ARU Configuration Register Overview

Table 14 ARU Configuration Register Overview

Register name	Description	see Page
ARU_ACCESS	ARU access register	53
ARU_DATA_H	ARU access register upper data word	54
ARU_DATA_L	ARU access register lower data word	55
ARU_DBG_ACCESS0	ARU debug access channel 0	55
ARU_DBG_DATA0_H	ARU debug access 0 transfer register upper data word	57
ARU_DBG_DATA0_L	ARU debug access 0 transfer register lower data word	57
ARU_DBG_ACCESS1	ARU debug access channel 0	58
ARU_DBG_DATA1_H	ARU debug access 1 transfer register upper data word	59
ARU_DBG_DATA1_L	ARU debug access 1 transfer register lower data word	60
ARU_IRQ_NOTIFY	ARU interrupt notification register	60
ARU_IRQ_EN	ARU interrupt enable register	61
ARU_IRQ_FORCINT	ARU force interrupt register	62
ARU_IRQ_MODE	ARU interrupt mode register	63
ARU_CADDR_END	ARU caddr counter end value	63
ARU_CADDR	ARU caddr counter value	64
ARU_CTRL	ARU enable dynamic routing	65
ARU_[z]_DYN_CTRL	ARU z dynamic routing control register	66
ARU_[z]_DYN_RDADDR	ARU z read ID for dynamic routing	66
ARU_[z]_DYN_ROUTE_LOW	ARU z lower bits of DYN_ROUTE register	67
ARU_[z]_DYN_ROUTE_HIGH	ARU z higher bits of DYN_ROUTE register	68
ARU_[z]_DYN_ROUTE_SR_LOW	ARU z shadow register for ARU_[z]_DYN_ROUTE_LOW	68
ARU_[z]_DYN_ROUTE_SR_HIGH	ARU z shadow register for ARU_[z]_DYN_ROUTE_HIGH	69

28.5.7 ARU Configuration Register Description

28.5.7.1 Register ARU_ACCESS

ARU Access Register

Note: The register ARU_ACCESS can be used either for reading or for writing at the same point in time.

ARU_ACCESS
ARU Access Register **(000280$_H$)** **Application Reset Value: 0000 01FE$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		WREQ	RREQ	0			ADDR								
r		rw	rw	r			rw								

Field	Bits	Type	Description
ADDR	8:0	rw	**ARU address** Define the ARU address used for transferring data. For an ARU write request, the preferred address 0x0 have to be used. A write request to the address 0x1FF (always full address) or 0x1FE (always empty address) are ignored and doesn't have any effect. ARU address bits ADDR are only writable if RREQ and WREQ bits are zero.
RREQ	12	rw	**Initiate read request** This bit is cleared automatically after transaction. Moreover, it can be cleared by software to cancel a read request. RREQ bit is only writable if WREQ bit is zero, so to switch from RREQ to WREQ a cancel request has to be performed before. Configuring both RREQ and WREQ bits results in a read request, so RREQ bit will be set if the WREQ bit of the register isn't already set. The ARU read request on address ADDR is served immediately when no other destination has actually a read request when the RREQ bit is set by CPU. In a worst case scenario, the read request is served after one round trip of the ARU, but this is only the case when every destination channel issues a read request at consecutive points in time. 0_B No read request is pending 1_B Set read request to source channel addressed by ADDR

Field	Bits	Type	Description
WREQ	13	rw	**Initiate write request** This bit is cleared automatically after transaction. Moreover, it can be cleared by software to cancel a write request. WREQ bit is only writable if RREQ bit is zero, so to switch from WREQ to RREQ a cancel request has to be performed before. Configuring both RREQ and WREQ bits results in a read request, so WREQ bit will not be set The data is provided at address ADDR. This address has to be programmed as the source address in the destination sub-module channel. In worst case, the data is provided after one full ARU round trip. 0_B No write request is pending 1_B Mark data in registers ARU_DATA_H and ARU_DATA_L as valid
0	11:9, 31:14	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.2 Register ARU_DATA_H

ARU Access Register Upper Data Word

ARU_DATA_H
ARU Access Register Upper Data Word **(000284_H)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**DATA**												
r			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															
rw															

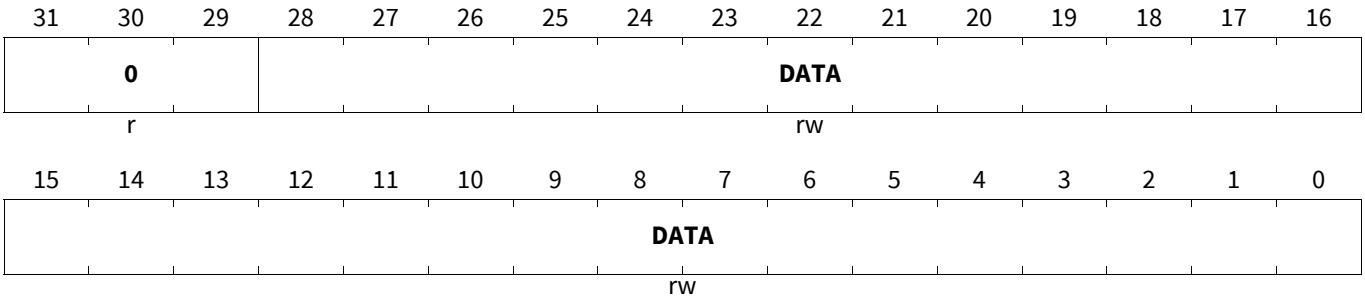
Field	Bits	Type	Description
DATA	28:0	rw	**Upper ARU data word** Transfer upper ARU data word addressed by ADDR. The data bits 24 to 52 of an ARU word are mapped to the data bits 0 to 28 of this register.
0	31:29	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.3 Register ARU_DATA_L

ARU Access Register Lower Data Word

ARU_DATA_L

ARU Access Register Lower Data Word **(000288$_H$)** **Application Reset Value: 0000 0000$_H$**



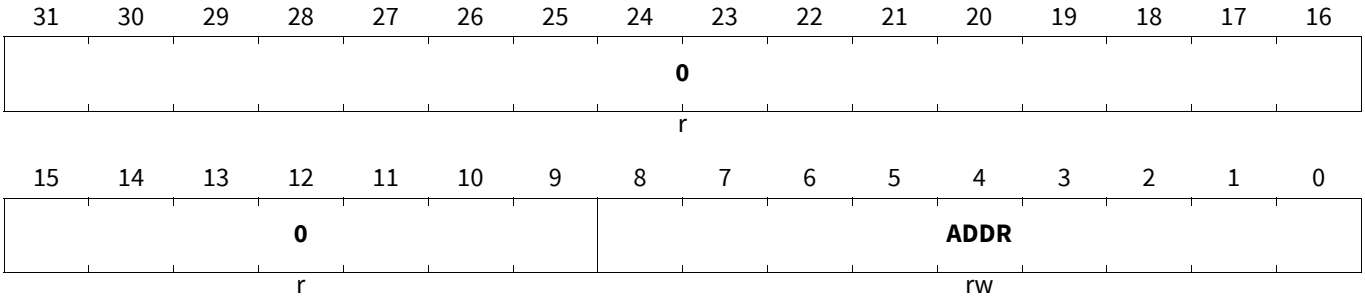
Field	Bits	Type	Description
DATA	28:0	rw	**Lower ARU data word** Transfer lower ARU data word addressed by ADDR. The data bits 0 to 23 of an ARU word are mapped to the data bits 0 to 23 of this register and the data bits 48 to 52 of an ARU word are mapped to the data bits 24 to 28 of this register when data is read by the CPU. For writing data into the ARU by the CPU the bits 24 to 28 are **not** transferred to bit 48 to 52 of the ARU word. Only bits 0 to 23 are written to bits 0 to 23 of the ARU word.
0	31:29	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.4 Register ARU_DBG_ACCESS0

ARU Debug Access Channel 0

ARU_DBG_ACCESS0

ARU Debug Access Channel 0 **(00028C$_H$)** **Application Reset Value: 0000 01FE$_H$**



Field	Bits	Type	Description
ADDR	8:0	rw	**ARU debugging address** Define address of ARU debugging channel 0.

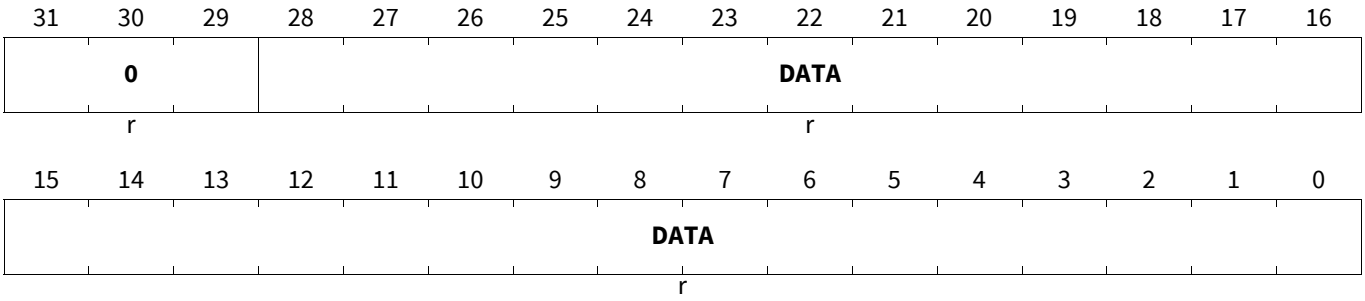
Field	Bits	Type	Description
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.5 Register ARU_DBG_DATA0_H

ARU Debug Access 0 Transfer Register Upper Data Word

ARU_DBG_DATA0_H

ARU Debug Access 0 Transfer Register Upper Data Word(000290$_H$) **Application Reset Value: 0000 0000$_H$**



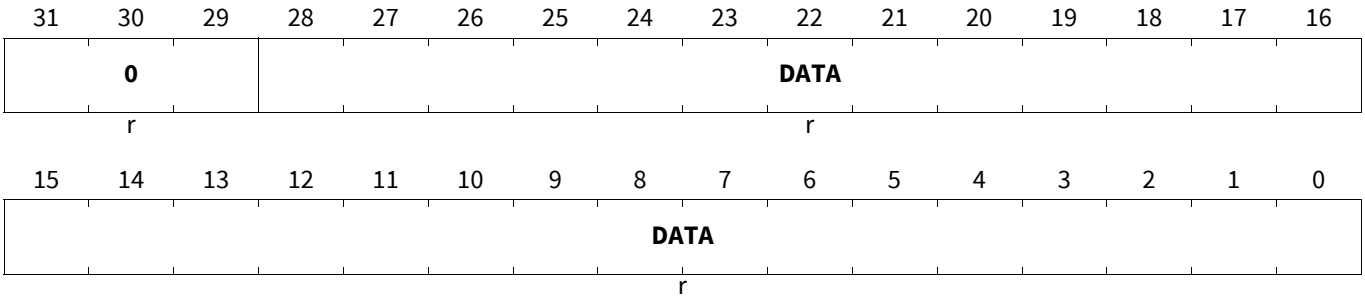
Field	Bits	Type	Description
DATA	28:0	r	**Upper debug data word** Transfer upper ARU data word addressed by register **DBG_ACCESS0**. The data bits 24 to 52 of an ARU word are mapped to the data bits 0 to 28 of this register The interrupt *ARU_NEW_DATA0_IRQ* is raised if a new data word is available.
0	31:29	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.6 Register ARU_DBG_DATA0_L

ARU Debug Access 0 Transfer Register Lower Data Word

ARU_DBG_DATA0_L

ARU Debug Access 0 Transfer Register Lower Data Word(000294$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DATA	28:0	r	**Lower debug data word** Transfer lower ARU data word addressed by register **DBG_ACCESS0**. The data bits 0 to 23 of an ARU word are mapped to the data bits 0 to 23 of this register and the data bits 48 to 52 of an ARU word is mapped to the data bits 24 to 28 of this register. The interrupt *ARU_NEW_DATA0_IRQ* is raised if a new data word is available.
0	31:29	r	**Reserved** Read as zero, shall be written as zero

28.5.7.7 Register ARU_DBG_ACCESS1

ARU Debug Access Channel 1

ARU_DBG_ACCESS1
ARU Debug Access Channel 1 **(000298_H)** **Application Reset Value: 0000 01FE$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**ADDR**								
r							rw								

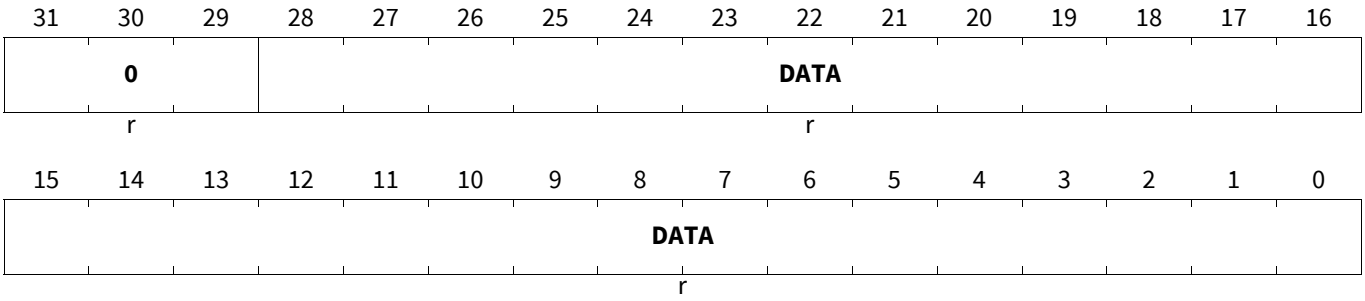
Field	Bits	Type	Description
ADDR	8:0	rw	**ARU debugging address** Define address of ARU debugging channel 1.
0	31:9	r	**Reserved** Read as zero, shall be written as zero

28.5.7.8 Register ARU_DBG_DATA1_H

ARU Debug Access 1 Transfer Register Upper Data Word

ARU_DBG_DATA1_H

ARU Debug Access 1 Transfer Register Upper Data Word(00029C$_H$) **Application Reset Value: 0000 0000$_H$**



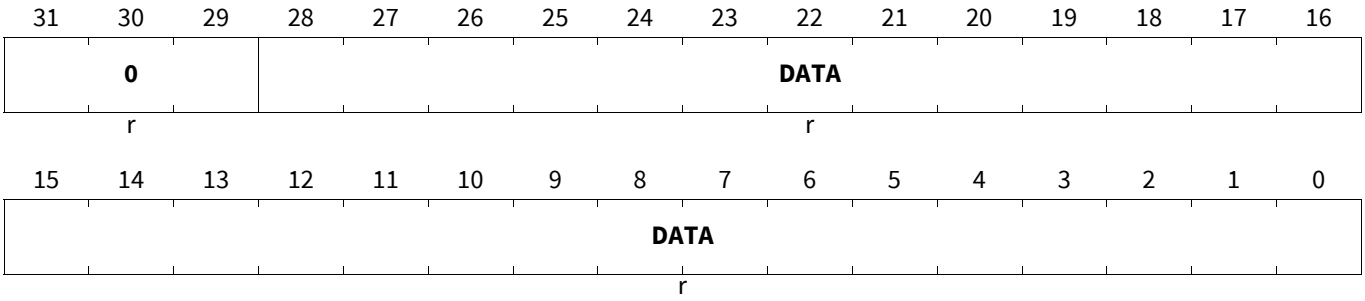
Field	Bits	Type	Description
DATA	28:0	r	**Upper debug data word** Transfer upper ARU data word addressed by register **DBG_ACCESS1**. The data bits 24 to 52 of an ARU word are mapped to the data bits 0 to 28 of this register. The interrupt *ARU_NEW_DATA1_IRQ* is raised if a new data word is available.
0	31:29	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.9 Register ARU_DBG_DATA1_L

ARU Debug Access 1 Transfer Register Lower Data Word

ARU_DBG_DATA1_L

ARU Debug Access 1 Transfer Register Lower Data Word(0002A0$_H$) **Application Reset Value: 0000 0000$_H$**



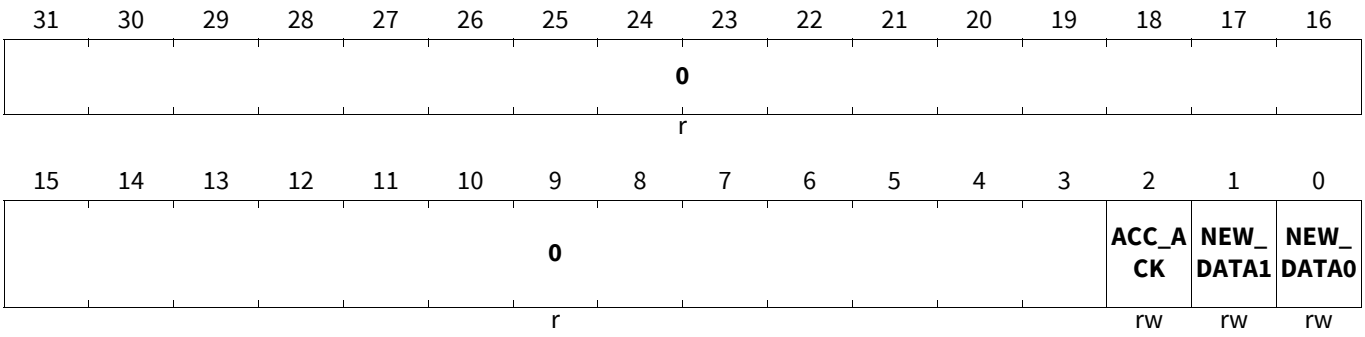
Field	Bits	Type	Description
DATA	28:0	r	**Lower debug data word** Transfer lower ARU data word addressed by register **DBG_ACCESS1**. The data bits 0 to 23 of an ARU word are mapped to the data bits 0 to 23 of this register and the data bits 48 to 52 of an ARU word is mapped to the data bits 24 to 28 of this register. The interrupt *ARU_NEW_DATA1_IRQ* is raised if a new data word is available.
0	31:29	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.10 Register ARU_IRQ_NOTIFY

ARU Interrupt Notification Register

ARU_IRQ_NOTIFY

ARU Interrupt Notification Register **(0002A4$_H$)** **Application Reset Value: 0000 0000$_H$**

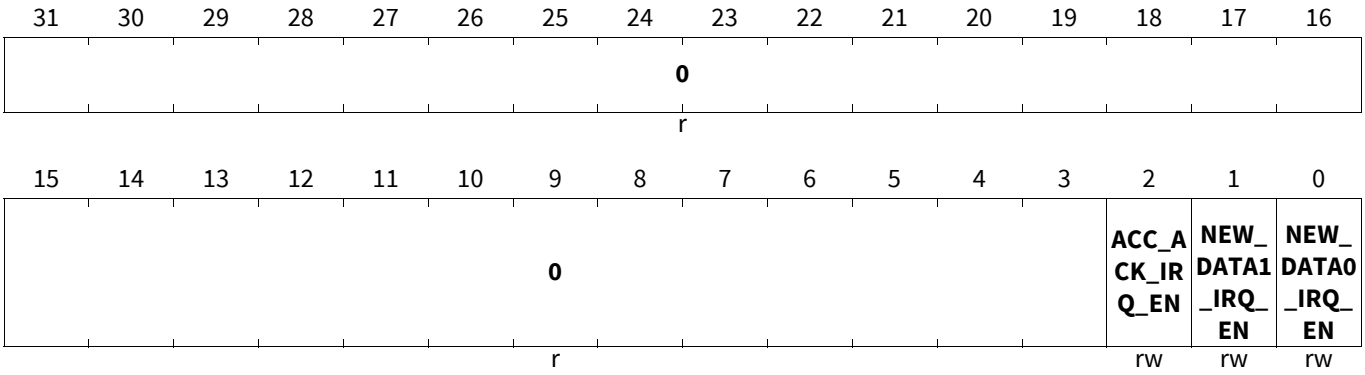


Field	Bits	Type	Description
NEW_DATA0	0	rw	**Data was transferred for addr ARU_DBG_ACCESS0** This bit will be cleared on a CPU write access of value '1'. (As the bit is rw, otherwise no clear.) A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B ARU_NEW_DATA0_IRQ interrupt was raised by the ARU
NEW_DATA1	1	rw	**Data was transferred for addr ARU_DBG_ACCESS1** This bit will be cleared on a CPU write access of value '1'. (As the bit is rw, otherwise no clear.) A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B ARU_NEW_DATA1_IRQ interrupt was raised by the ARU
ACC_ACK	2	rw	**AEI to ARU access finished, on read access data are valid** This bit will be cleared on a CPU write access of value '1'. (As the bit is rw, otherwise no clear.) A read access leaves the bit unchanged.
0	31:3	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.11 Register ARU_IRQ_EN

ARU Interrupt Enable Register

ARU_IRQ_EN
ARU Interrupt Enable Register **($0002A8_H$)** **Application Reset Value: $0000\ 0000_H$**



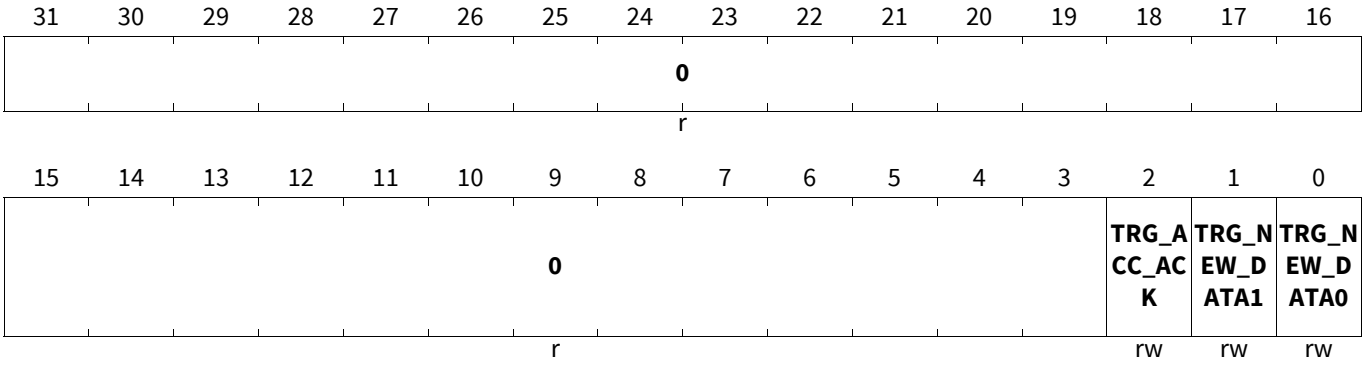
Field	Bits	Type	Description
NEW_DATA0_IRQ_EN	0	rw	**ARU_NEW_DATA0_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
NEW_DATA1_IRQ_EN	1	rw	**ARU_NEW_DATA1_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
ACC_ACK_IRQ_EN	2	rw	**ACC_ACK_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM

Field	Bits	Type	Description
0	31:3	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.12 Register ARU_IRQ_FORCINT

ARU Force Interrupt Register

ARU_IRQ_FORCINT
ARU Force Interrupt Register **(0002AC$_H$)** **Application Reset Value: 0000 0000$_H$**

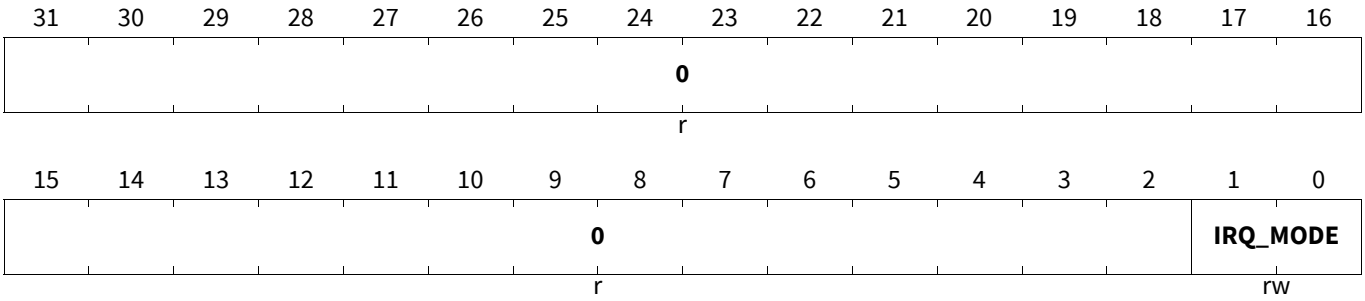


Field	Bits	Type	Description
TRG_NEW_DATA0	0	rw	**Trigger new data 0 interrupt** Note: This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B Corresponding bit in status register will not be forced 1_B Assert corresponding field in ARU_IRQ_NOTIFY register
TRG_NEW_DATA1	1	rw	**Trigger new data 1 interrupt** Note: This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B Corresponding bit in status register will not be forced 1_B Assert corresponding field in ARU_IRQ_NOTIFY register
TRG_ACC_ACK	2	rw	**Trigger ACC_ACK interrupt** Note: This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B Corresponding bit in status register will not be forced 1_B Assert corresponding field in ARU_IRQ_NOTIFY register
0	31:3	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.13 Register ARU_IRQ_MODE

ARU Interrupt Mode Register

ARU_IRQ_MODE
ARU Interrupt Mode Register **(0002B0$_H$)** **Application Reset Value: 0000 0000$_H$**

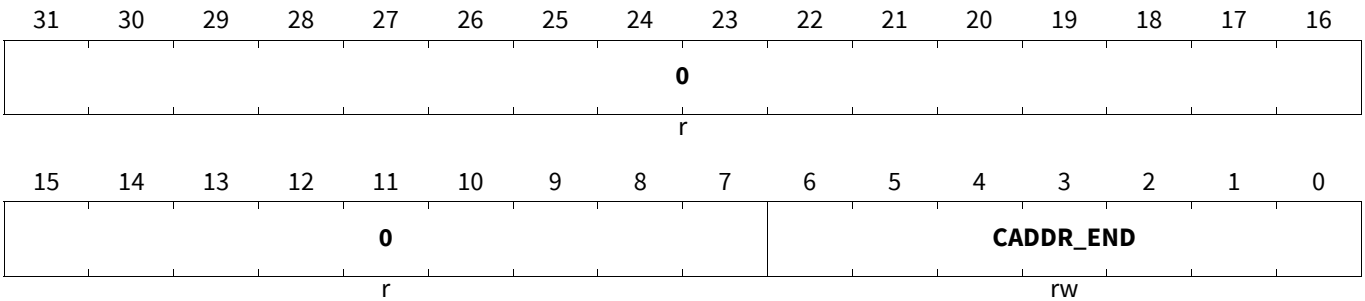


Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** The interrupt modes are described in **Section 28.4.5**. 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.14 Register ARU_CADDR_END

ARU caddr Counter End Value Register

ARU_CADDR_END
ARU caddr Counter End Value Register **(0002B4$_H$)** **Application Reset Value: 0000 007F$_H$**



Field	Bits	Type	Description
CADDR_END	6:0	rw	**Set end value of ARU caddr counter** The ARU roundtrip counter aru_caddr runs from zero to caddr_end value. Shorten the ARU roundtrip cycle by setting a smaller number than the defined reset value will cause that not all ARU-connected modules will be served. Making the roundtrip cycle longer than the reset value would cause longer ARU roundtrip time and as a result some ARU-connected modules will not be served as fast as possible for this device. This bit is write protected by bit RF_PROT of register GTM_CTRL
0	31:7	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.15 Register ARU_CADDR

ARU caddr Counter Value

Note: The registers CADDR_0 and CADDR_1 start incrementing with each clock cycle just after reset. Due to this the initial reset value cannot be read back.

ARU_CADDR

ARU caddr Counter Value **(0002FC$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									**CADDR_1**						
r									r						

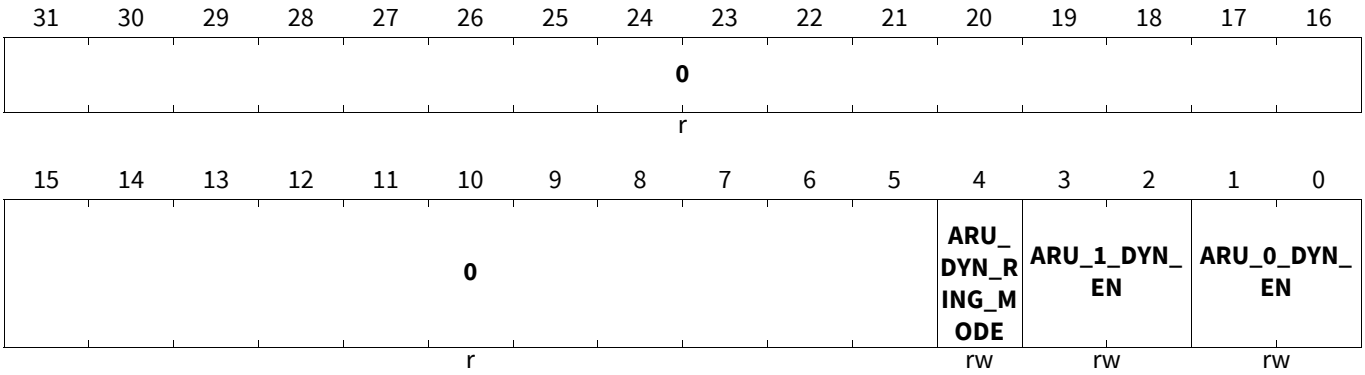
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									**CADDR_0**						
r									r						

Field	Bits	Type	Description
CADDR_0	6:0	r	**Value of ARU-0 caddr counter**
CADDR_1	22:16	r	**Value of ARU-1 caddr counter**
0	15:7, 31:23	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.16 Register ARU_CTRL

ARU Enable Dynamic Routing Register

ARU_CTRL
ARU Enable Dynamic Routing Register (0002BC$_H$) Application Reset Value: 0000 0000$_H$

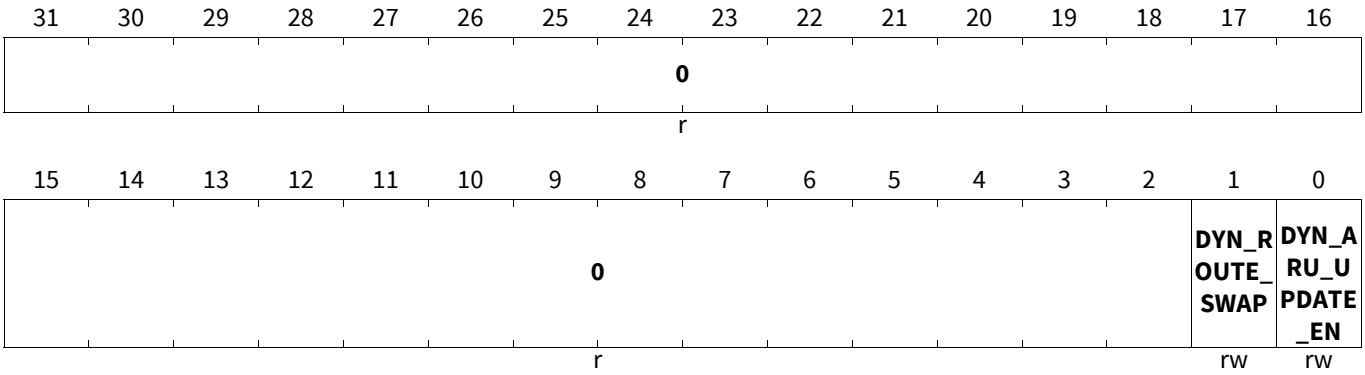


Field	Bits	Type	Description
ARU_0_DYN_EN	1:0	rw	**Enable dynamic routing for ARU-0** Dynamic routing enable of ARU-0. Write of following double bit values is possible: If dynamic routing is disabled, the normal ARU routing scheme for ARU-0 is executed. 00_B no change 01_B Disable dynamic routing 10_B Enable dynamic routing 11_B no change
ARU_1_DYN_EN	3:2	rw	**Enable dynamic routing for ARU-1** Dynamic routing enable of ARU-1. Write of following double bit values is possible: If dynamic routing is disabled, the normal ARU routing scheme for ARU-1 is executed. 00_B no change 01_B Disable dynamic routing 10_B Enable dynamic routing 11_B no change
ARU_DYN_RING_MODE	4	rw	**Enable dynamic routing ring mode** Dynamic routing ring mode for both ARU-0 and ARU-1. 0_B Different dynamic routing scheme for ARU-0 and ARU-11 1_B Same dynamic routing scheme for ARU-0 and ARU-1 with 24 possible read-ID's (dynamic routing ring mode)
0	31:5	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.17 Register ARU_[z]_DYN_CTRL

ARU z Dynamic Routing Control Register

ARU_z_DYN_CTRL (z=0-1)
ARU z Dynamic Routing Control Register (0002C0$_H$+z*4) Application Reset Value: 0000 0000$_H$

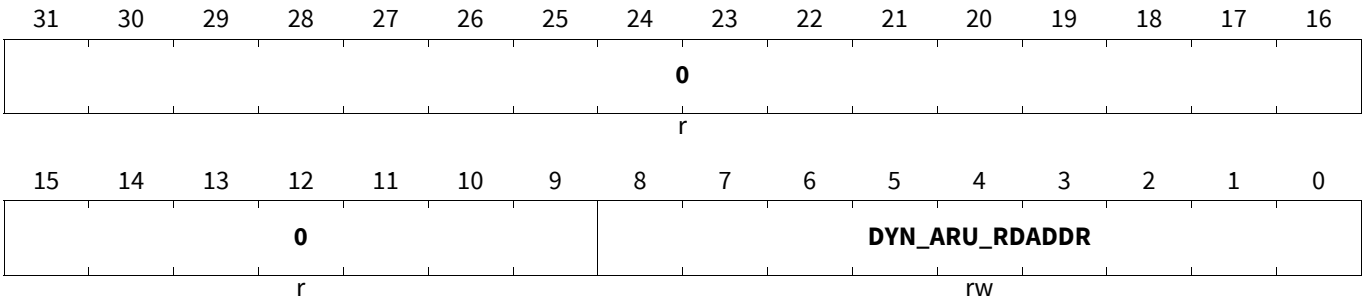


Field	Bits	Type	Description
DYN_ARU_UPDATE_EN	0	rw	**Enable reload of DYN_ROUTE register from ARU itself** Enable reload of DYN_ROUTE register from ARU itself.
DYN_ROUTE_SWAP	1	rw	**Enable swapping DYN_ROUTE_SR with DYN_ROUTE register** Enable swapping **DYN_ROUTE_SR** with **DYN_ROUTE** register.
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.18 Register ARU_[z]_DYN_RDADDR

ARU z Read ID for Dynamic Routing

ARU_z_DYN_RDADDR (z=0-1)
ARU z Read ID for Dynamic Routing (0002E8$_H$+z*4) Application Reset Value: 0000 0000$_H$



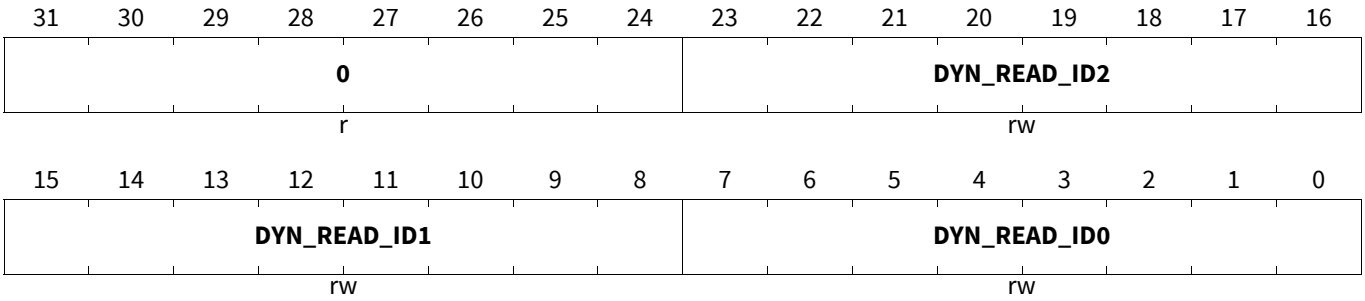
Field	Bits	Type	Description
DYN_ARU_RDADDR	8:0	rw	**ARU read address ID to reload the DYN_ROUTE register** ARU read address ID to reload the DYN_ROUTE register from ARU itself.
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.19 Register ARU_[z]_DYN_ROUTE_LOW

ARU z Lower Bits of DYN_ROUTE Register

ARU_z_DYN_ROUTE_LOW (z=0-1)

ARU z Lower Bits of DYN_ROUTE Register (0002C8$_H$+z*4) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
DYN_READ_ID0	7:0	rw	**ARU read ID 0** ARU read ID 0 for dynamic routing.
DYN_READ_ID1	15:8	rw	**ARU read ID 2** ARU read ID 1 for dynamic routing.
DYN_READ_ID2	23:16	rw	**ARU read ID 2** ARU read ID 2 for dynamic routing.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.5.7.20 Register ARU_[z]_DYN_ROUTE_HIGH

ARU z Higher Bits of DYN_ROUTE Register

ARU_z_DYN_ROUTE_HIGH (z=0-1)

ARU z Higher Bits of DYN_ROUTE Register (0002D0$_H$+z*4) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				DYN_CLK_WAIT				DYN_READ_ID5							
r				rw				rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DYN_READ_ID4								DYN_READ_ID3							
rw								rw							

Field	Bits	Type	Description
DYN_READ_ID3	7:0	rw	**ARU read ID 3** ARU read ID 3 for dynamic routing.
DYN_READ_ID4	15:8	rw	**ARU read ID 4** ARU read ID 4 for dynamic routing.
DYN_READ_ID5	23:16	rw	**ARU read ID 5** ARU read ID 5 for dynamic routing.
DYN_CLK_WAIT	27:24	rw	**Number of clk cycles for dynamic routing** Defines the number of clk cycles between each dynamic routing ID.
0	31:28	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.21 Register ARU_[z]_DYN_ROUTE_SR_LOW

ARU z Shadow Register for ARU_z_DYN_ROUTE_LOW

NOTE : This is the shadow register for register **ARU_[z]_DYN_ROUTE_LOW**

ARU_z_DYN_ROUTE_SR_LOW (z=0-1)

ARU z Shadow Register for ARU_z_DYN_ROUTE_LOW(0002D8$_H$+z*4) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								DYN_READ_ID8							
r								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DYN_READ_ID7								DYN_READ_ID6							
rw								rw							

Field	Bits	Type	Description
DYN_READ_ID6	7:0	rw	**ARU read ID 6** ARU read ID 6 for dynamic routing. These bits are mapped to ARU data bits aru_data(7:0).
DYN_READ_ID7	15:8	rw	**ARU read ID 7** ARU read ID 7 for dynamic routing. These bits are mapped to ARU data bits aru_data(15:8).
DYN_READ_ID8	23:16	rw	**ARU read ID 8** ARU read ID 8 for dynamic routing. These bits are mapped to ARU data bits aru_data(23:16).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.5.7.22 Register ARU_[z]_DYN_ROUTE_SR_HIGH

ARU z Shadow Register for ARU_z_DYN_ROUTE_HIGH

NOTE : This is the shadow register for register **ARU_[z]_DYN_ROUTE_HIGH**

ARU_z_DYN_ROUTE_SR_HIGH (z=0-1)

ARU z Shadow Register for ARU_z_DYN_ROUTE_HIGH(0002E0$_H$+z*4) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			DYN_UPDATE_EN	DYN_CLK_WAIT				DYN_READ_ID11							
r			rw	rw				rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DYN_READ_ID10								DYN_READ_ID9							
rw								rw							

Field	Bits	Type	Description
DYN_READ_ID9	7:0	rw	**ARU read ID 9** ARU read ID 9 for dynamic routing. These bits are mapped to ARU data bits aru_data(31:24).
DYN_READ_ID10	15:8	rw	**ARU read ID 10** ARU read ID 10 for dynamic routing. These bits are mapped to ARU data bits aru_data(39:32).
DYN_READ_ID11	23:16	rw	**ARU read ID 11** ARU read ID 11 for dynamic routing. These bits are mapped to ARU data bits aru_data(47:40).
DYN_CLK_WAIT	27:24	rw	**Number of clk cycles for dynamic routing** Defines the number of clk cycles between each dynamic routing ID. These bits are mapped to ARU data bits aru_data(51:48).

Field	Bits	Type	Description
DYN_UPDATE_EN	28	rw	**Update enable from shadow register** Enable update **ARU_[z]_DYN_ROUTE_LOW/_HIGH** registers from shadow registers **ARU_[z]_DYN_ROUTE_SR_LOW/_HIGH**. This bit is mapped to ARU data bit aru_data(52).
0	31:29	r	**Reserved** Read as zero, shall be written as zero.

28.6 Broadcast Module (BRC)

28.6.1 Overview

Since each write address for the sub-module channels of the GTM that are able to write to the ARU can only be read by a single module, it is impossible to provide a data stream to different modules in parallel (This statement holds not for sources, which do not invalidate their data after the data were read by any consumer, e.g. DPLL).

To overcome this issue for regular modules, the sub-module Broadcast (BRC) enables to duplicate data streams multiple times.

The BRC sub-module provides 12 input channels as well as 22 output channels.

In order to clone an incoming data stream, the corresponding input channel can be mapped to zero or more output channels.

When mapped to zero no channel is read.

To destroy an incoming data stream, the **EN_TRASHBIN** bit inside the **BRC_SRC_[x]_DEST** register has to be set.

The total number of output channels that are assigned to a single input channel is variable. However, the total number of assigned output channels must be less than or equal to 22.

28.6.2 BRC Configuration

As it is the case with all other sub-modules connected to the ARU, the input channels can read arbitrary ARU address locations and the output channels provide the broadcast data to fixed ARU write address locations.

The associated write addresses for the BRC sub-module are fixed and can be obtained from the product specific appendix.

The read address for each input channel is defined by the corresponding register **BRC_SRC_[x]_ADDR** (x: 0…11).

The mapping of an input channel to several output channels is defined by setting the appropriate bits in the register **BRC_SRC_[x]_DEST** (x: 0…11). Each output channel is represented by a single bit in the register **BRC_SRC_[x]_DEST**.

If no output channel bit is set within a register **BRC_SRC_[x]_DEST**, no data is provided to the corresponding ARU write address location from the defined read input specified by **BRC_SRC_[x]_ADDR**. This means that the channel does not broadcast any data and is disabled (reset state).

Besides the possibility of mapping an input channel to several output channels, the bit **EN_TRASHBIN** of register **BRC_SRC_[x]_DEST** may be set, which results in dropping an incoming data stream. In this case the data of an input channel defined by **BRC_SRC_[x]_ADDR** is consumed by the BRC module and not routed to any succeeding sub-module. In consequence, the output channels defined in the register **BRC_SRC_[x]_DEST** are ignored. Therefore, the bits 0 to 21 are set to zero (0) when trash bin functionality is enabled.

In general, the BRC sub-module can work in two independent operation modes. In the first operation mode the data consistency is guaranteed since a BRC channel requests only new data from a source when all destination channels for the BRC have consumed the old data value. This mode is called *Data Consistency Mode* (DCM).

In a second operation mode the BRC channel always requests data from a source and distributes this data to the destination regardless whether all destinations have already consumed the old data. This mode is called *Maximum Throughput Mode* (MTM).

MTM ensures that always the newest available data is routed through the system, while it is not guaranteed data consistency since some of the destination channels can be provided with the old data while some other destination channels are provided with the new data. If this is the case, the Data Inconsistency Detected Interrupt *BRC_DID_IRQ[x]* is raised but the channel continues to work.

Furthermore in MTM mode it is guaranteed that it is not possible to read a data twice by a read channel. This is blocked.

The channel mode can be configured inside the **BRC_SRC_[x]_ADDR** register.

To avoid invalid configurations of the registers **BRC_SRC_[x]_DEST**, the BRC also implements a plausibility check for these configurations. If the software assigns an already used output channel to a second input channel, BRC performs an auto correction of the lastly configured register **BRC_SRC_[x]_DEST** and it triggers the interrupt *BRC_DEST_ERR*.

Consider the following example for clarification of the auto correction mechanism. Assume that the following configuration of the 22 lower significant bits for the registers **BRC_SRC_[x]_DEST**:

BRC_SRC_0_DEST: 00 0000 0000 1000 1000 0000 (binary)

BRC_SRC_1_DEST: 00 0000 0000 0100 0000 0100 (binary)

BRC_SRC_2_DEST: 00 0000 0000 0001 0100 0010 (binary)

BRC_SRC_3_DEST: 00 0000 0000 0010 0001 1001 (binary)

If the software overwrites the value for register **BRC_SRC_2_DEST** with

BRC_SRC_2_DEST: 00 0000 0000 1001 0010 0010 (binary)

(changed bits are underlined), then the BRC releases a *BRC_DEST_ERR* interrupt since bit 11 is already assigned in register **BRC_SRC_0_DEST**. The auto correction forces bit 11 to be cleared. The modifications of the bits 5 and 6 are accepted, since there is no violation with previous configurations. So the result of the write access mentioned above results in the following modified register configuration:

BRC_SRC_2_DEST: 00 0000 0000 0001 0010 0010 (binary)

For debug purposes, the interrupt *BRC_DEST_ERR* can also be released by writing to register **BRC_IRQ_FORCINT**. Nevertheless, the interrupt has to be enabled to be visible outside of the GTM.

28.6.3 BRC Interrupt Signals

Table 15 BRC Interrupt Signals

Signal	Description
BRC_DEST_ERR_IRQ	Indicating configuration errors for BRC module
BRC_DID_IRQ[x]	Data inconsistency occurred in MTM mode (x:0…11)

28.6.4 BRC Configuration Register Overview

Table 16 BRC Configuration Register Overview

Register Name	Description	see Page
BRC_SRC_[z]_ADDR	BRC read address for input channel z	74
BRC_SRC_[z]_DEST	BRC destination channels for input channel z	75
BRC_IRQ_NOTIFY	BRC interrupt notification register	76
BRC_IRQ_EN	BRC interrupt enable register	77
BRC_EIRQ_EN	BRC error interrupt enable register	79
BRC_IRQ_FORCINT	BRC force interrupt register	78

Table 16 BRC Configuration Register Overview (cont'd)

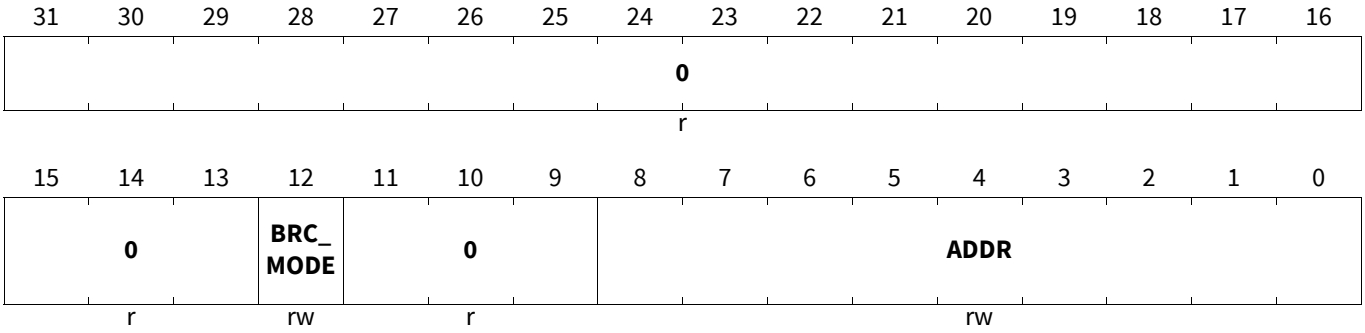
Register Name	Description	see Page
BRC_RST	BRC software reset register	80
BRC_IRQ_MODE	BRC interrupt mode configuration register	78

28.6.5 BRC Configuration Register Description

28.6.5.1 Register BRC_SRC_[z]_ADDR

BRC Read Address for Input Channel z

BRC_SRC_z_ADDR (z=0-11)
BRC Read Address for Input Channel z **(000400_H+z*8)** **Application Reset Value: 0000 $01FE_H$**



Field	Bits	Type	Description
ADDR	8:0	rw	**Source ARU address. Defines an ARU read address used as data source for input channel z** This bit field is only writable if channel is disabled.
BRC_MODE	12	rw	**BRC_MODE: BRC Operation mode select** This bit field is only writable if channel is disabled. 0_B Consistency Mode (DCM) selected 1_B Maximum Throughput Mode (MTM) selected
0	11:9, 31:13	r	**Reserved** Read as zero, shall be written as zero.

28.6.5.2 Register BRC_SRC_[z]_DEST

BRC Destination Channels for Input Channel z

BRC_SRC_z_DEST (z=0-11)

BRC Destination Channels for Input Channel z(000404_H+z*8) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									EN_TRASHBIN	EN_DEST21	EN_DEST20	EN_DEST19	EN_DEST18	EN_DEST17	EN_DEST16
r									rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_DEST15	EN_DEST14	EN_DEST13	EN_DEST12	EN_DEST11	EN_DEST10	EN_DEST9	EN_DEST8	EN_DEST7	EN_DEST6	EN_DEST5	EN_DEST4	EN_DEST3	EN_DEST2	EN_DEST1	EN_DEST0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
EN_DESTq (q=0-21)	q	rw	**Enable BRC destination address q** The bits 0 to 21 are cleared by auto correction mechanism if a destination channel is assigned to multiple source channels. When a BRC input channel is disabled (all EN_DESTq (q: 0...21) bits are reset to zero), the internal states are reset to their reset value. 0_B Destination address q not mapped to source BRC_SRC_[x]_ADDR 1_B Destination address q mapped to source BRC_SRC_[x]_ADDR
EN_TRASHBIN	22	rw	**EN_TRASHBIN: Control trash bin functionality** When bit **EN_TRASHBIN** is enabled bits 0 to 21 are ignored for this input channel. Therefore, the bits 0 to 21 are set to zero (0) when trash bin functionality is enabled. 0_B Trash bin functionality disabled 1_B Trash bin functionality enabled
0	31:23	r	**Reserved** Read as zero, shall be written as zero.

28.6.5.3 Register BRC_IRQ_NOTIFY

BRC Interrupt Notification Register

BRC_IRQ_NOTIFY
BRC Interrupt Notification Register **(000460_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			DID11	DID10	DID9	DID8	DID7	DID6	DID5	DID4	DID3	DID2	DID1	DID0	DEST_ERR
r			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
DEST_ERR	0	rw	**Configuration error interrupt for BRC sub-module** This bit will be cleared on a CPU write access of value '1'. (As the bit is rw, otherwise no clear.) A read access leaves the bit unchanged. 0_B No BRC configuration error occurred 1_B BRC configuration error occurred
DIDx (x=0-11)	x+1	rw	**Data inconsistency occurred for channel x in MTM mode** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged.
0	31:13	r	**Reserved** Read as zero, shall be written as zero

28.6.5.4 Register BRC_IRQ_EN

BRC Interrupt Enable Register

BRC_IRQ_EN
BRC Interrupt Enable Register **(000464_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			DID_IRQ_EN11	DID_IRQ_EN10	DID_IRQ_EN9	DID_IRQ_EN8	DID_IRQ_EN7	DID_IRQ_EN6	DID_IRQ_EN5	DID_IRQ_EN4	DID_IRQ_EN3	DID_IRQ_EN2	DID_IRQ_EN1	DID_IRQ_EN0	DEST_ERR_IRQ_EN
r			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
DEST_ERR_IRQ_EN	0	rw	**BRC_DEST_ERR_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
DID_IRQ_ENx (x=0-11)	x+1	rw	**Enable DID interrupt for channel x** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
0	31:13	r	**Reserved** Read as zero, shall be written as zero

28.6.5.5 Register BRC_IRQ_FORCINT

BRC Force Interrupt Register

BRC_IRQ_FORCINT
BRC Force Interrupt Register **(000468$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			TRG_DID11	TRG_DID10	TRG_DID9	TRG_DID8	TRG_DID7	TRG_DID6	TRG_DID5	TRG_DID4	TRG_DID3	TRG_DID2	TRG_DID1	TRG_DID0	TRG_DEST_ERR
r			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TRG_DEST_ERR	0	rw	**Trigger destination error interrupt** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B Corresponding bit in status register will not be forced 1_B Assert corresponding field in BRC_IRQ_NOTIFY register
TRG_DIDx (x=0-11)	x+1	rw	**Trigger DID interrupt for channel x** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL.
0	31:13	r	**Reserved** Read as zero, shall be written as zero

28.6.5.6 Register BRC_IRQ_MODE

BRC Interrupt Mode Configuration Register

BRC_IRQ_MODE
BRC Interrupt Mode Configuration Register **(00046C$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														IRQ_MODE	
r														rw	

Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** Note: The interrupt modes are described in **Section 28.4.5**. 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero

28.6.5.7 Register BRC_EIRQ_EN

BRC Error Interrupt Enable Register

BRC_EIRQ_EN
BRC Error Interrupt Enable Register **(000474_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**DID_EIRQ_EN11**	**DID_EIRQ_EN10**	**DID_EIRQ_EN9**	**DID_EIRQ_EN8**	**DID_EIRQ_EN7**	**DID_EIRQ_EN6**	**DID_EIRQ_EN5**	**DID_EIRQ_EN4**	**DID_EIRQ_EN3**	**DID_EIRQ_EN2**	**DID_EIRQ_EN1**	**DID_EIRQ_EN0**	**DEST_ERR_EIRQ_EN**
r			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

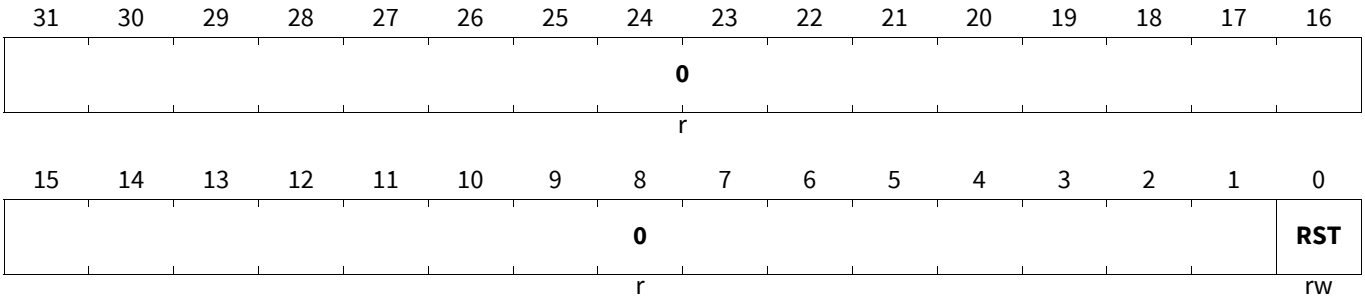
Field	Bits	Type	Description
DEST_ERR_EIRQ_EN	0	rw	**BRC_DEST_ERR_EIRQ error interrupt enable** 0_B Disable error interrupt, error interrupt is not visible outside GTM 1_B Enable error interrupt, error interrupt is visible outside GTM
DID_EIRQ_ENx (x=0-11)	x+1	rw	**Enable DID interrupt for channel x** 0_B Disable error interrupt, error interrupt is not visible outside GTM 1_B Enable error interrupt, error interrupt is visible outside GTM
0	31:13	r	**Reserved** Read as zero, shall be written as zero.

28.6.5.8 Register BRC_RST

BRC Software Reset Register

BRC_RST

BRC Software Reset Register **(000470_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RST	0	rw	**Software reset** This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately. 0_B No action 1_B Reset BRC
0	31:1	r	**Reserved** Read as zero, shall be written as zero

28.7 First In First Out Module (FIFO)

28.7.1 Overview

The FIFO unit is the storage part of the FIFO sub-module. The F2A described in (FIFO to ARU Unit) and the AFD described in chapter “AEI to FIFO Data Interface” implement the interface part of the FIFO sub-module to the ARU and the AEI bus. Each FIFO unit embeds eight logical FIFOs. These logical FIFOs are configurable in the following manner:

- FIFO size (defines start and end address)
- FIFO operation modes (normal mode or ring buffer operation mode)
- Fill level control / memory region read protection

Each logical FIFO represents a data stream between the sub-modules of the GTM and the microcontroller connected to AFD sub-module (see chapter AEI to FIFO Data Interface). The FIFO RAM counts 1K words, where the word size is 29 bit. This gives the freedom to program or receive 24 bit of data together with the five control bits inside an ARU data word.

The FIFO unit provides three ports for accessing its content. One port is connected to the F2A interface, one port is connected to the AFD interface and one port has its own AEI bus interface.

The AFD interface has always the highest priority. Accesses to the FIFO from AFD interface and direct AEI interface in parallel - which means at the same time - is not possible, because both interfaces are driven from the same AEI bus interface of the GTM.

The priority between F2A and direct AEI interface can be defined by software. This can be done by using the register **FIFO[i]_CH[x]_CTRL** for all FIFO channels of the sub-module.

The FIFO is organized as a single RAM that is also accessible through the FIFO AEI interface connected to one of the FIFO ports. To provide the direct RAM access, the RAM is mapped into the address space of the microcontroller. The addresses for accessing the RAM via AEI can be found in [1].

After reset, the FIFO RAM isn't initialized by hardware.

The FIFO channels can be flushed individually. Each of the eight FIFO channels can be used whether in normal FIFO operation mode or in ring buffer operation mode.

Beside the possibility of flushing each FIFO channel directly, a write access to FIFO[i]_CH[x]_END_ADDR or to FIFO[i]_CH[x]_START_ADDR will also flush the regarding channel which means that the read and write pointer and also the fill level of the regarding channel will be reset. In consequence of this existing data in the concerned FIFO channel are not longer valid- thereafter the channel is empty.

28.7.2 Operation Modes

28.7.2.1 FIFO Operation Mode

In normal FIFO operation mode the content of the FIFO is written and read in first-in first-out order, where the data is destroyed after it is delivered to the system bus or the F2A sub-module (see chapter “FIFO to ARU Unit ”).

The upper and lower watermark registers (registers **FIFO[i]_CH[x]_UPPER_WM** and **FIFO[i]_CH[x]_LOWER_WM**) are used for controlling the FIFO’s fill level. If the fill level falls below the lower watermark or it exceeds the upper watermark, an interrupt signal is triggered by the FIFO sub-module if enabled inside the **FIFO[i]_IRQ_EN**.

The interrupt signals are sent to the Interrupt Concentrator Module (ICM) (see chapter “Interrupt Concentrator Module”). The ICM can also initiate specific DMA transfers.

28.7.2.2 Ring Buffer Operation Mode

The ring buffer mode can be used to provide a continuous data or configuration stream to the other GTM sub-modules without CPU interaction. In ring buffer mode the FIFO provides a continuous data stream to the F2A sub-module. The first word of the FIFO is delivered first and after the last word is provided by the FIFO to the ARU, the first word can be obtained again.

If in ring buffer mode the read pointer reaches the write pointer it will be set again to the configured start address. So the read pointer always rotates cyclic between the configured start address of the regarding FIFO channel (first written data) and the write pointer which points to the last written data of the channel.

It is possible to add data to the FIFO channel via the AEI to FIFO interface (AFD) using the register AFD[i]_CH[x]_BUF_ACC while running in ring buffer mode. The new written data will be added in the next ring buffer cycle.

However, the register AFD[i]_CH[x]_BUF_ACC should not be read in read buffer mode.

It is recommended to fill the FIFO channel first before enabling the data stream in the FIFO to ARU interface (F2A).

Modifications of the continuous data stream can be achieved by using direct memory access which is provided by the FIFO AEI interface.

28.7.2.3 DMA Hysteresis Mode

The DMA hysteresis mode can be enabled by setting bit DMA_HYSTERESIS=1 in the **FIFO[i]_CH[x]_IRQ_MODE** register.

In the DMA hysteresis mode the lower and upper watermark will be masked to generate the DMA request (=*fifo_irq*) in the following manner.

If a DMA is writing data to a FIFO (configured by setting bit DMA_HYST_DIR=1 in register **FIFO[i]_CH[x]_IRQ_MODE)**, the DMA request will be generated by the lower watermark. The upper watermark does not generate a DMA request. The next DMA request will be generated by the next lower watermark until the upper watermark was reached.

If a DMA is reading data from a FIFO (configured by setting bit DMA_HYST_DIR=0 in register **FIFO[i]_CH[x]_IRQ_MODE)**, the DMA request will be generated by the upper watermark. The lower watermark does not generate a DMA request. The next DMA request will be generated by the next upper watermark until the lower watermark was reached.

Note that the watermarks have to achieve the following condition depending on the irq mode

- IRQ_MODE = Level / Pulse / Pulse-Notify mode:
  - Upper watermark > Lower watermark
- IRQ_MODE = Single-Pulse mode:
  - Upper watermark > Lower watermark + 1

28.7.3 FIFO Interrupt

Table 17 FIFO Interrupt Signals

Signal	Description
FIFO[i]_CH[x]_EMPTY	Indicating empty FIFO x (x:0…7) was reached
FIFO[i]_CH[x]_FULL	Indicating full FIFO x (x:0…7) was reached

Table 17 FIFO Interrupt Signals (cont'd)

Signal	Description
FIFO[i]_CH[x]_LOWER_WM	Indicating FIFO x (x:0…7) reached lower watermark.
FIFO[i]_CH[x]_UPPER_WM	Indicating FIFO x (x:0…7) reached upper watermark.

28.7.4 FIFO Configuration Register Overview

Table 18 FIFO Configuration Register Overview

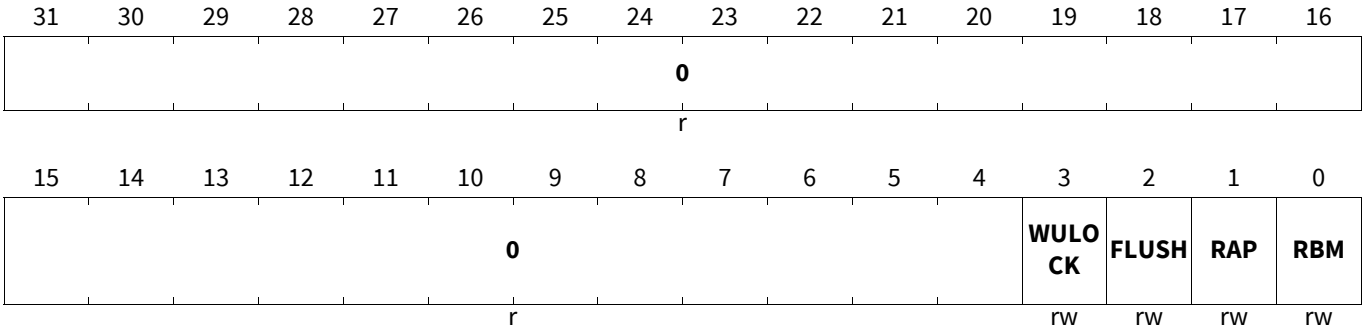
Register Name	Description	see Page
FIFO[i]_CH[z]_CTR	FIFOi channel z control register	84
FIFO[i]_CH[z]_END_ADDR	FIFOi channel z end address register	85
FIFO[i]_CH[z]_START_ADDR	FIFOi channel z start address register	85
FIFO[i]_CH[z]_UPPER_WM	FIFOi channel z upper watermark register	86
FIFO[i]_CH[z]_LOWER_WM	FIFOi channel z lower watermark register	86
FIFO[i]_CH[z]_STATUS	FIFOi channel z status register	87
FIFO[i]_CH[z]_FILL_LEVEL	FIFOi channel z fill level register	88
FIFO[i]_CH[z]_WR_PTR	FIFOi channel z write pointer register	88
FIFO[i]_CH[z]_RD_PTR	FIFOi channel z read pointer register	89
FIFO[i]_CH[z]_IRQ_NOTIFY	FIFOi channel z interrupt notification register	90
FIFO[i]_CH[z]_IRQ_EN	FIFOi channel z interrupt enable register	91
FIFO[i]_CH[z]_EIRQ_EN	FIFOi channel z error interrupt enable register	94
FIFO[i]_CH[z]_IRQ_FORCINT	FIFOi channel z force interrupt register	92
FIFO[i]_CH[z]_IRQ_MODE	FIFOi channel z interrupt mode control register	93

28.7.5 FIFO Configuration Registers Description

28.7.5.1 Register FIFO[i]_CH[z]_CTRL

FIFOi Channel z Control Register

FIFOi_CHz_CTRL (i=0-2;z=0-7)
FIFOi Channel z Control Register **(018400_H+i*4000_H+z*40_H)** **Application Reset Value: 0000 0000_H**



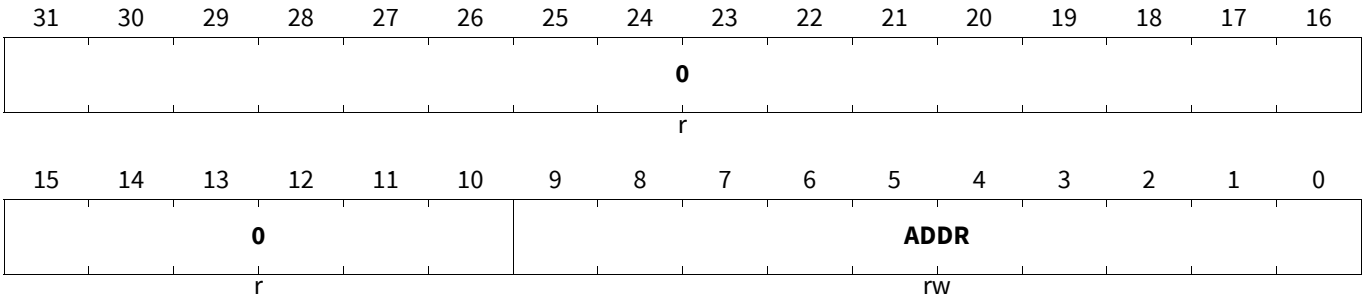
Field	Bits	Type	Description
RBM	0	rw	**Ring buffer mode enable** 0_B Normal FIFO operation mode 1_B Ring buffer mode
RAP	1	rw	**RAM access priority** RAP bit is only functional in register FIFO_0_CTRL. The priority is defined for all FIFO channels there. 0_B FIFO ports have higher access priority than AEI-IF 1_B AEI-IF has higher access priority than FIFO ports
FLUSH	2	rw	**FIFO Flush control** A FIFO Flush operation resets the **FIFO[i]_CH[z]_FILL_LEVEL**, **FIFO[i]_CH[z]_WR_PTR** and **FIFO[i]_CH[z]_RD_PTR** registers to their initial values. 0_B Normal operation 1_B Execute FIFO flush (bit is automatically cleared after flush)
WULOCK	3	rw	**RAM write unlock** Enable/disable direct RAM write access to the memory mapped FIFO region. Only the bit WULOCK of register FIFO[i]_CH0_CTRL enables/disables the direct RAM write access for all FIFO channel (whole FIFO RAM). The WULOCK bits of the other channels are writeable but have no effect. 0_B Direct RAM write access disabled 1_B Direct RAM write access enabled
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.7.5.2 Register FIFO[i]_CH[z]_END_ADDR

FIFOi Channel z End Address Register

FIFOi_CHz_END_ADDR (i=0-2;z=0-7)

FIFOi Channel z End Address Register (018404_H+i*4000_H+z*40_H) **Reset Value: Table 19**



Field	Bits	Type	Description
ADDR	9:0	rw	**End address for FIFO channel z** value for ADDR is calculated as ADDR = 128*(x+1)-1 A write access will flush the regarding channel
0	31:10	r	**Reserved** Read as zero, shall be written as zero.

Table 19 Reset Values of FIFOi_CHz_END_ADDR (i=0-2;z=0-7)

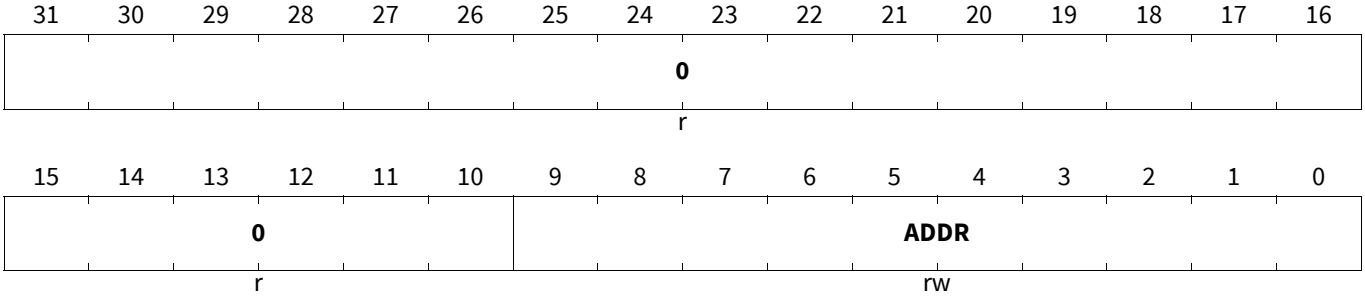
Reset Type	Reset Value	Note
Application Reset	0x80 * (z + 1) - 1	

28.7.5.3 Register FIFO[i]_CH[z]_START_ADDR

FIFOi Channel z Start Address Register

FIFOi_CHz_START_ADDR (i=0-2;z=0-7)

FIFOi Channel z Start Address Register (018408_H+i*4000_H+z*40_H) **Reset Value: Table 20**



Field	Bits	Type	Description
ADDR	9:0	rw	**Start address for FIFO channel z** Initial value for ADDR is calculated as ADDR = 128*z. A write access will flush the regarding channel.

Field	Bits	Type	Description
0	31:10	r	**Reserved** Read as zero, shall be written as zero.

Table 20 Reset Values of FIFOi_CHz_START_ADDR (i=0-2;z=0-7)

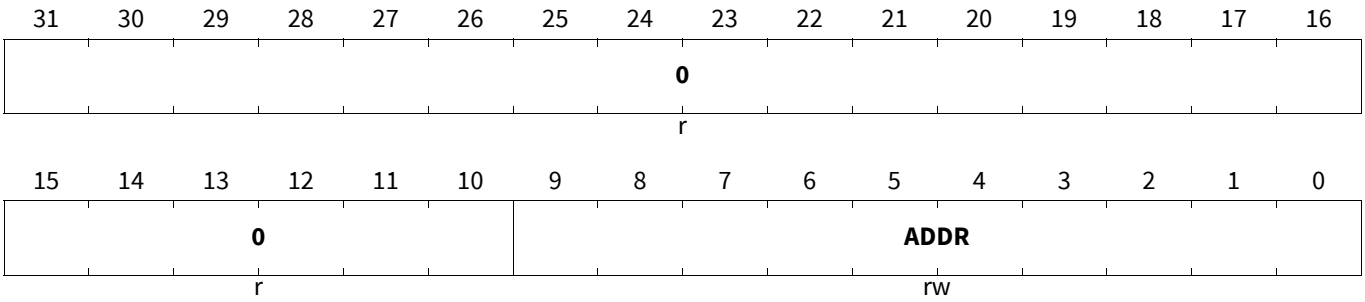
Reset Type	Reset Value	Note
Application Reset	0x80 * z	

28.7.5.4 Register FIFO[i]_CH[z]_UPPER_WM

FIFOi Channel z Upper Watermark Register

FIFOi_CHz_UPPER_WM (i=0-2;z=0-7)

FIFOi Channel z Upper Watermark Register($01840C_H$+i*4000_H+z*40_H) Application Reset Value: 0000 0060_H



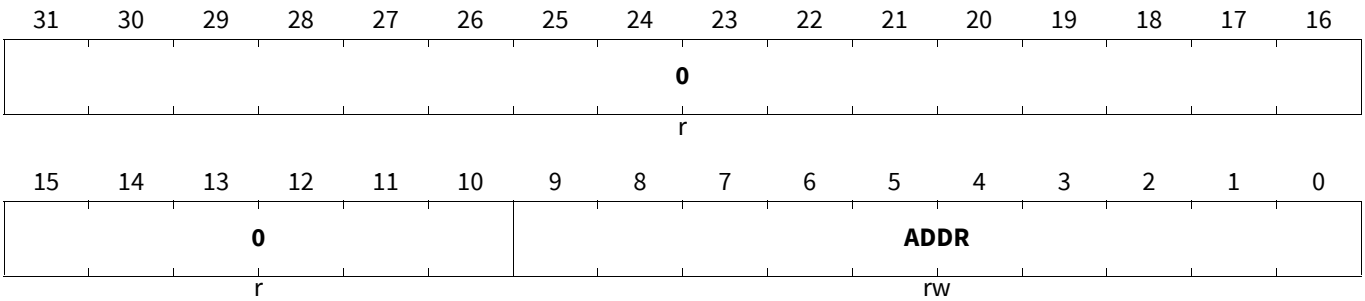
Field	Bits	Type	Description
ADDR	9:0	rw	**Upper watermark address for channel z** The upper watermark is configured as a relative fill level of the FIFO. ADDR must be in range: 0 <= ADDR <= **FIFO[i]_CH[x]_END_ADDR** - **FIFO[i]_CH[x]_START_ADDR**. Initial value for ADDR is defined as ADDR = 0x60.
0	31:10	r	**Reserved** Read as zero, shall be written as zero.

28.7.5.5 Register FIFO[i]_CH[z]_LOWER_WM

FIFOi Channel z Lower Watermark Register

FIFOi_CHz_LOWER_WM (i=0-2;z=0-7)

FIFOi Channel z Lower Watermark Register(018410_H+i*4000_H+z*40_H) Application Reset Value: 0000 0020_H



Field	Bits	Type	Description
ADDR	9:0	rw	**Lower watermark address for channel z** The lower watermark is configured as a relative fill level of the FIFO. ADDR must be in range: 0 <= ADDR <= **FIFO[i]_CH[z]_END_ADDR** - **FIFO[i]_CH[z]_START_ADDR**. Initial value for ADDR is defined as ADDR = 0x20.
0	31:10	r	**Reserved** Read as zero, shall be written as zero.

28.7.5.6 Register FIFO[i]_CH[z]_STATUS

FIFOi Channel z Status Register

FIFOi_CHz_STATUS (i=0-2;z=0-7)
FIFOi Channel z Status Register **(018414_H+i*4000_H+z*40_H)** **Application Reset Value: 0000 0005_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												UP_WM	LOW_WM	FULL	EMPTY
r												r	r	r	r

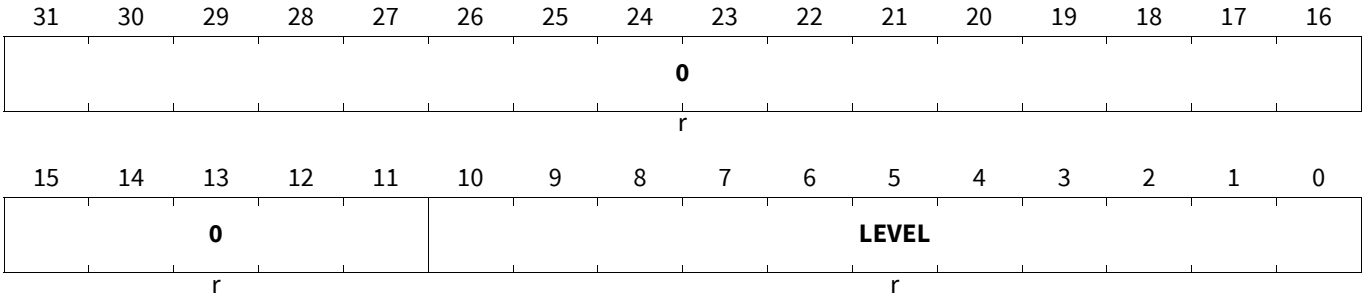
Field	Bits	Type	Description
EMPTY	0	r	**FIFO is empty** Bit only applicable in normal mode. 0_B Fill level greater then 0 1_B Fill level = 0
FULL	1	r	**FIFO is full** Bit only applicable in normal mode. 0_B Fill level less than FIFO[i]_CH[z]_END_ADDR − FIFO[i]_CH[z]_START_ADDR + 1 1_B Fill level = FIFO[i]_CH[z]_END_ADDR − FIFO[i]_CH[z]_START_ADDR + 1
LOW_WM	2	r	**Lower watermark reached** Bit only applicable in normal mode. 0_B Fill level greater than lower watermark 1_B Fill level less than or equal to lower watermark
UP_WM	3	r	**Upper watermark reached** Bit only applicable in normal mode. 0_B Fill level less than upper watermark 1_B Fill level greater than or equal to upper watermark

Field	Bits	Type	Description
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.7.5.7 Register FIFO[i]_CH[z]_FILL_LEVEL

FIFOi Channel z Fill Level Register

FIFOi_CHz_FILL_LEVEL (i=0-2;z=0-7)
FIFOi Channel z Fill Level Register (018418_H+i*4000_H+z*40_H) Application Reset Value: 0000 0000_H

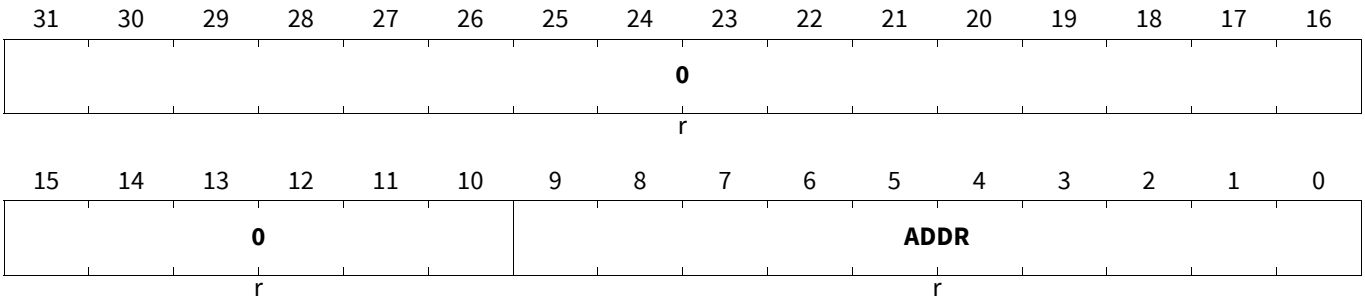


Field	Bits	Type	Description
LEVEL	10:0	r	**Fill level of the current FIFO** LEVEL is in range: 0 ≤ LEVEL ≤ FIFO[i]_CH[z]_END_ADDR − FIFO[i]_CH[z]_START_ADDR + 1 Register content is compared to the upper and lower watermark values for this channel to detect watermark over- and underflow.
0	31:11	r	**Reserved** Read as zero, shall be written as zero.

28.7.5.8 Register FIFO[i]_CH[z]_WR_PTR

FIFOi Channel z Write Pointer Register

FIFOi_CHz_WR_PTR (i=0-2;z=0-7)
FIFOi Channel z Write Pointer Register ($01841C_H$+i*4000_H+z*40_H) Reset Value: Table 21



Field	Bits	Type	Description
ADDR	9:0	r	**Position of the write pointer** ADDR must be in range 0 <= ADDR <= 1023. Initial value for ADDR is defined as ADDR = **FIFO[i]_CH[z]_START_ADDR**
0	31:10	r	**Reserved** Read as zero, shall be written as zero.

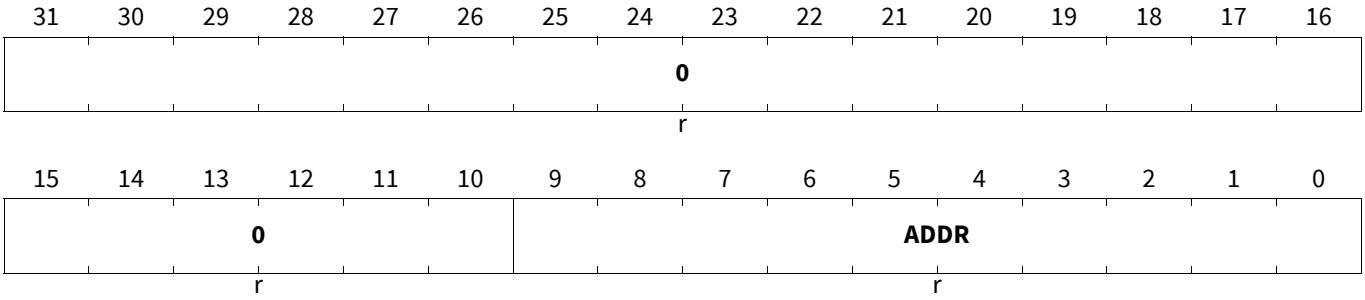
Table 21 Reset Values of FIFOi_CHz_WR_PTR (i=0-2;z=0-7)

Reset Type	Reset Value	Note
Application Reset	0x80 * z	

28.7.5.9 Register FIFO[i]_CH[z]_RD_PTR

FIFOi Channel z Read Pointer Register

FIFOi_CHz_RD_PTR (i=0-2;z=0-7)
FIFOi Channel z Read Pointer Register (018420_H+i*4000_H+z*40_H) **Reset Value: Table 22**



Field	Bits	Type	Description
ADDR	9:0	r	**Position of the read pointer** ADDR must be in range 0 ≤ ADDR ≤ 1023. Initial value for ADDR is defined as ADDR = FIFO[i]_CH[z]_START_ADDR
0	31:10	r	**Reserved** Read as zero, shall be written as zero.

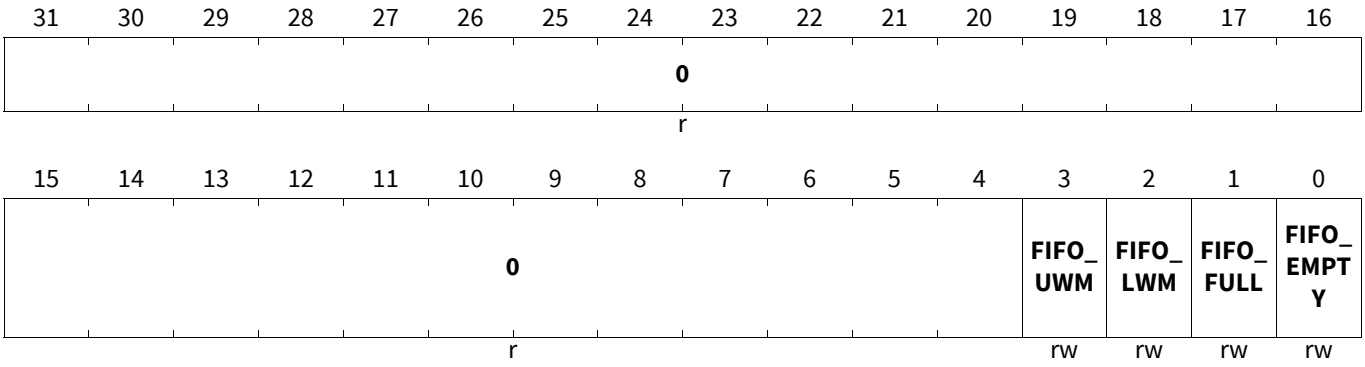
Table 22 Reset Values of FIFOi_CHz_RD_PTR (i=0-2;z=0-7)

Reset Type	Reset Value	Note
Application Reset	0x80 * z	

28.7.5.10 Register FIFO[i]_CH[z]_IRQ_NOTIFY

FIFOi Channel z Interrupt Notification Register

FIFOi_CHz_IRQ_NOTIFY (i=0-2;z=0-7)
FIFOi Channel z Interrupt Notification Register(018424_H+i*4000_H+z*40_H) Application Reset Value: $0000\ 0005_H$



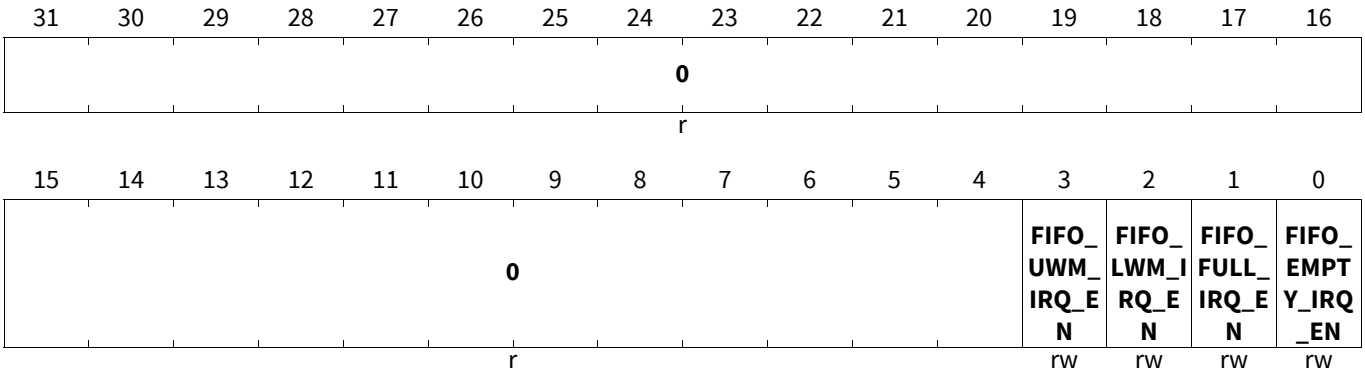
Field	Bits	Type	Description
FIFO_EMPTY	0	rw	**FIFO is empty** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B FIFO Empty interrupt occurred
FIFO_FULL	1	rw	**FIFO is full** See bit 0.
FIFO_LWM	2	rw	**FIFO lower watermark was under-run** See bit 0.
FIFO_UWM	3	rw	**FIFO upper watermark was over-run** See bit 0.
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.7.5.11 Register FIFO[i]_CH[z]_IRQ_EN

FIFOi Channel z Interrupt Enable Register

FIFOi_CHz_IRQ_EN (i=0-2;z=0-7)

FIFOi Channel z Interrupt Enable Register(018428_H+i*4000_H+z*40_H) Application Reset Value: 0000 0000_H



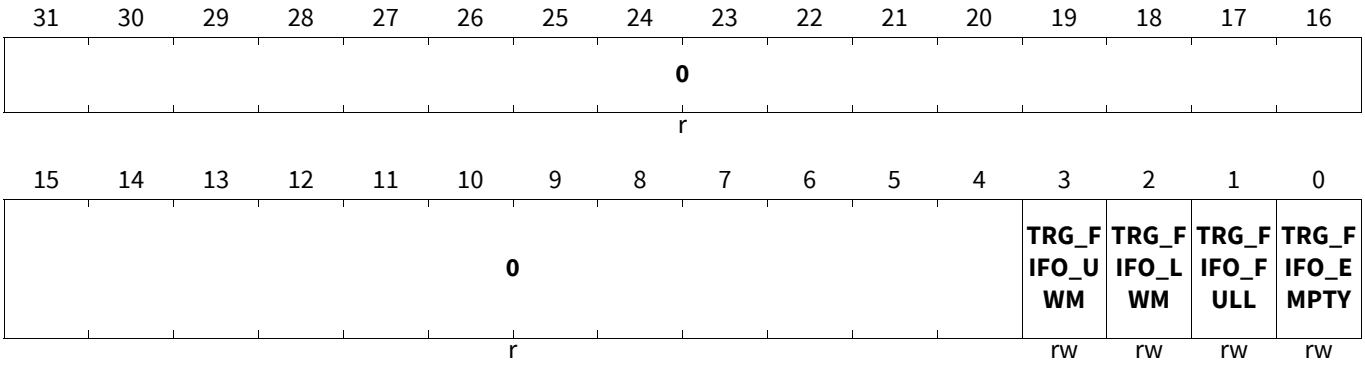
Field	Bits	Type	Description
FIFO_EMPTY_IRQ_EN	0	rw	**FIFO Empty interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
FIFO_FULL_IRQ_EN	1	rw	**FIFO Full interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
FIFO_LWM_IRQ_EN	2	rw	**FIFO Lower Watermark interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
FIFO_UWM_IRQ_EN	3	rw	**FIFO Upper Watermark interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.7.5.12 Register FIFO[i]_CH[z]_IRQ_FORCINT

FIFOi Channel z Force Interrupt Register

FIFOi_CHz_IRQ_FORCINT (i=0-2;z=0-7)

FIFOi Channel z Force Interrupt Register($01842C_H$+i*4000_H+z*40_H) **Application Reset Value: 0000 0000_H**

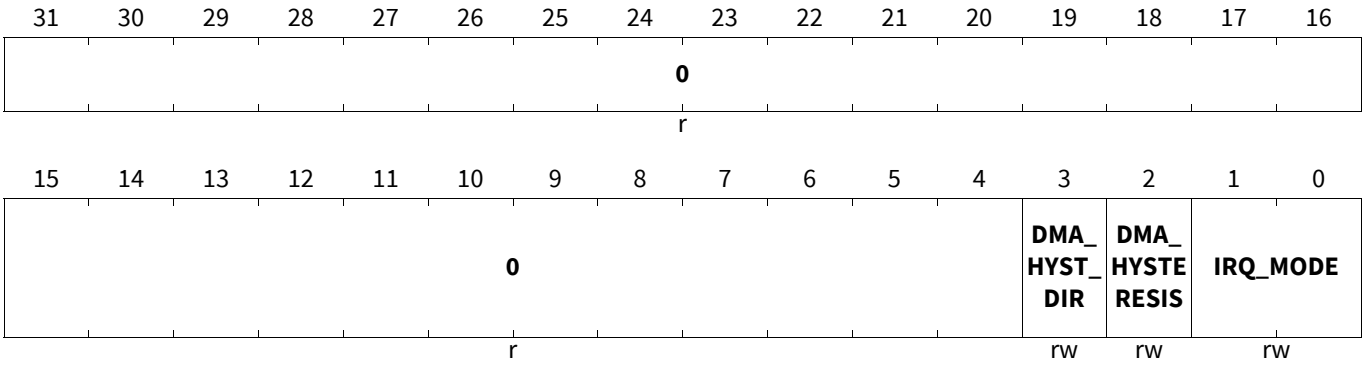


Field	Bits	Type	Description
TRG_FIFO_EMPTY	0	rw	**Force interrupt of FIFO Empty status** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL 0_B Corresponding bit in status register will not be forced 1_B Assert corresponding field in FIFO[i]_CH[z]_IRQ_NOTIFY register
TRG_FIFO_FULL	1	rw	**Force interrupt of FIFO Full status** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B Corresponding bit in status register will not be forced 1_B Assert corresponding field in FIFO[i]_CH[z]_IRQ_NOTIFY register
TRG_FIFO_LWM	2	rw	**Force interrupt of Lower Watermark** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B Corresponding bit in status register will not be forced 1_B Assert corresponding field in FIFO[i]_CH[z]_IRQ_NOTIFY register
TRG_FIFO_UWM	3	rw	**Force interrupt of Upper Watermark** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B Corresponding bit in status register will not be forced 1_B Assert corresponding field in FIFO[i]_CH[z]_IRQ_NOTIFY register
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.7.5.13 Register FIFO[i]_CH[z]_IRQ_MODE

FIFOi Channel z Interrupt Mode Control Register

FIFOi_CHz_IRQ_MODE (i=0-2;z=0-7)
FIFOi Channel z Interrupt Mode Control Register(018430_H+i*4000_H+z*40_H) Application Reset Value: $0000\ 0000_H$

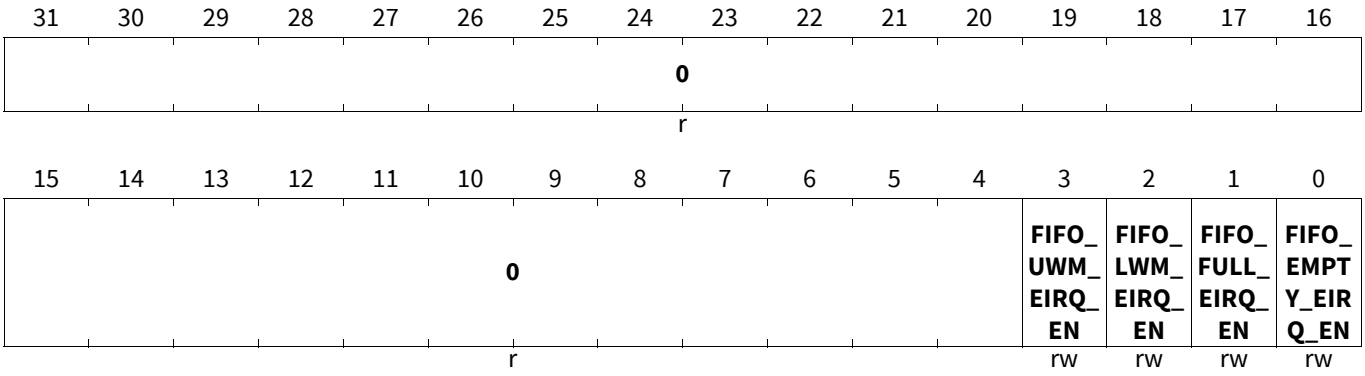


Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** The interrupt modes are described in **Section 28.4.5**. 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
DMA_HYSTERESIS	2	rw	**Enable DMA hysteresis mode** 0_B Disable FIFO hysteresis for DMA access 1_B Enable FIFO hysteresis for DMA access
DMA_HYST_DIR	3	rw	**DMA direction in hysteresis mode** In the case of DMA writing data to a FIFO, the DMA requests must be generated by the lower watermark. If the DMA hysteresis is enabled, the FIFO does not generate a new DMA request until the upper watermark is reached. In the case of DMA reading data from FIFO, the DMA requests must be generated by the upper watermark. If the DMA hysteresis is enabled, the FIFO does not generate a new DMA request until the lower watermark is reached. 0_B DMA direction read in hysteresis mode 1_B DMA direction write in hysteresis mode
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.7.5.14 Register FIFO[i]_CH[z]_EIRQ_EN

FIFOi Channel z Error Interrupt Enable Register

FIFOi_CHz_EIRQ_EN (i=0-2;z=0-7)
FIFOi Channel z Error Interrupt Enable Register(018434_H+i*4000_H+z*40_H) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
FIFO_EMPTY_EIRQ_EN	0	rw	**FIFO Empty error interrupt enable** 0_B Disable error interrupt, error interrupt is not visible outside GTM 1_B Enable error interrupt, error interrupt is visible outside GTM
FIFO_FULL_EIRQ_EN	1	rw	**FIFO Full error interrupt enable** 0_B Disable error interrupt, error interrupt is not visible outside GTM 1_B Enable error interrupt, error interrupt is visible outside GTM
FIFO_LWM_EIRQ_EN	2	rw	**FIFO Lower Watermark error interrupt enable** 0_B Disable error interrupt, error interrupt is not visible outside GTM 1_B Enable error interrupt, error interrupt is visible outside GTM
FIFO_UWM_EIRQ_EN	3	rw	**FIFO Upper Watermark error interrupt enable** 0_B Disable error interrupt, error interrupt is not visible outside GTM 1_B Enable error interrupt, error interrupt is visible outside GTM
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.8 AEI to FIFO Data Interface (AFD)

28.8.1 Overview

The AFD sub-module implements a data interface between the AEI bus and the FIFO sub-module, which consists of eight logical FIFO channels.

The AFD sub-module provides one buffer registers that are dedicated to the logical channels of the FIFO. Access to the corresponding FIFO channel is given by reading or writing this buffer registers **AFD[i]_CH[x]_BUF_ACC**.

An AEI write access to the buffer register where the corresponding FIFO channel is full will be ignored. The data will be lost.

An AEI read access to the buffer register where the corresponding FIFO channel is empty will be served with zero data.

28.8.2 AFD Register overview

Table 23 AFD Register overview

Register Name	Description	see Page
AFD[i]_CH[z]_BUF_ACC	AFD i FIFO z buffer access register	95

28.8.3 AFD Register description

28.8.3.1 Register AFD[i]_CH[z]_BUF_ACC

AFD i FIFO x Buffer Access Register

AFDi_CHx_BUF_ACC (i=0-2;x=0-7)

AFD i FIFO x Buffer Access Register **(018080_H+i*4000_H+x*10_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			DATA												
r			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															
rw															

Field	Bits	Type	Description
DATA	28:0	rw	**Read/write data from/to FIFO**
0	31:29	r	**Reserved** Read as zero, shall be written as zero.

28.9 FIFO to ARU Unit (F2A)

28.9.1 Overview

The F2A is the interface between the ARU and the FIFO sub-module. Since the data width of the ARU (ARU word) is 53 bit (two 24 bit values and five control bits) and the data width of the FIFO is only 29 bit, the F2A has to distribute the data from and to the FIFO channels in a configurable manner.

The data transfer between FIFO and ARU is organized with eight different streams that are connected to the eight different channels of the corresponding FIFO module. A stream represents a data flow from/to ARU to/from the FIFO via the F2A.

The general definition of 'channels' and 'streams' in the ARU context is done in the subparagraph “ARU routing concept”.

Each FIFO channel can act as a write stream (data flow from FIFO to ARU) or as a read stream (data flow from ARU to FIFO).

Within these streams the F2A can transmit/receive the lower, the upper or both 24 bit values of the ARU together with the ARU control bits according to the configured transfer modes as described in **Section 28.9.2**.

Each stream can be enabled/disabled separately within the register **F2A[i]_ENABLE**. If a stream will be disabled, the stream data which are stored inside the F2A will be deleted. This is necessary to ensure, that no old data are transferred after enabling a stream.

28.9.2 Transfer modes

The F2A unit provides several transfer modes to map 29 bit data of the FIFO from/to 53 bit data of the ARU. E.g. it is configurable that the 24 bit FIFO data is written to the lower ARU data entry (means bits 0 to 23) or to the higher 24 bit ARU data entry (means bits 24 to 47). Bits 24 to 28 of the FIFO data entry (the five control bits) are written/read in both cases to/from bits 48 to 52 of the ARU entry.

When both values of the ARU have to be stored in the FIFO the values are stored behind each other inside the FIFO if the FIFO is not full.

If there is only space for one 24 bit data word plus the five control bits, the F2A transfers one part of the 53 bits first and then waits for transferring the second part before new data is requested from the ARU.

When two values from the FIFO have to be written to one ARU location the words have to be located behind each other inside the FIFO.

The transfer to ARU is only established when both parts could be read out of the FIFO otherwise if only one 29 bit word was provided by the FIFO the F2A waits until the second part is available before the data is made available at the ARU.

Figure 25 shows the data ordering of the FIFO when both ARU values must be transferred between ARU and FIFO.

When reading from the ARU the F2A first writes the lower word to the FIFO.

In case of writing to the ARU the F2A reads the lower word first from the FIFO, thus the lower word must be written first to the FIFO through the AFD interface.

Please note, that the five control bits (bits 48 to 52 of the ARU data word) are duplicated as bit 24 to 28 of both FIFO words in case of reading from ARU.

In the case of writing to the ARU, bits 24 to 28 of the last written FIFO word (the higher ARU word) are copied to bits 48 to 52 of the corresponding ARU location.

The transfer modes can be configured with the **TMODE** bits of registers **F2A[i]_CH[x]_STR_CFG**.

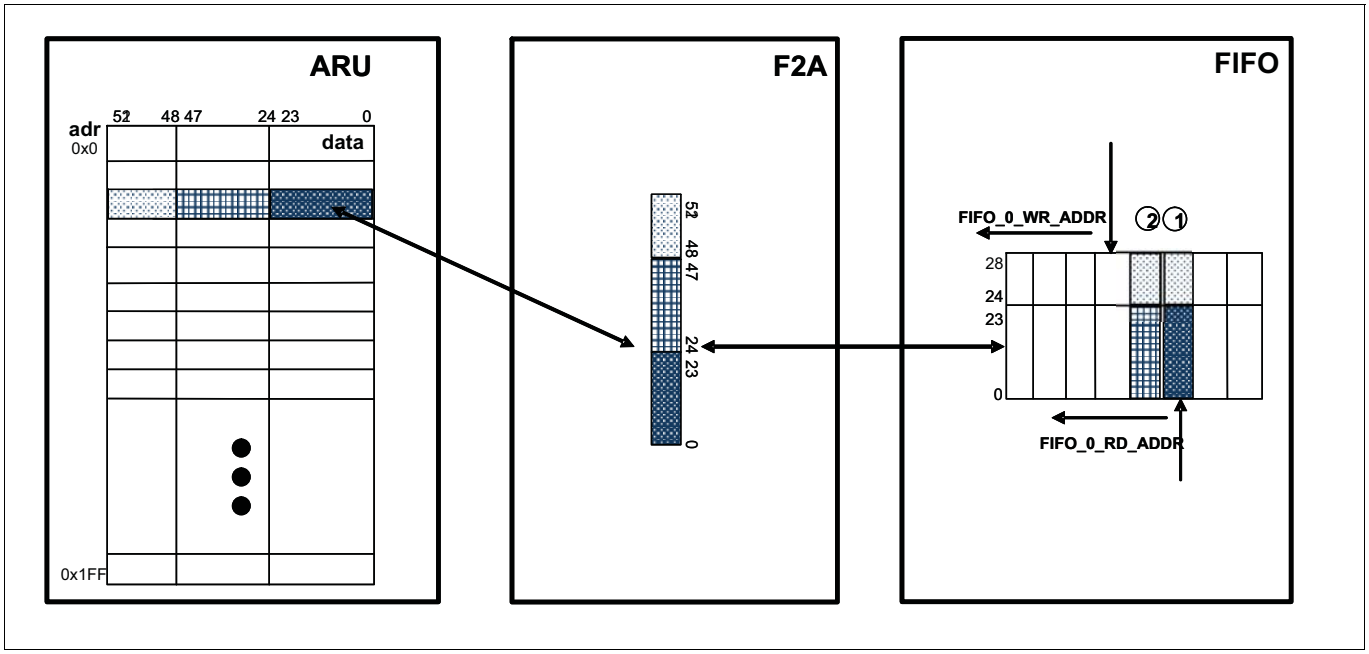


Figure 25 Data transfer of both ARU words between ARU and FIFO

28.9.3 Internal buffer mode

It is possible to use a FIFO channel as a buffer which is accessed only internally from ARU side. To do this, a read and a write stream of the F2A to one FIFO channel are needed. Therefore it is possible to reconfigure the upper 4 F2A streams 4…7 to the lower 4 FIFO channels 0…3 in the following manner:

- F2A stream 4 (+ F2A stream 0) -> FIFO channel 0
- F2A stream 5 (+ F2A stream 1) -> FIFO channel 1
- F2A stream 6 (+ F2A stream 2) -> FIFO channel 2
- F2A stream 7 (+ F2A stream 3) -> FIFO channel 3

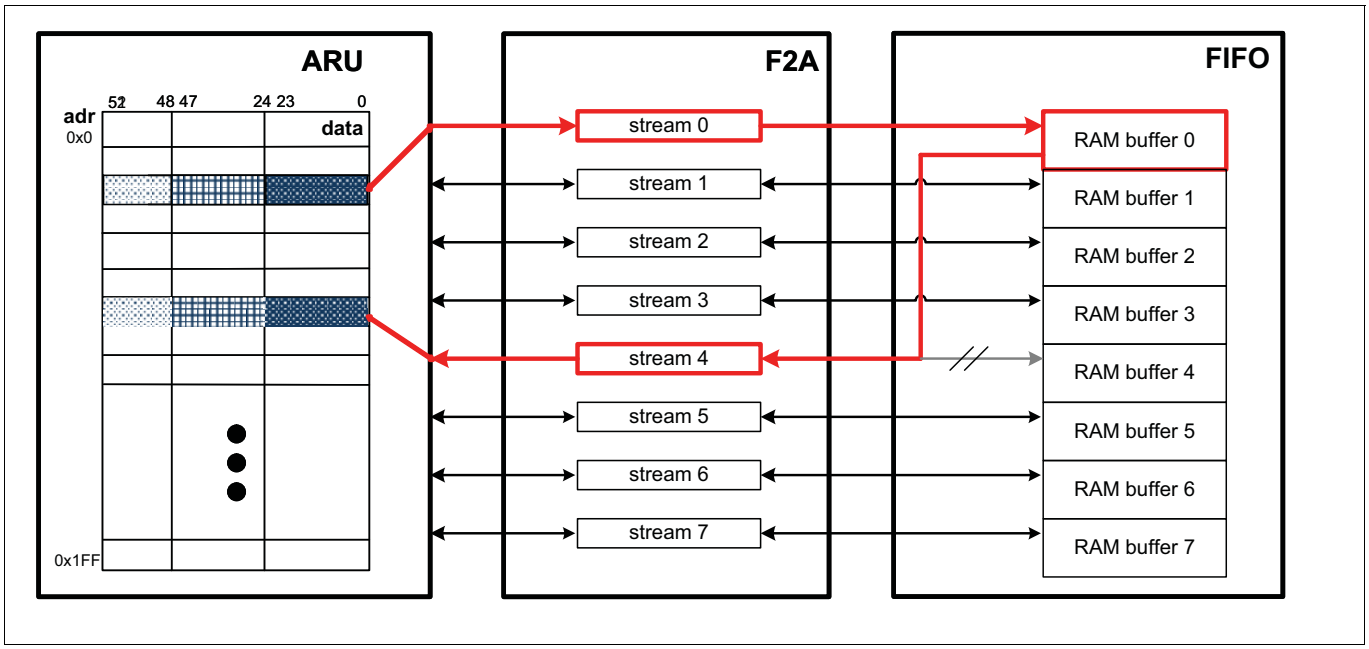


Figure 26 Re-configuration of F2A stream 4 to FIFO channel 0

The configuration of each 4 upper F2A streams can be done separately for each stream in the configuration register **F2A[i]_CTRL**.

Note that the corresponding upper FIFO channel (4…7) cannot be used in this configuration.

28.9.4 F2A Configuration Register Overview

Table 24 F2A Configuration Register Overview

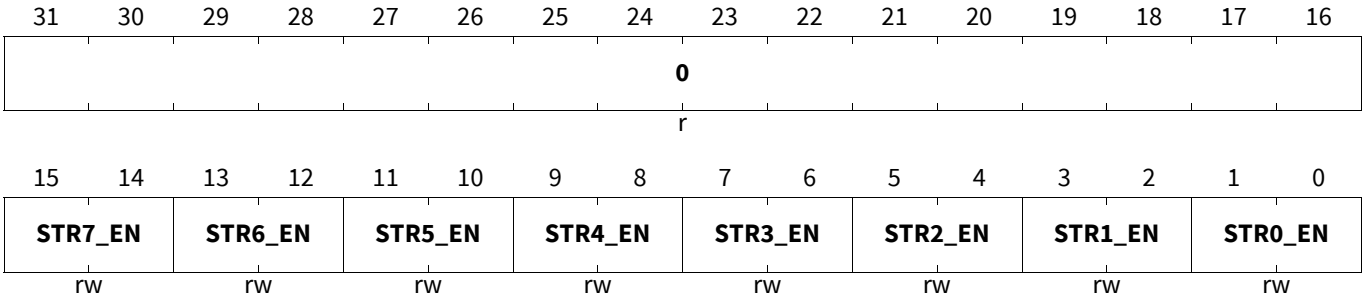
Register name	Description	see Page
F2A[i]_ENABLE	F2Ai stream activation register	99
F2A[i]_CH[z]_ARU_RD_FIFO	F2Ai channel z read address register	99
F2A[i]_CH[z]_STR_CFG	F2Ai stream z configuration register	100
F2A[i]_CTRL	F2Ai stream control register	101

28.9.5 F2A Configuration Register description

28.9.5.1 Register F2A[i]_ENABLE

F2Ai Stream Activation Register

F2Ai_ENABLE (i=0-2)
F2Ai Stream Activation Register **(018040_H+i*4000_H)** **Application Reset Value: 0000 0000_H**

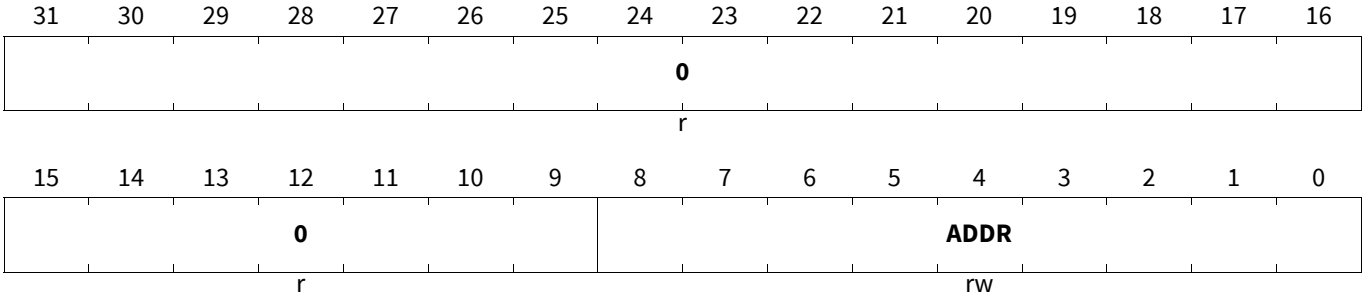


Field	Bits	Type	Description
STRx_EN (x=0-7)	2*x+1:2*x	rw	**Enable/disable stream x** Write of following double bit values is possible: 00_B Write: Don't care, bits 1:0 will not be changed / Read: Stream disabled 01_B Stream 0 is disabled and internal states are reset 10_B Stream 0 is enabled 11_B Write: Don't care, bits 1:0 will not be changed / Read: Stream enabled
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.9.5.2 Register F2A[i]_CH[z]_ARU_RD_FIFO

F2Ai Stream z Read Address Register

F2Ai_CHz_ARU_RD_FIFO (i=0-2;z=0-7)
F2Ai Stream z Read Address Register **(018000_H+i*4000_H+z*4)** **Application Reset Value: 0000 $01FE_H$**



Field	Bits	Type	Description
ADDR	8:0	rw	**ARU Read address** This bit field is only writable if channel is disabled.
0	31:9	r	**Reserved** Read as zero, shall be written as zero

28.9.5.3 Register F2A[i]_CH[z]_STR_CFG

F2Ai Stream z Configuration Register

Note: The write protected bits of register **F2A_CH[z]_STR_CFG** are only writable if the corresponding enable bit STRx_EN of register **F2A_ENABLE** is cleared.

F2Ai_CHz_STR_CFG (i=0-2;z=0-7)
F2Ai Stream z Configuration Register **(018020_H+i*4000_H+z*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													DIR	TMODE	
r													rw	rw	

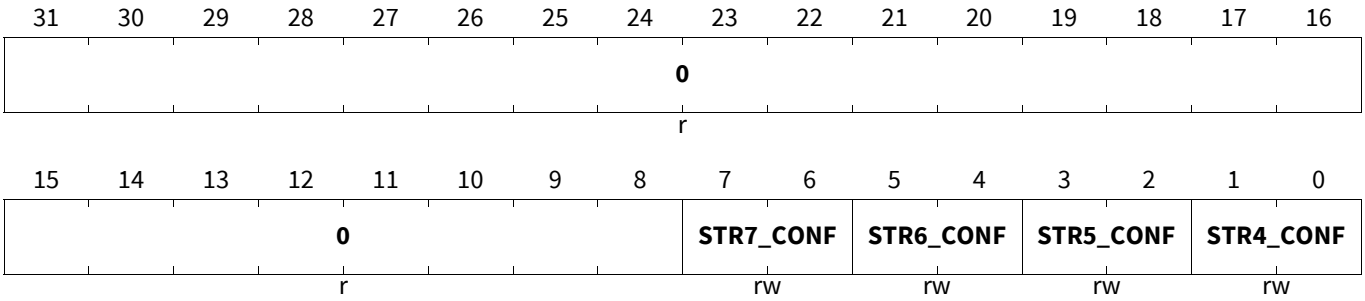
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
TMODE	17:16	rw	**Transfer mode for 53 bit ARU data from/to FIFO** 00_B Transfer low word (ARU bits 23:0) from/to FIFO 01_B Transfer high word (ARU bits 47:24) from/to FIFO 10_B Transfer both words from/to FIFO 11_B Reserved
DIR	18	rw	**Data transfer direction** 0_B Transport from ARU to FIFO 1_B Transport from FIFO to ARU
0	15:0, 31:19	r	**Reserved** Read as zero, shall be written as zero

28.9.5.4 Register F2A[i]_CTRL

F2Ai Stream Control Register

F2Ai_CTRL (i=0-2)
F2Ai Stream Control Register **(018044_H+i*4000_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
STR4_CONF	1:0	rw	**Reconfiguration of stream 4 to FIFO channel 0** Write of following double bit values is possible: The write protected bits of register **F2A[i]_CTRL** are only writable if the corresponding enable bit STR0_EN and STR4_EN of register **F2A_ENABLE** is cleared. 00_B Write: don't care, bits 1:0 will not be changed / Read: Stream 4 is mapped to FIFO buffer 4 01_B Stream 4 is mapped to FIFO buffer 4 10_B Stream 4 is mapped to FIFO buffer 0 11_B Write: don't care, bits 1:0 will not be changed / Read: Stream 4 is mapped to FIFO buffer 0
STR5_CONF	3:2	rw	**Reconfiguration of stream 5 to FIFO channel 1** Write of following double bit values is possible: The write protected bits of register **F2A[i]_CTRL** are only writable if the corresponding enable bit STR1_EN and STR5_EN of register **F2A_ENABLE** is cleared. 00_B Write: don't care, bits 1:0 will not be changed / Read: Stream 5 is mapped to FIFO buffer 5 01_B Stream 5 is mapped to FIFO buffer 5 10_B Stream 5 is mapped to FIFO buffer 1 11_B Write: don't care, bits 1:0 will not be changed / Read: Stream 5 is mapped to FIFO buffer 1

Field	Bits	Type	Description
STR6_CONF	5:4	rw	**Reconfiguration of stream 6 to FIFO channel 2** Write of following double bit values is possible: The write protected bits of register **F2A[i]_CTRL** are only writable if the corresponding enable bit STR2_EN and STR6_EN of register **F2A_ENABLE** is cleared. 00_B Write: don't care, bits 1:0 will not be changed / Read: Stream 6 is mapped to FIFO buffer 6 01_B Stream 6 is mapped to FIFO buffer 6 10_B Stream 6 s mapped to FIFO buffer 2 11_B Write: don't care, bits 1:0 will not be changed / Read: Stream 6 is mapped to FIFO buffer 2
STR7_CONF	7:6	rw	**Reconfiguration of stream 7 to FIFO channel 3** Write of following double bit values is possible: The write protected bits of register **F2A[i]_CTRL** are only writable if the corresponding enable bit STR3_EN and STR7_EN of register **F2A_ENABLE** is cleared. 00_B Write: don't care, bits 1:0 will not be changed / Read: Stream 7 is mapped to FIFO buffer 7 01_B Stream 7 is mapped to FIFO buffer 7 10_B Stream 7 s mapped to FIFO buffer 3 11_B Write: don't care, bits 1:0 will not be changed / Read: Stream 7 is mapped to FIFO buffer 3
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.10 Clock Management Unit (CMU)

28.10.1 Overview

The Clock Management Unit (CMU) is responsible for clock generation of the counters and of the GTM. The CMU consists of three sub-units that generate different clock sources for the whole GTM. The primary clock source for this sub-module is the cluster 0 clock signal cls0_clk which is defined by the value of bit field CLS0_CLK_DIV in the register GTM_CLS_CLK_CFG. **Figure 27** shows a block diagram of the CMU.

The Configurable Clock Generation (CFGU) sub-unit provides eight dedicated clock sources for the following GTM modules: TIM, ATOM, TBU, and MON. Each instance of such a module can choose an arbitrary clock source, in order to specify wide-ranging time bases.

The Fixed Clock Generation (FXU) sub-unit generates predefined non-configurable clocks *CMU_FXCLK[y]* (y: 0…4) for the TOM modules and the MON module. The *CMU_FXCLK[y]* signals are derived from the *CMU_GCLK_EN* signal generated by the Global Clock Divider. The dividing factors are defined as 2^0, 2^4, 2^8, 2^{12}, and 2^{16}.

The External Clock Generation (EGU) sub-unit is able to generate up to three chip external clock signals visible at *CMU_ECLK[z]* (z: 0…2) with a duty cycle of about 50%.

The External Clock Generation (EGU) sub-unit is able to generate clock *CMU_CLK8* for module CCM to manage 2 clock domains.

The clock source signals *CMU_CLK[x]* (x: 0…7) and *CMU_FXCLK[y]* are implemented in form of enable signals for the corresponding registers, which means that the actual clock signal of all registers always use the *CLS0_CLK* signal.

The four configurable clock signals *CMU_CLK0*, *CMU_CLK1*, *CMU_CLK6* and *CMU_CLK7* are used for the TIM filter counters.

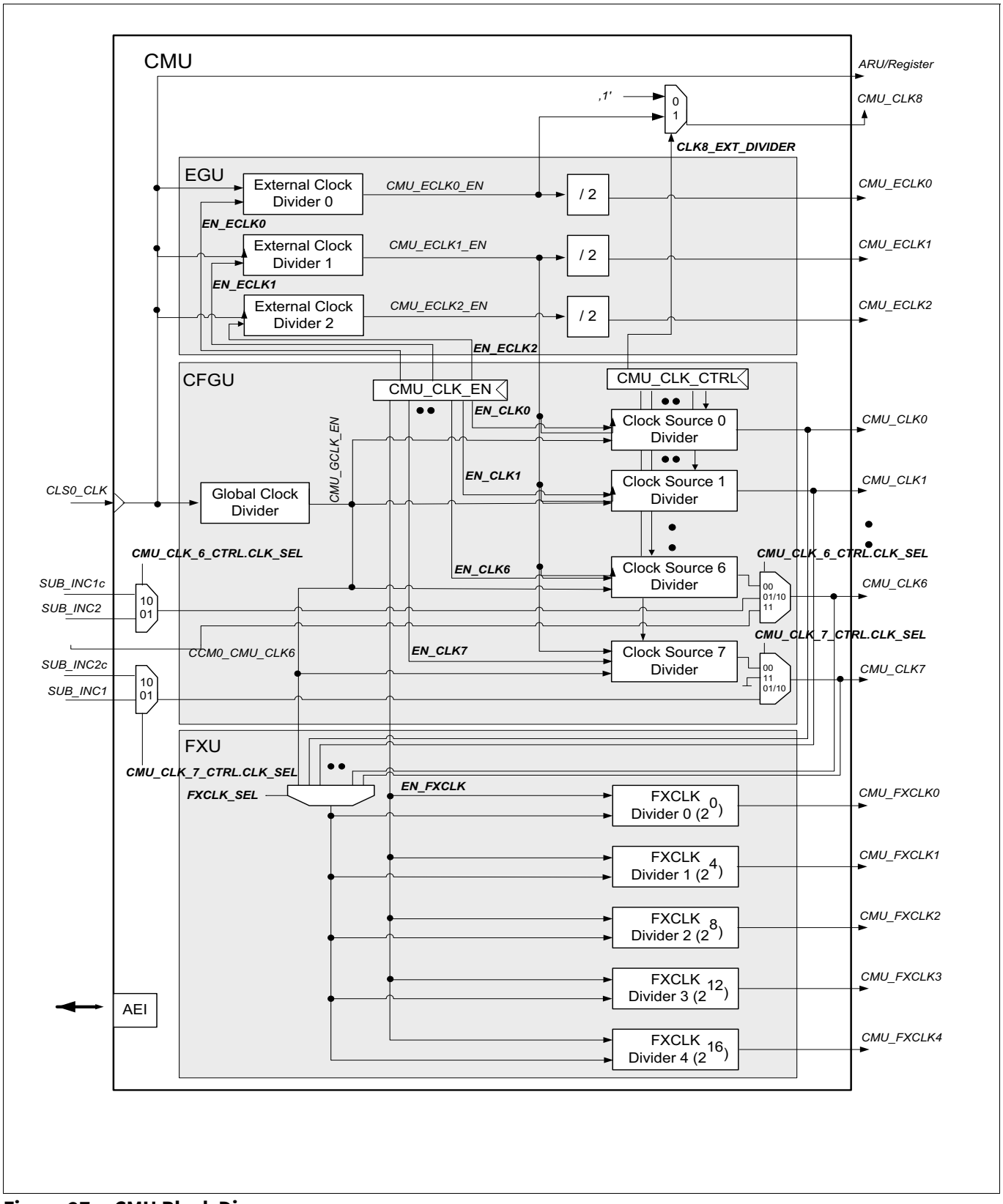


Figure 27 CMU Block Diagram

28.10.2 Global Clock Divider

The sub block Global Clock Divider can be used to divide the CMU primary source signal CLS0_CLK into a common subdivided clock signal.

The divided clock signal of the sub block Global Clock Divider is implemented as an enable signal that enables dedicated clocks from the *CLS0_CLK* signal to generate the user specified divided clock frequency.

The resulting fractional divider (Z/N) specified through equation:

$$T_{CMU\_GCLK\_EN}=(Z/N)*T_{CLS0\_CLK}$$

is implemented according the following algorithm

(*Z*: *CMU_GCLK_NUM(23:0)*; *N*: *CMU_GCLK_DEN(23:0)*; *Z,N* >0)

(1) Set remainder (*R*), operand1 (*OP1*) and operand2 (*OP2*) register during INIT-phase (with implicit conversion to signed):

R=Z, OP1=N, OP2=N-Z;

(2) After leaving INIT-phase (at least one *CMU_CLK[x]* has been enabled) the sign of remainder R for each CLS0_*CLK* cycle will be checked:

(3) If *R>0* keep updating remainder and keep *CMU_GCLK_EN='0'*:

R=R-OP1;

(4) If *R<0* update remainder and set *CMU_GCLK_EN='1'*:

R=R-OP2

After at most (*Z/N*+1) subtractions (3) there will be a negative *R* and an active phase of the generated clock enable (for one cycle) will be triggered (4). The remainder *R* is a measure for the distance to a real *Z/N* clock and will be regarded for the next generated clock enable cycle phase. The new *R* value will be *R=R+(Z-N)*. In the worst case the remainder *R* will sum up to an additional cycle in the generated clock enable period after *Z*-cycles. In the other cases equally distributed additional cycles will be inserted for the generated clock enable. If *Z* is an integer multiple of *N* no additional cycles will be included for the generated clock enable at all.

Note that for a better resource sharing all arithmetic has been reduced to subtractions and the initialization of the remainder *R* uses the complement of (*Z-N*).

28.10.3 Configurable Clock Generation sub-unit (CFGU)

The CMU sub-unit CFGU provides up to eight configurable clock divider blocks that divide the common C*MU_GCLK_EN* signal into dedicated enable signals for the GTM sub blocks.

The configuration of the eight different clock signals *CMU_CLK[x]* (x: 0…7) always depends on the configuration of the global clock enable signal *CMU_GCLK_EN*. Additionally, each clock source has its own configuration data, provided by the control register **CMU_CLK_[x]_CTRL (x=0..7)**.

According to the configuration of the Global Clock Divider, the configuration of the Clock Source x Divider is done by setting an appropriate value in the bit field **CLK_CNT[x]** of the register **CMU_CLK_[x]_CTRL**.

The frequency *fx = 1/Tx* of the corresponding clock enable signal *CMU_CLK[x]* can be determined by the unsigned representation of **CLK_CNT[x]** of the register **CMU_CLK_[x]_CTRL** in the following way:

$$T_{CMU\_CLK}[x]=(CLK\_CNT[x]+1)*T_{CMU\_GCLK\_EN}$$

The corresponding wave form is shown in **Figure 28**.

Each clock signal *CMU_CLK[x]* can be enabled individually by setting the appropriate bit field **EN_CLK[x]** in the register **CMU_CLK_EN**. Except for *CMU_CLK6* and *CMU_CLK7* individual enabling and disabling is active only if **CLK_SEL** is reset.

Alternatively, clock source six and seven (*CMU_CLK6* and *CMU_CLK7*) may provide the signal *SUB_INC1* and *SUB_INC2* coming from module DPLL as clock enable signal depending on the bit field **CLK_SEL(1:0)** of the register **CMU_CLK_6_CTRL** and on the bit field **CLK_SEL(1:0)** of the register **CMU_CLK_7_CTRL**.

CMU_CLK8 is switched by **CLK8_EXT_DIVIDER** of the register **CMU_CLK_CTRL** between *CLS0_CLK* and *CMU_ECLK0*.

To switch the clock reference CMU_GCLK_EN with *CMU_ECLK1*_EN an input selector are used in all Clock Source Divider. The *CMU_ECLK1*_EN source is enabled by setting the appropriate bit field **CMU[x]_EXT_DIVIDER** in the register **CMU_CLK_CTRL**.

To avoid unexpected behavior of the hardware, the configuration of register **CMU_CLK_[x]_CTRL** and **CMU_CLK_CTRL** can only be changed, when the corresponding clock signal *CMU_CLK[x]* and *CMU_ECLK[1]* is disabled.

Further, any changes to the registers **CMU_GCLK_NUM** and **CMU_GCLK_DEN** can only be performed, when all clock enable signals *CMU_CLK[x]* and the **EN_FXCLK** bit inside the **CMU_CLK_EN** register are disabled.

The clock source signals *CMU_CLK[x]* (x: 0…7) and *CMU_FXCLK[y]* are implemented in form of enable signals for the corresponding registers, which means that the actual clock signal of all registers always use the *CLS0_CLK* signal.

The hardware guarantees that all clock signals *CMU_CLK[x]* (x: 0…7), which were enabled simultaneous, are synchronized to each other. Simultaneous enabling does mean that the bits **EN_CLK[x]** in the register **CMU_CLK_EN** are set by the same write access.

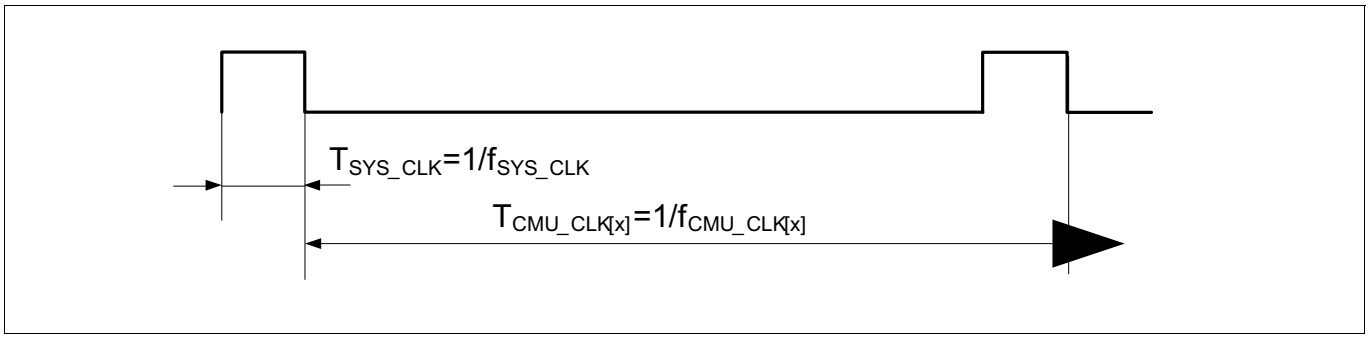


Figure 28 Wave Form of Generated Clock Signal CMU_CLK[x]

28.10.4 Fixed Clock Generation (FXU)

The FXU sub-unit generates fixed clock enables out of the *CMU_GCLK_EN* or one of the eight *CMU_CLK[x]* enable signal depending on the **FXCLK_SEL** bit field of the **CMU_FXCLK_CTRL** register. These clock enables are used for the PWM generation inside the TOM modules.

All clock enables *CMU_FXCLK[y]* can be enabled or disabled simultaneous by setting the appropriate bit field **EN_FXCLK** in the register **CMU_CLK_EN**.

The dividing factors are defined as 2^0, 2^4, 2^8, 2^{12}, and 2^{16}. The signals *CMU_FXCLK[y]* are implemented in form of enable signals for the corresponding registers (see also **Figure 28**)

28.10.5 External Generation Unit (EGU)

The EGU sub-unit generate up to three separate clock output signals *CMU_ECLK[z]* (z: 0…2).

Each of these clock signals is derived from the corresponding External Clock Divider z sub block, which generates a clock signal derived from the GTM input clock *CLS0_CLK*.

In contrast to the signals *CMU_CLK[x]* and *CMU_FXCLK[y]*, which are treated as simple enable signals for the registers, the signals *CMU_ECLK[z]* have a duty cycle of about 50% that is used as a true clock signal for external peripheral components.

To manage a second global frequency CMU_GCLK_EN could be replaced by ECLK[1]_EN for CMU_CLK[x](x:0..7). Also the all-time enabled CMU_CLK8 could be replaced by ECLK[0]_EN.

Each of the external clocks dividers are enabled and disabled by setting the appropriate bit field **EN_ECLK[z]** in the register **CMU_CLK_EN**.

The clock frequencies $f_{CMU\_ECLK[z]} = 1/T_{CMU\_ECLK[z]}$ of the external clocks are controlled with the registers **CMU_ECLK_[z]_NUM** and **CMU_ECLK_[z]_DEN** as follows:

$$T_{CMU\_ECLK}[z] = (ECLK[z]\_NUM / ECLK[z]\_DEN) * T_{CLS0\_CLK}$$

and is implemented according the following algorithm

(*Z:CMU_ECLK_[z]_NUM(23:0); N: CMU_ECLK_[z]_DEN(23:0*); *Z,N >0*; *Z>=N; CMU_ECLK[z]*='0'):

(1) Set remainder (*R*), operand1 (*OP1*) and operand2 (*OP2*) register during INIT-phase (with implicit conversion to signed): *R=Z, OP1=N, OP2=N-Z;*

(2) After leaving INIT-phase (*CMU_ECLK[z]* has been enabled) the sign of remainder R for each *CLS0_CLK* cycle will be checked:

(3) If *R>0* keep updating remainder and keep *CMU_ECLK[z]*: *R=R-OP1;*

(4) If *R<0* update remainder and toggle *CMU_ECLK[z]*: *R=R-OP2*;

After at most (*Z/N*+1) subtractions (3) there will be a negative *R* and an active phase of the generated clock enable (for one cycle) will be triggered (4). The remainder *R* is a measure for the distance to a real *Z/N* clock and will be regarded for the next generated clock toggle phase. The new *R* value will be *R=R+(Z-N)*. In the worst case the remainder *R* will sum up to an additional cycle in the generated clock toggle period after *Z*-cycles. In the other cases equally distributed additional cycles will be inserted for the generated clock toggle. If *Z* is an integer multiple of *N* no additional cycles will be included for the generated clock toggle at all. Note that for a better resource sharing all arithmetic has been reduced to subtractions and the initialization of the remainder *R* uses the complement of (*Z-N*).

The default value of the *CMU_ECLK[z]* output is low.

28.10.6 CMU Configuration Register Overview

Table 25 CMU Configuration Register Overview

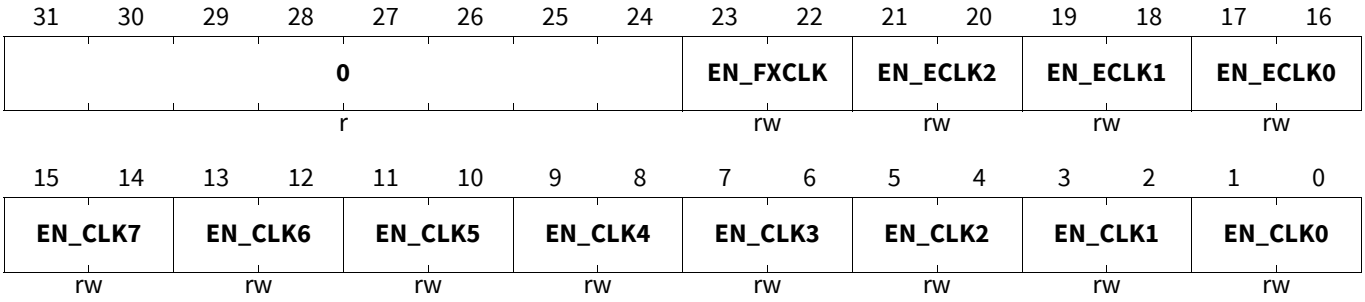
Register Name	Description	see Page
CMU_CLK_EN	CMU clock enable	107
CMU_GCLK_NUM	CMU global clock control numerator	108
CMU_GCLK_DEN	CMU global clock control denominator	108
CMU_CLK_[z]_CTRL	CMU control for clock source z	109
CMU_ECLK_[z]_NUM	CMU external clock z control numerator	110
CMU_ECLK_[z]_DEN	CMU external clock z control denominator	110
CMU_FXCLK_CTRL	CMU control FXCLK sub-unit input clock	111
CMU_GLB_CTRL	CMU synchronizing ARU and clock source	112
CMU_CLK_CTRL	CMU control for clock source selection	113

28.10.7 CMU Configuration Register Description

28.10.7.1 Register CMU_CLK_EN

CMU Clock Enable Register

CMU_CLK_EN
CMU Clock Enable Register **(000300_H)** **Application Reset Value: $0000\ 0000_H$**

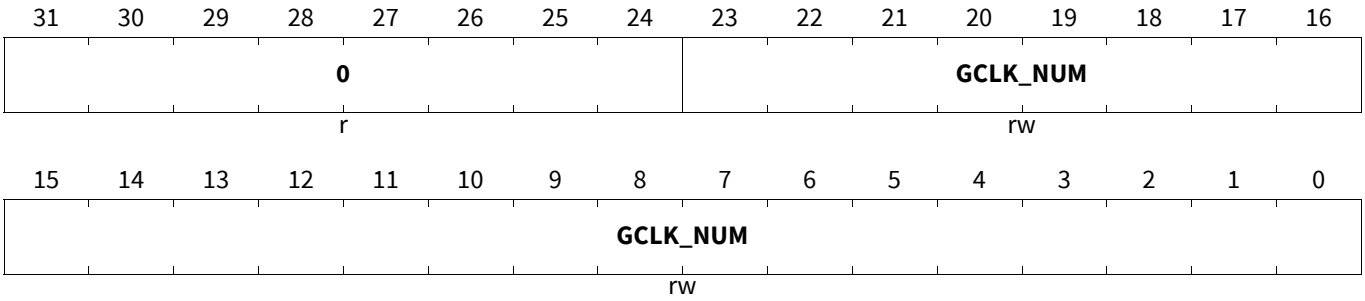


Field	Bits	Type	Description
EN_CLKx (x=0-7)	2*x+1:2*x	rw	**Enable clock source x** Any read access to an **EN_CLK[x]**, **EN_ECLK[z]** or **EN_FXCLK** bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored. Any disabling to **EN_CLK[x]** will be reset internal counters for configurable clocks. 00_B Clock source is disabled (ignore write access) 01_B Disable clock signal and reset internal states 10_B Enable clock signal 11_B Clock signal enabled (ignore write access)
EN_ECLKx (x=0-2)	2*x+17:2*x +16	rw	**Enable ECLK x generation sub-unit** Coding see bit EN_CLKx.
EN_FXCLK	23:22	rw	**Enable all CMU_FXCLK, see bits 1:0** An enable to **EN_FXCLK** from disable state will be reset internal fixed clock counters.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.10.7.2 Register CMU_GCLK_NUM

CMU Global Clock Control Numerator

CMU_GCLK_NUM
CMU Global Clock Control Numerator **(000304_H)** **Application Reset Value: $0000\ 0001_H$**



Field	Bits	Type	Description
GCLK_NUM	23:0	rw	**GCLK_NUM** Value can only be modified when all clock enables **EN_CLK[x]** and the **EN_FXCLK** are disabled. The CMU hardware alters the content of **CMU_GCLK_NUM** and **CMU_GCLK_DEN** automatically to 0x1, if **CMU_GCLK_NUM** is specified less than **CMU_GCLK_DEN** or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register **CMU_GCLK_NUM** followed by a single write to register **CMU_GCLK_DEN**.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.10.7.3 Register CMU_GCLK_DEN

CMU Global Clock Control Denominator

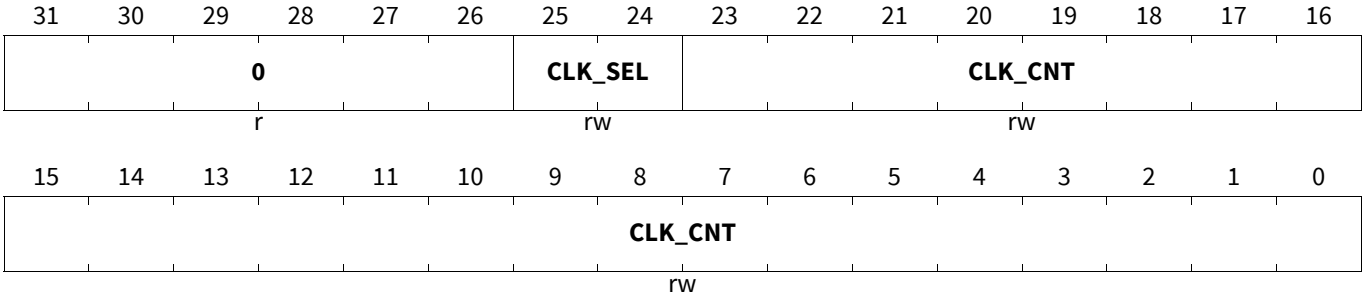
CMU_GCLK_DEN
CMU Global Clock Control Denominator **(000308_H)** **Application Reset Value: $0000\ 0001_H$**

Field	Bits	Type	Description
GCLK_DEN	23:0	rw	**GCLK_DEN** Value can only be modified when all clock enables **EN_CLK[x]** and the **EN_FXCLK** are disabled. The CMU hardware alters the content of **CMU_GCLK_NUM** and **CMU_GCLK_DEN** automatically to 0x1, if **CMU_GCLK_NUM** is specified less than **CMU_GCLK_DEN** or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register **CMU_GCLK_NUM** followed by a single write to register **CMU_GCLK_DEN**.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.10.7.4 Register CMU_CLK_[z]_CTRL

CMU Control for Clock Source z

CMU_CLK_z_CTRL (z=0-7)
CMU Control for Clock Source z **($00030C_H$+z*4)** **Application Reset Value: 0000 0000_H**

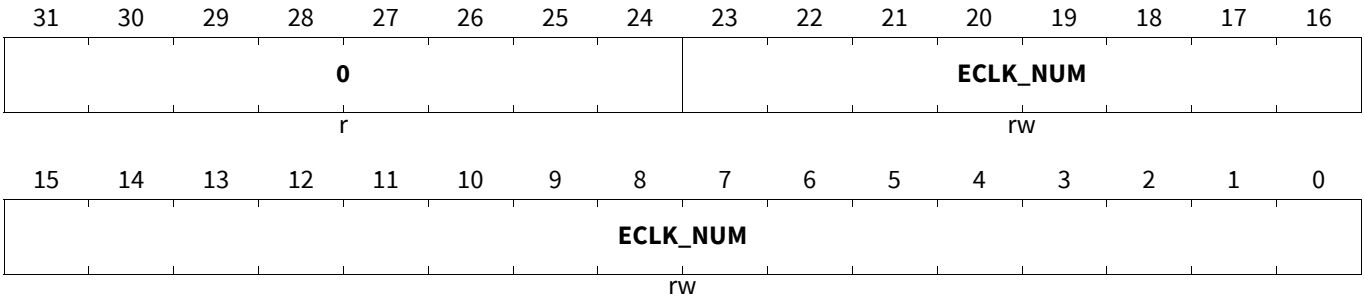


Field	Bits	Type	Description
CLK_CNT	23:0	rw	**Clock count** Defines count value for the clock divider. Value can only be modified when clock enable **EN_CLKz** and **EN_ECLK1** are disabled.
CLK_SEL	25:24	rw	**Clock source selection for CMU_CLKz** Value can only be modified when clock enable **EN_CLKz** and **EN_ECLK1** are disabled. *Note: The existence and interpretation of this bit field depends on z. z>5*
0	31:26	r	**Reserved** Read as zero, shall be written as zero.

28.10.7.5 Register CMU_ECLK_[z]_NUM

CMU External Clock z Control Numerator

CMU_ECLK_z_NUM (z=0-2)
CMU External Clock z Control Numerator (00032C$_H$+z*8) Application Reset Value: 0000 0001$_H$



Field	Bits	Type	Description
ECLK_NUM	23:0	rw	**ECLK_NUM** Numerator for external clock divider. Defines numerator of the fractional divider. *Note: Value can only be modified when clock enable **EN_ECLK[z]** disabled.* *Note: The CMU hardware alters the content of **CMU_ECLK_[z]_NUM** and **CMU_ECLK_[z]_DEN** automatically to 0x1, if **CMU_ECLK_[z]_NUM** is specified less than **CMU_ECLK_[z]_DEN** or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register **CMU_ECLK_[z]_NUM** followed by a single write to register **CMU_ECLK_[z]_DEN**.*
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.10.7.6 Register CMU_ECLK_[z]_DEN

CMU External Clock z Control Denominator

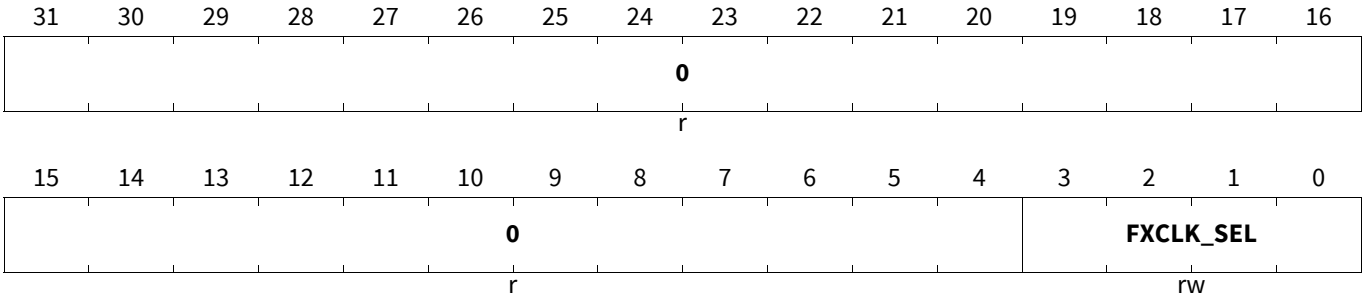
CMU_ECLK_z_DEN (z=0-2)
CMU External Clock z Control Denominator (000330$_H$+z*8) Application Reset Value: 0000 0001$_H$

Field	Bits	Type	Description
ECLK_DEN	23:0	rw	**ECLK_DEN** Denominator for external clock divider. Defines denominator of the fractional divider *Note: Value can only be modified when clock enable **EN_ECLK[z]** disabled.* *Note: The CMU hardware alters the content of **CMU_ECLK_[z]_NUM** and **CMU_ECLK_[z]_DEN** automatically to 0x1, if **CMU_ECLK_[z]_NUM** is specified less than **CMU_ECLK_[z]_DEN** or one of the values is specified with a value zero. Thus, a secure way for altering the values is writing twice to the register **CMU_ECLK_[z]_NUM** followed by a single write to register **CMU_ECLK_[z]_DEN**.*
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.10.7.7 Register CMU_FXCLK_CTRL

CMU Control FXCLK Sub-Unit Input Clock

CMU_FXCLK_CTRL
CMU Control FXCLK Sub-Unit Input Clock (000344$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
FXCLK_SEL	3:0	rw	**Input clock selection for EN_FXCLK line** This value can only be written when the CMU_FXCLK generation is disabled. See bits 23..22 in register **CMU_CLK_EN**. Other values for FXCLK_SEL are reserved and should not be used, but they behave like FXCLK_SEL = 0. 0_H CMU_GCLK_EN selected 1_H CMU_CLK0 selected 2_H CMU_CLK1 selected 3_H CMU_CLK2 selected 4_H CMU_CLK3 selected 5_H CMU_CLK4 selected 6_H CMU_CLK5 selected 7_H CMU_CLK6 selected 8_H CMU_CLK7 selected CMU_CLK7 selected
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.10.7.8 Register CMU_GLB_CTRL

CMU Synchronizing ARU and Clock Source

CMU_GLB_CTRL
CMU Synchronizing ARU and Clock Source (000348$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															ARU_ADDR_RSTGLB
r															rw

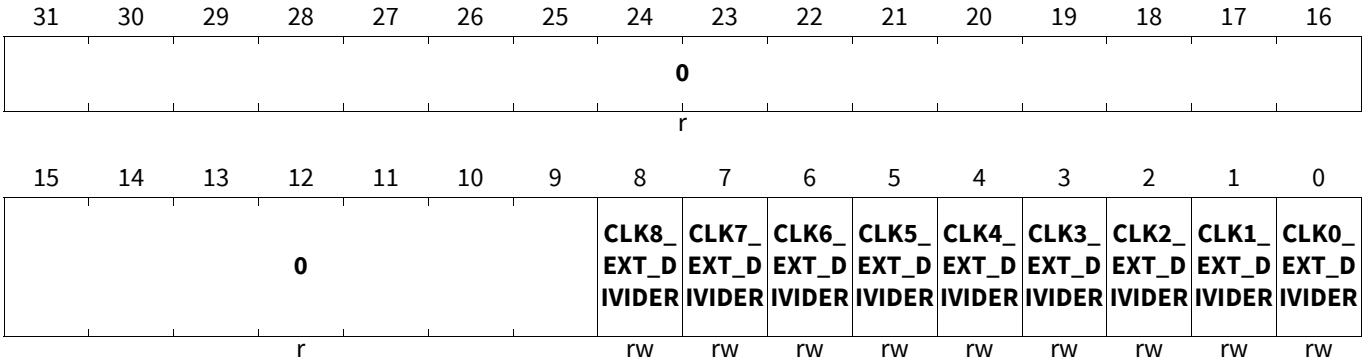
Field	Bits	Type	Description
ARU_ADDR_RSTGLB	0	rw	**Reset ARU caddr counter and ARU dynamic route counter** Writing value 1 to this bit field results in a request to reset the ARU caddr counter and ARU dynamic route counter. The next following write access to register **CMU_CLK_EN** applies the ARU caddr counter reset, ARU dynamic route counter reset and resets this bit. This feature can be used to synchronize the ARU round trip time to the CMU clocks. This bit is write protected. Before writing to this bit set bit RF_PROT of register **GTM_CTRL** to 0.

Field	Bits	Type	Description
0	31:1	r	**Reserved** Read as zero, shall be written as zero

28.10.7.9 Register CMU_CLK_CTRL

CMU Control for Clock Source Selection

CMU_CLK_CTRL
CMU Control for Clock Source Selection **(00034C$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLK0_EXT_DIVIDER	0	rw	**Clock source selection for CMU_CLK_0_CTRL** Value can only be modified when clock enable **EN_CLK0** and **EN_ECLK1** are disabled. 0_B Use Clock Source CMU_GCLK_EN 1_B Use Clock Source CMU_ECLK1
CLK1_EXT_DIVIDER	1	rw	**Clock source selection for CMU_CLK_1_CTRL** Value can only be modified when clock enable **EN_CLK1** and **EN_ECLK1** are disabled.
CLK2_EXT_DIVIDER	2	rw	**Clock source selection for CMU_CLK_2_CTRL** Value can only be modified when clock enable **EN_CLK2** and **EN_ECLK1** are disabled.
CLK3_EXT_DIVIDER	3	rw	**Clock source selection for CMU_CLK_3_CTRL** Value can only be modified when clock enable **EN_CLK3** and **EN_ECLK1** are disabled.
CLK4_EXT_DIVIDER	4	rw	**Clock source selection for CMU_CLK_4_CTRL** Value can only be modified when clock enable **EN_CLK4** and **EN_ECLK1** are disabled.
CLK5_EXT_DIVIDER	5	rw	**Clock source selection for CMU_CLK_5_CTRL** Value can only be modified when clock enable **EN_CLK5** and **EN_ECLK1** are disabled.
CLK6_EXT_DIVIDER	6	rw	**Clock source selection for CMU_CLK_6_CTRL** Value can only be modified when clock enable **EN_CLK6** and **EN_ECLK1** are disabled.

Field	Bits	Type	Description
CLK7_EXT_DIVIDER	7	rw	**Clock source selection for CMU_CLK_7_CTRL** Value can only be modified when clock enable **EN_CLK7** and **EN_ECLK1** are disabled.
CLK8_EXT_DIVIDER	8	rw	**Clock source selection for CMU_CLK8** Value can only be modified when **EN_ECLK0** is disabled. 0_B Use Clock Source CLS0_CLK 1_B Use Clock Source CMU_ECLK0
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

28.11 Cluster Configuration Module (CCM)

28.11.1 Overview

As already mentioned in the chapter “GTM architecture”, each submodule of the GTM is aligned explicitly to a cluster. The Cluster Configuration Module (CCM) enables the configuration of several cluster specific options namely

- cluster's clock frequency,
- module clock gating,
- Status observation of the cluster's MCS bus master (AEM),
- address range protection, and
- global architecture configuration.

The register **CCM[i]_CFG** allows disabling the system clock signal for unused sub modules of the i-th cluster. The registers **CCM[i]_CMU_CLK_CFG** and **CCM[i]_CMU_FXCLK_CFG** allows the configuration of various cluster clock frequencies.

Figure 29 shows important details about the wiring of the cluster's local clock signals.

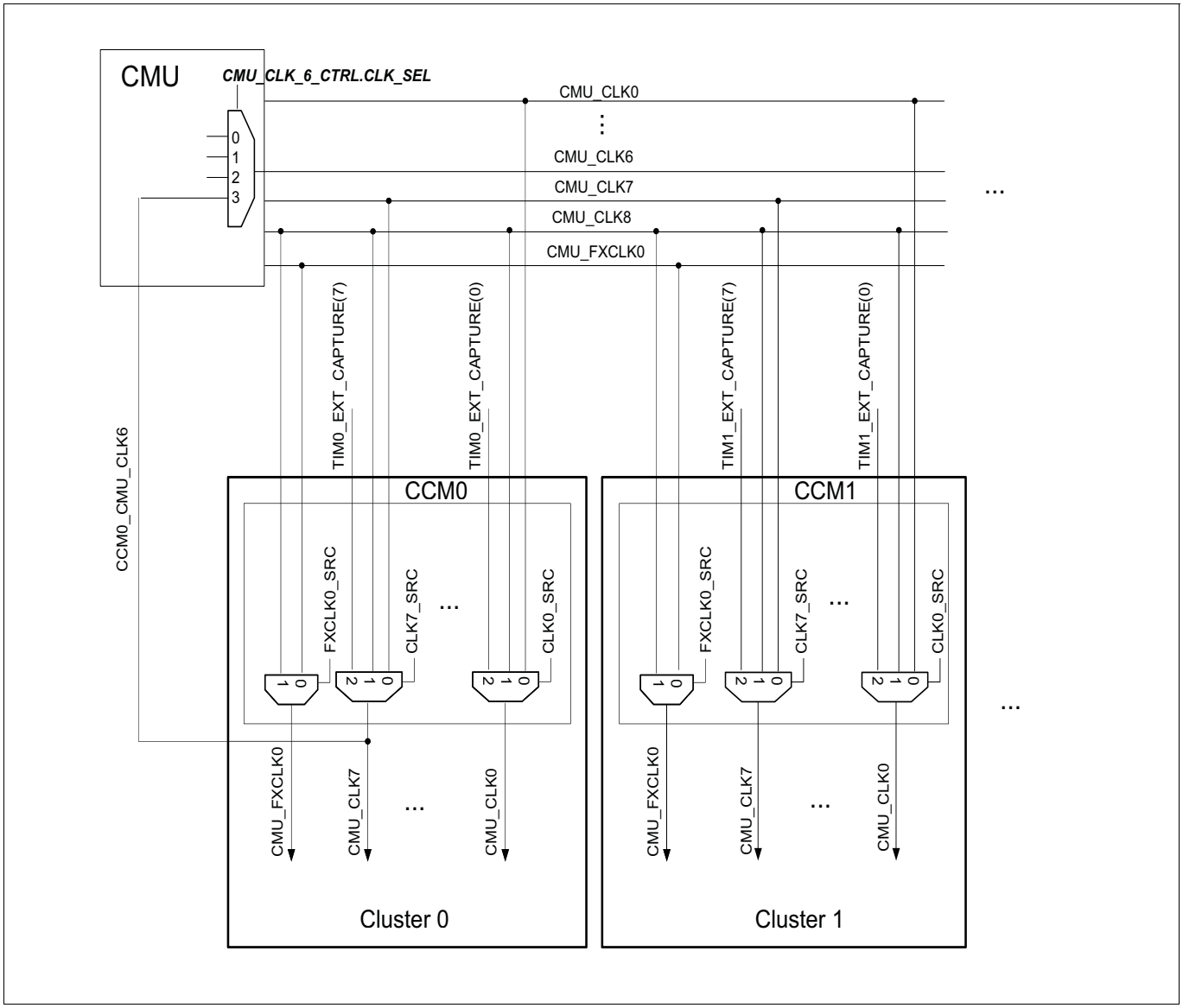


Figure 29 Cluster Clock Signal Wiring

The register **CCM[i]_AEIM_STA** captures the address and the reason of the first invalid AEIM bus master access of the cluster's MCS module.

The registers **CCM[i]_HW_CONF**, **CCM[i]_AUX_IN_SRC**, **CCM[i]_EXT_CAP_EN**, **CCM[i]_TOM_OUT**, and **CCM[i]_ATOM_OUT** are global status and configuration registers that are mirrored from the group of TOP-Level registers. The intention of these registers is to bring up cluster specific configuration registers into the address space of the bus master of the cluster's MCS module.

28.11.2 Address Range Protection

The CCM also provides up to NARP so called address range protectors (ARPs), where the number NARP depends on the actual device configuration (defined in device specific appendix). An ARP can be used to define a configurable write protected address range in order to support enhanced safety features. The address width of and ARP is also device dependent and it is determined by the parameter AAW as defined in device specific appendix.

The protected address range is mapped to the address range of the cluster's MCS RAM port.

Each ARP z (with z = 0…NARP-1) can be configured by the registers **CCM[i]_ARP[z]_CTRL** and **CCM[i]_ARP[z]_PROT**, where the register **CCM[i]_ARP[z]_CTRL** enables to configure the size and base address offset for an ARP and the register **CCM[i]_ARP[z]_PROT** configures, that an MCS channel x with a set bit field **WPROTx** cannot write to the corresponding z-th ARP. Whenever an MCS channel x is writing to an ARP that does not allow a write access from channel x by the configuration register **CCM[i]_ARP[z]_PROT**, the write access is discard. The bit field **WPROT_AEI** of register **CCM[i]_ARP[z]_CTRL** allows to configure if a CPU write access (via AEI slave interface) to the z-th ARP is protected. If the CPU wants to write to the z-th ARP while **WPROT_AEI** is set, the write access will be discard and the AEI status signal will signalize an invalid module access.

Considering the size and base address of an ARP, it should be noted that the configuration possibilities are limited. Details about the configuration can be found in the register description of **MCS[i]_ARP[z]_CTRL**.

The bit field **DIS_PROT** of register **CCM[i]_ARP[z]_CTRL** changes the meaning of an ARP configuration in a way that it explicitly allows an MCS channel x with a set bit field **WPROTx** to write to the z-th ARP. Accordingly, if the bit **DIS_PROT** is set while the bit **WPROT_AEI** is also set in the register **CCM[i]_ARP[z]_CTRL**, the z-th ARP explicitly allows a write access from the CPU to the z-th ARP. A meaningful application of an ARP z with a set bit field **DIS_PROT** for an MCS channel x has another ARP with a surrounding wider address range that is defining a write protection for MCS channel x and some other MCS channels.

Since the address range of an ARP can surround another ARP it is possible to configure contradictory conditions for MCS channels or the CPU within the overlapping area (e.g. if ARP y surrounds ARP z and ARP y allows a write access for an MCS channel x but ARP z prohibits a write access for MCS channel x). In order to resolve this ambiguity, the following rule is defined: A write protection for a specific address c concerning MCS channel x (the CPU) is active, if and only if, address c is covered by at least one ARP with a cleared bit **DIS_PROT** and a set bit **WPROTx** (**WPROT_AEI**) and there exists no ARP covering address c with a set bit field **DIS_PROT** and a set bit field **WPROTx** (**WPROT_AEI**).

28.11.3 CCM Configuration Register Overview

Table 26 CCM Configuration Register Overview

Register name	Description	see Page
CCM[i]_PROT	CCMi Protection Register	119
CCM[i]_CFG	CCMi Configuration Register	119
CCM[i]_CMU_CLK_CFG	CCMi CMU Clock Configuration Register	121
CCM[i]_CMU_FXCLK_CFG	CCMi CMU Fixed Clock Configuration Register	122
CCM[i]_AEIM_STA	CCMi MCS Bus Master Status Register	122
CCM[i]_ARP[z]_CTRL	CCMi Address Range Protector z Control Register	123
CCM[i]_ARP[z]_PROT	CCMi Address Range Protector z Protection Register	124
CCM[i]_HW_CONF	CCMi Hardware Configuration Register	125
CCM[i]_TIM_AUX_IN_SRC	CCMi TIM AUX input source Register.	128
CCM[i]_EXT_CAP_EN	CCMi External Capture Enable Register.	129
CCM[i]_TOM_OUT	CCMi TOM Output Register.	130
CCM[i]_ATOM_OUT	CCMi ATOM Output Register.	130

28.11.4 CCM Configuration Register description

28.11.4.1 Register CCM[i]_PROT

CCMi Protection Register

CCMi_PROT (i=0-11)
CCMi Protection Register **($0E21FC_H$+i*200_H)** **Application Reset Value: 0000 0001_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															CLS_PROT
r															rw

Field	Bits	Type	Description
CLS_PROT	0	rw	**Cluster Protection** 0_B Write protection of cluster configuration registers is disabled 1_B Write protection of cluster configuration registers is enabled
0	31:1	r	**Reserved** Read as zero, shall be written as zero.

28.11.4.2 Register CCM[i]_CFG

CCMi Configuration Register

NOTE: The module specific clock enable registers (bit field **EN_***) are only implemented if the corresponding module is available in the i-th cluster.

NOTE: For the Clusters greater than 4, (only 100MHz capable), the only allowed settings for the CLS_CLK_DIV are 00 and 10 (clock divider 2).

CCMi_CFG (i=0-11)
CCMi Configuration Register **($0E21F8_H$+i*200_H)** **Application Reset Value: XXXX $XXXX_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBU_DIR2	TBU_DIR1	0												CLS_CLK_DIV	
r	r	r												r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								EN_CMP_MON	EN_PSM	EN_BRC	EN_DPLL_MAP	EN_MCS	EN_ATOM_ADTM	EN_TOM_SPE_TDTM	EN_TIM
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
EN_TIM	0	rw	**Enable TIM** This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Disable clock signal for sub module TIM 1_B Enable clock signal for sub module TIM
EN_TOM_SPE_TDTM	1	rw	**Enable TOM, SPE and TDTM** This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Disable clock signal for modules TOM, SPE, and their related DTM modules 1_B Enable clock signal for modules TOM, SPE, and their related DTM modules.
EN_ATOM_ADTM	2	rw	**Enable ATOM and ADTM** This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Disable clock signal for modules ATOM and their related DTM modules. 1_B Enable clock signal for modules ATOM and their related DTM modules.
EN_MCS	3	rw	**Enable MCS** This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Disable clock signal for module MCS 1_B Enable clock signal for module MCS
EN_DPLL_MAP	4	rw	**Enable DPLL and MAP** This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Disable clock signal for modules DPLL and MAP 1_B Enable clock signal for modules DPLL and MAP
EN_BRC	5	rw	**Enable BRC** This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Disable clock signal for module BRC 1_B Enable clock signal for module BRC
EN_PSM	6	rw	**Enable PSM** This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Disable clock signal for module PSM 1_B Enable clock signal for module PSM
EN_CMP_MON	7	rw	**Enable CMP and MON** This bit is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Disable clock signal for modules CMP and MON 1_B Enable clock signal for modules CMP and MON

Field	Bits	Type	Description
CLS_CLK_DIV	17:16	r	**Cluster Clock Divider** The value of this bit field mirrors the bit field **CLS[i]_CLK_DIV** of register **GTM_CLS_CLK_CFG**, whereas i equals the cluster index. 00_B Cluster is disabled 01_B Cluster is enabled without clock divider 10_B Cluster is enabled with clock divider 2 11_B Reserved, do not use.
TBU_DIR1	30	r	**DIR1 input signal of module TBU** 0_B Indicating forward direction 1_B Indicating backward direction
TBU_DIR2	31	r	**DIR2 input signal of module TBU** 0_B Indicating forward direction 1_B Indicating backward direction
0	15:8, 29:18	r	**Reserved** Read as zero, shall be written as zero.

28.11.4.3 Register CCM[i]_CMU_CLK_CFG

CCMi CMU Clock Configuration Register

The bit fields of this register are only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

CCMi_CMU_CLK_CFG (i=0-11)
CCMi CMU Clock Configuration Register **($0E21F0_H$+i*200_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**CLK7_SRC**		**0**		**CLK6_SRC**		**0**		**CLK5_SRC**		**0**		**CLK4_SRC**	
r		rw		r		rw		r		rw		r		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**CLK3_SRC**		**0**		**CLK2_SRC**		**0**		**CLK1_SRC**		**0**		**CLK0_SRC**	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
CLKx_SRC (x=0-7)	4*x+1:4*x	rw	**Clock x source signal selector** 00_B Use CMU_CLKx signal of CMU as CMU_CLKx signal within cluster 01_B Use CMU_CLK8 signal of CMU as CMU_CLKx signal within cluster 10_B Use TIM[i]_EXT_CAPTURE(x) signal as CMU_CLKx signal within cluster 11_B Reserved

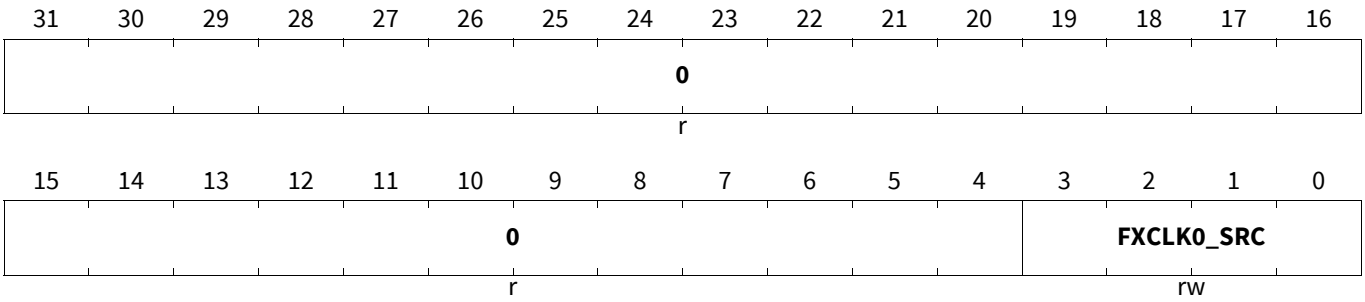
Field	Bits	Type	Description
0	31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2	r	**Reserved** Read as zero, shall be written as zero.

28.11.4.4 Register CCM[i]_CMU_FXCLK_CFG

CCMi CMU Fixed Clock Configuration Register

CCMi_CMU_FXCLK_CFG (i=0-11)

CCMi CMU Fixed Clock Configuration Register(0E21F4$_H$+i*200$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
FXCLK0_SRC	3:0	rw	**Fixed clock 0 source signal selector** Bit field values that are not mentioned above are reserved. These bits are only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_H Use CMU_FXCLK0 signal of CMU as CMU_FXCLK0 signal within cluster 1_H Use CMU_CLK8 signal of CMU as CMU_FXCLK0 signal within cluster
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.11.4.5 Register CCM[i]_AEIM_STA

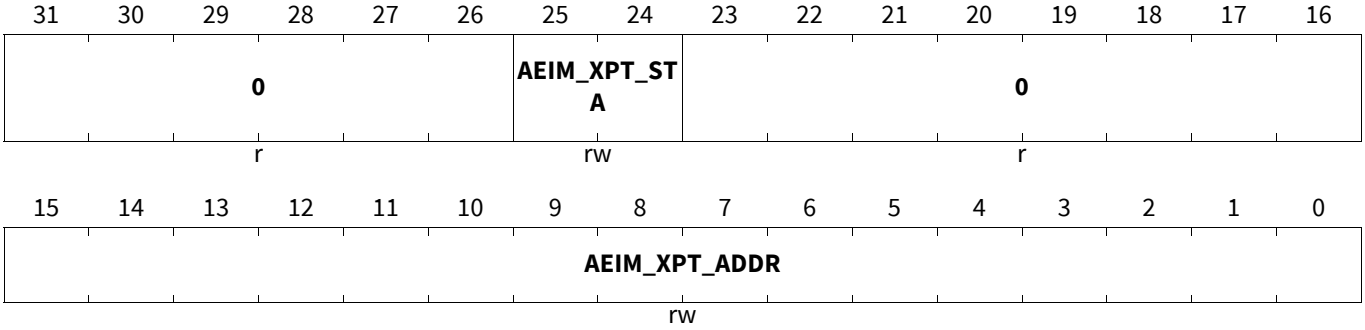
CCMi MCS Bus Master Status Register

Note: Only the first invalid AEIM bus master access of the MCS is updating this register with the invalid AEIM address (bit field AEIM_XPT_ADDR) and the reason of the invalid access (bit field AEIM_XPT_STA). A write access to this register (independent of the written data) always resets the bit fields AEIM_XPT_STA and AEIM_XPT_ADDR, and the next invalid AEIM access is captured by this register, again.

Note: If the i-th cluster does not provide an MCS module, this register is not available.

CCMi_AEIM_STA (i=0-11)

CCMi MCS Bus Master Status Register (0E21D8$_H$+i*200$_H$) Application Reset Value: 0000 0000$_H$



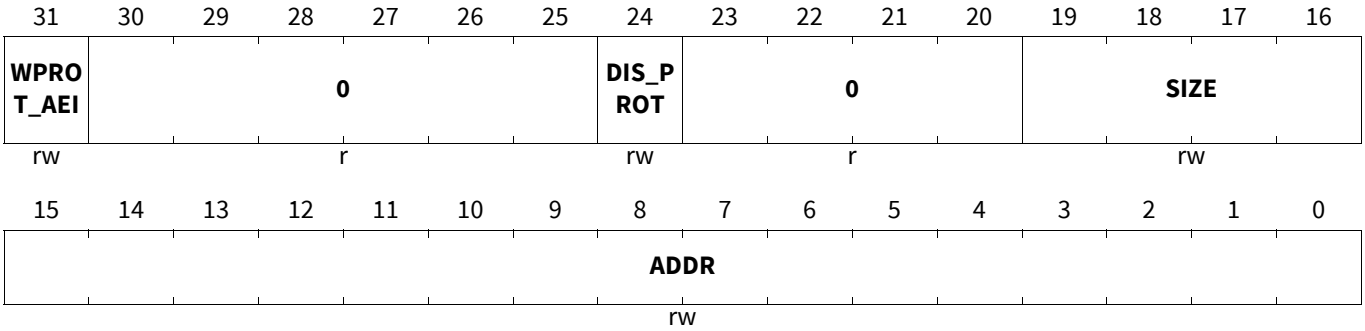
Field	Bits	Type	Description
AEIM_XPT_ADDR	15:0	rw	**Exception Address** Invalid bus master (AEIM) address of MCS module.
AEIM_XPT_STA	25:24	rw	**AEIM exception status** 00_B No invalid MCS bus master access occurred 01_B Invalid byte addressing of MCS bus master access 10_B Illegal module access of MCS bus master access 11_B Invalid MCS bus master access to an unsupported address
0	23:16, 31:26	r	**Reserved** Read as zero, shall be written as zero.

28.11.4.6 Register CCM[i]_ARP[z]_CTRL

CCM0 Address Range Protector z Control Register

CCMi_ARPz_CTRL (i=0-9;z=0-9)

CCMi Address Range Protector z Control Register(0E2000$_H$+i*200$_H$+z*8) Application Reset Value: 0003 0000$_H$



Field	Bits	Type	Description
ADDR	15:0	rw	**ARP base address** Base address for address range protector z. Only the bits 5 to AAW-1 of this bit field are implemented as registers. The bits AAW to 15 are reserved bits and always read and written as zeros. The bits 0 and 1 are functionally used for the definition of an ARP but they are always read and written as zeros. The actual base address for a protected address range is only defined by the upper AAW-(**SIZE**+2) bits (bit position 2+**SIZE** to bit position AAW-1) of bit field **ADDR**. The lower **SIZE**+2 bits (bit 0 to **SIZE**+1) are ignored for the address calculation and assumed as zeros. This bit field is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.
SIZE	19:16	rw	**Size of ARP** Size of memory range protector z. The actual size of a protected memory range is defined as 2^{SIZE} address locations, whereas the bit field **SIZE** is interpreted as an unsigned integer number. This bit field is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.
DIS_PROT	24	rw	**Disable ARP protection** This bit field is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Bit WPROTx (WPROT_AEI) defines write protection for selected address range 1_B Bit WPROTx (WPROT_AEI) explicitly allows write access to selected address range
WPROT_AEI	31	rw	**AEI slave write protection** The address range interval that is protected by this ARP can be calculated as [(**ADDR** AND NOT 4*(2^{SIZE}-1)); (**ADDR** AND NOT 4*(2^{SIZE}-1)) + 4*(2^{SIZE}-1)], assuming a byte-wise addressing, an unsigned integer representation for the bit fields **SIZE** and **ADDR**. NOT and AND are bitwise logical operators. The incrementation interval for neighboring memory locations is always 4. This bit field is only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared. 0_B Write protection to address range from AEI slave is disabled 1_B Write protection to address range from AEI slave is enabled
0	23:20, 30:25	r	**Reserved** Read as zero, shall be written as zero.

28.11.4.7 Register CCM[i]_ARP[z]_PROT

CCM0 Address Range Protector z Protection Register

Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits. Parameter T reflects the number of available MCS channels in the cluster's MCS module.
These bit fields of this register are only writable if bit field **CLS_PROT** of register **CCM[i]_PROT** is cleared.

The meaning of the bit fields **WPROTx** can be changed by the bit field **DIS_PROT** of register **CCM[i]_ARP[z]_CTRL**.

CCMi_ARPz_PROT (i=0-9;z=0-9)
CCMi Address Range Protector z Protection Register($0E2004_H$+i*200_H+z*8) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								WPROT7	WPROT6	WPROT5	WPROT4	WPROT3	WPROT2	WPROT1	WPROT0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
WPROTy (y=0-7)	y	rw	**Write Protection MCS channel y** 0_B Write protection to ARP's address range for MCS channel y is disabled 1_B Write protection to ARP's address range for MCS channel y is enabled
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.11.4.8 Register CCM[i]_HW_CONF

CCMi Hardware Configuration Register

CCMi_HW_CONF (i=0-11)
CCMi Hardware Configuration Register ($0E21DC_H$+i*200_H) Application Reset Value: 084F $022E_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		INT_CLK_EN_GEN	TOM_TRIG_INTCHAIN					ATOM_TRIG_INTCHAIN				IRQ_MODE_SINGLE_PULS	IRQ_MODE_PULSE_NOTIF	IRQ_MODE_PULSE	IRQ_MODE_LEVEL
r		r	r					r				r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ARU_CONNECT_CONFIG	ERM	RAM_INIT_RST	TOM_TRIG_CHAIN			TOM_OUT_RST	ATOM_TRIG_CHAIN			ATOM_OUT_RST	CFG_CLOCK_RATE	SYNC_INPUT_REG	BRIDGE_MODE_RST	GRSTEN
r	r	r	r	r			r	r			r	r	r	r	r

Field	Bits	Type	Description
GRSTEN	0	r	**Global Reset Enable** 0_B Global GTM reset register disabled 1_B Global GTM reset register enabled
BRIDGE_MODE_RST	1	r	**Bridge mode after reset** 0_B Bridge starts in synchronous mode after reset 1_B Bridge starts in asynchronous mode after reset
SYNC_INPUT_REG	2	r	**Additional pipelined stage in synchronous bridge mode** *Note: this register is only relevant (if existing) for synchronous bridge mode* 0_B No additional pipelined stage implemented. 1_B Additional pipelined stage implemented. All accesses in synchronous mode will be increased by one clock cycle.
CFG_CLOCK_RATE	3	r	**Clocks per ARU transfer** *Note: This value defines also the availability of configuration bits in register GTM_CLS_CLK_CFG.* If CFG_CLOCK_RATE=0, only the values 00_B and 01_B are valid for bit fields CLS[x]x_CLK_DIV. If CFG_CLOCK_RATE=1, only the values 00_B, 01_B and 10_B are valid for bit fields CLS[x]x_CLK_DIV. 0_B Each system clock an ARU transfer is scheduled 1_B Each second system clock an ARU transfer is scheduled. ARU transfer rate is half the system clock frequency.
ATOM_OUT_RST	4	r	**ATOM_OUT reset level** *Note: This value represents the ATOM output level after reset. The inverse value of this bit is the reset value of bit SL in all ATOM channels.* 0_B ATOM_OUT reset level is '0' 1_B ATOM_OUT reset level is '1'
ATOM_TRIG_CHAIN	7:5	r	**ATOM trigger chain length without synchronization register** It defines after which ATOM instance count a synchronization register is introduced into trigger chain (after ATOM_TRIG_<i> output if instance i and ATOM_TRIG_<i+1> input of instance i+1). Valid values are 1 to 7. 1 means that after each instance, a synchronization register is placed.
TOM_OUT_RST	8	r	**TOM_OUT reset level** *Note: This value represents the TOM output level after reset. The inverse value of this bit is the reset value of bit SL in all TOM channels.* 0_B TOM_OUT reset level is '0' 1_B TOM_OUT reset level is '1'

Field	Bits	Type	Description
TOM_TRIG_CHAIN	11:9	r	**TOM trigger chain length without synchronization register** It defines after which TOM instance count a synchronization register is introduced into trigger chain (after TOM_TRIG_<i> output if instance i and TOM_TRIG_<i+1> input of instance i+1). Valid values are 1 to 7. 1 means that after each instance, a synchronization register is placed.
RAM_INIT_RST	12	r	**RAM initialization from reset** 0_B RAM is not initialized after reset 1_B RAM is initialized after reset
ERM	13	r	**Enable RAM1 MSB for available MCS modules** *Note: The bit reflects the state of the configuration parameter ERM mentioned in the specification of MCFG.* 0_B MSB of MCS RAM1 address not used 1_B MSB of MCS RAM1 address used
ARU_CONNECT_CONFIG	14	r	**Defines number of parallel ARU ports** 0_B Two ARU ports available (two independent counter) 1_B One ARU port available
IRQ_MODE_LEVEL	16	r	**IRQ_MODE_LEVEL** 0_B Level mode not available 1_B Level mode available
IRQ_MODE_PULSE	17	r	**IRQ_MODE_PULSE** 0_B Pulse mode not available 1_B Pulse mode available
IRQ_MODE_PULSE_NOTIFY	18	r	**IRQ_MODE_PULSE_NOTIFY** 0_B Pulse notify mode not available 1_B Pulse notify mode available
IRQ_MODE_SINGLE_PULSE	19	r	**IRQ_MODE_SINGLE_PULSE** 0_B Single pulse mode not available 1_B Single pulse mode available
ATOM_TRIG_INTCHAIN	23:20	r	**ATOM internal trigger chain length without synchronization register** ATOM internal trigger chain length without synchronization register It defines after which ATOM channel count a synchronization register is introduced into trigger chain. Valid values are 1 to 8. 4 means that in channel 4 of the atom instances a synchronization register is placed.
TOM_TRIG_INTCHAIN	28:24	r	**TOM internal trigger chain length without synchronization register** It defines after which TOM channel count a synchronization register is introduced into trigger chain. Valid values are 1 to 16. 8 means that in channel 8 of the TOM instances, a synchronization register is placed.
INT_CLK_EN_GEN	29	r	**Internal clock enable generation** 0_B GTM external clock enable signals in use 1_B GTM internal clock enable signals in use

Field	Bits	Type	Description
0	15, 31:30	r	**Reserved** Read as zero, shall be written as zero.

28.11.4.9 Register CCM[i]_TIM_AUX_IN_SRC

CCMi TIM Module AUX_IN Source Selection Register

CCMi_TIM_AUX_IN_SRC (i=0-11)
CCMi TIM Module AUX_IN Source Selection Register($0E21E0_H$+i*200_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SEL_OUT_N_CH7	SEL_OUT_N_CH6	SEL_OUT_N_CH5	SEL_OUT_N_CH4	SEL_OUT_N_CH3	SEL_OUT_N_CH2	SEL_OUT_N_CH1	SEL_OUT_N_CH0
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								SRC_CH7	SRC_CH6	SRC_CH5	SRC_CH4	SRC_CH3	SRC_CH2	SRC_CH1	SRC_CH0
r								rw	rw	rw	rw	rw	rw	rw	rw

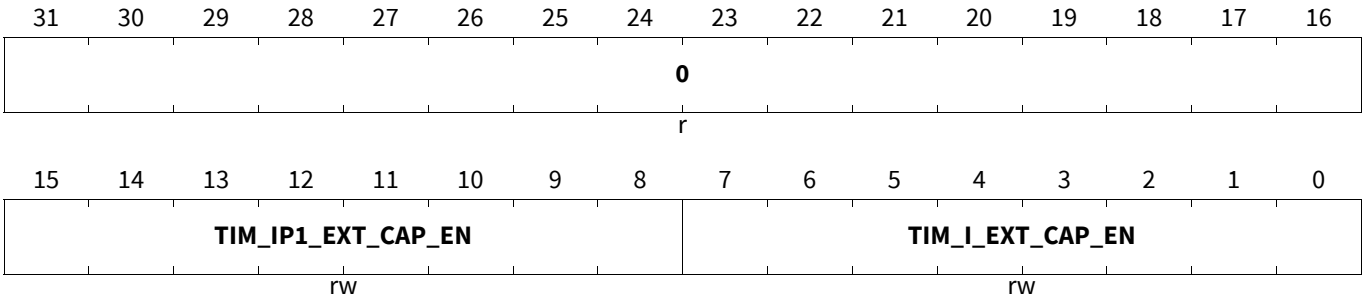
Field	Bits	Type	Description
SRC_CHz (z=0-7)	z	rw	**Defines AUX_IN source of TIM[i] channel z** SRC_CHz: Defines AUX_IN source of TIM[z] channel z SEL_OUT_N_CHz = 0 / SEL_OUT_N_CHz = 1: 0_B Defines AUX_IN source of TIM[i] channel (smaller number) CDTM[z].DTMz Output DTM_OUTz selected / CDTM[z].DTMzOutput 1_B Defines AUX_IN source of TIM[i] channel (higher number) CDTM[z].DTM4 Output DTM_OUT0 selected / CDTM[z].DTM4 Output DTM_OUT1_Nselected
SEL_OUT_N_CHz (z=0-7)	z+16	rw	**Use DTM_OUT or DTM_OUT_N signals as AUX_IN source of TIM[i] channel z** SEL_OUT_N_CHz: Use DTM_OUT or DTM_OUT_N signals as AUX_INsource of TIM[i] channel z 0_B Use DTM_OUT signal as AUX_IN source of TIM[0] 1_B Use DTM_OUT_N signal as AUX_IN source of TIM[0]
0	15:8, 31:24	r	**Reserved** Read as zero, shall be written as zero.

28.11.4.10Register CCM[i]_EXT_CAP_EN

CCMi External Capture Trigger Enable Register

CCMi_EXT_CAP_EN (i=0-11)

CCMi External Capture Trigger Enable Register(0E21E4$_H$+i*200$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TIM_I_EXT_CAP_EN	7:0	rw	**TIM[i]_EXT_CAPTURE signal forwarding enable** Note: The trigger event forwarding is possible from TIM[i] and TIM[i+1] to MCS[i]. 00_H Disable forwarding of signal TIM[i]_EXT_CAPTURE to MCS[i] 01_H Enable forwarding of signal TIM[i]_EXT_CAPTURE to MCS[i]
TIM_IP1_EXT_CAP_EN	15:8	rw	**TIM[i+1]_EXT_CAPTURE signal forwarding enable** 00_H Disable forwarding of signal TIM[i+1]_EXT_CAPTURE to MCS[i] 01_H Enable forwarding of signal TIM[i+1]_EXT_CAPTURE to MCS[i]
0	31:16	r	**Reserved** Note: The trigger event forwarding is possible from TIM[i] and TIM[i+1] to MCS[i].

28.11.4.11Register CCM[i]_TOM_OUT

CCMi TOM Output Level Register

CCMi_TOM_OUT (i=0-11)
CCMi TOM Output Level Register **($0E21E8_H$+i*200_H)** **Application Reset Value: XXXX $XXXX_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOM_OUT_N															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOM_OUT															
r															

Field	Bits	Type	Description
TOM_OUT	15:0	r	Output level snapshot of TOM[i]_OUT all channels
TOM_OUT_N	31:16	r	Output level snapshot of TOM[i]_OUT_N all channels

28.11.4.12Register CCM[i]_ATOM_OUT

CCMi ATOM Output Level Register

*Note: Reset value depends on the hardware configuration chosen by silicon vendor. See **GTM_HW_CONF** for chosen value.*

CCMi_ATOM_OUT (i=0-11)
CCMi ATOM Output Level Register **($0E21EC_H$+i*200_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATOM_IP1_OUT_N								ATOM_IP1_OUT							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOM_I_OUT_N								ATOM_I_OUT							
r								r							

Field	Bits	Type	Description
ATOM_I_OUT	7:0	r	Output level snapshot of ATOM[i]_OUT all channels
ATOM_I_OUT_N	15:8	r	Output level snapshot of ATOM[i]_OUT_N all channels
ATOM_IP1_OUT	23:16	r	Output level snapshot of ATOM[i+1]_OUT all channels
ATOM_IP1_OUT_N	31:24	r	Output level snapshot of ATOM[i+1]_OUT_N all channels

28.12 Time Base Unit (TBU)

28.12.1 Overview

The Time Base Unit TBU provides common time bases for the GTM. The TBU sub-module is organized in channels, where the number of channels is device dependent. There are up to four channels implemented inside the TBU. The time base register **TBU_CH0_BASE** of TBU channel 0 is 27 bits wide and it is configurable whether the lower 24 bit or the upper 24 bit are provided to the GTM as signal *TBU_TS0*. The two TBU channels 1 and 2 have a time base register **TBU_CH[y]_BASE** (y: 1, 2) of 24 bit length. The time base register value *TBU_TS[y]* is provided to subsequent sub-modules of the GTM.

The time base register of TBU channel 3 **TBU_CH3_BASE** is 24 bit wide. It used as a modulo counter by **TBU_CH3_BASE_MARK** to get a relative angle clock to **TBU_CH[y]_BASE**. The absolute angle clock value for the current **TBU_CH3_BASE** is captured in **TBU_CH3_BASE_CAPTURE**.

TBU_CH[y]_BASE = TBU_CH3_BASE_CAPTURE + **... + TBU_CH3_BASE*TBU_CH3_BASE_MARK**

DIRy: direction value for time base y (y:1…2)

0 up counter

1 down counter

Note: The right-hand sum is limited to 24 bit.

The *TBU_UP[y]* (y: 1…2) signals are set to high for a single SYS_CLK period, whenever the corresponding signal *TBU_TS[y]* (y: 1…2) is getting updated. The signal *TBU_UP0_L* is set to high for a single SYS_CLK period if the signal TBU_TS0 and TBU_TS0x is getting updated and *TBU_UP0_H* is set to high for a single SYS_CLK period, whenever the upper 24 bit of TBU_TS0 are updated.

The time base channels can run independently of each other and can be enabled and disabled synchronously by control bits in a global TBU channel enable register **TBU_CHEN**. **Figure 30** shows a block diagram of the Time Base Unit.

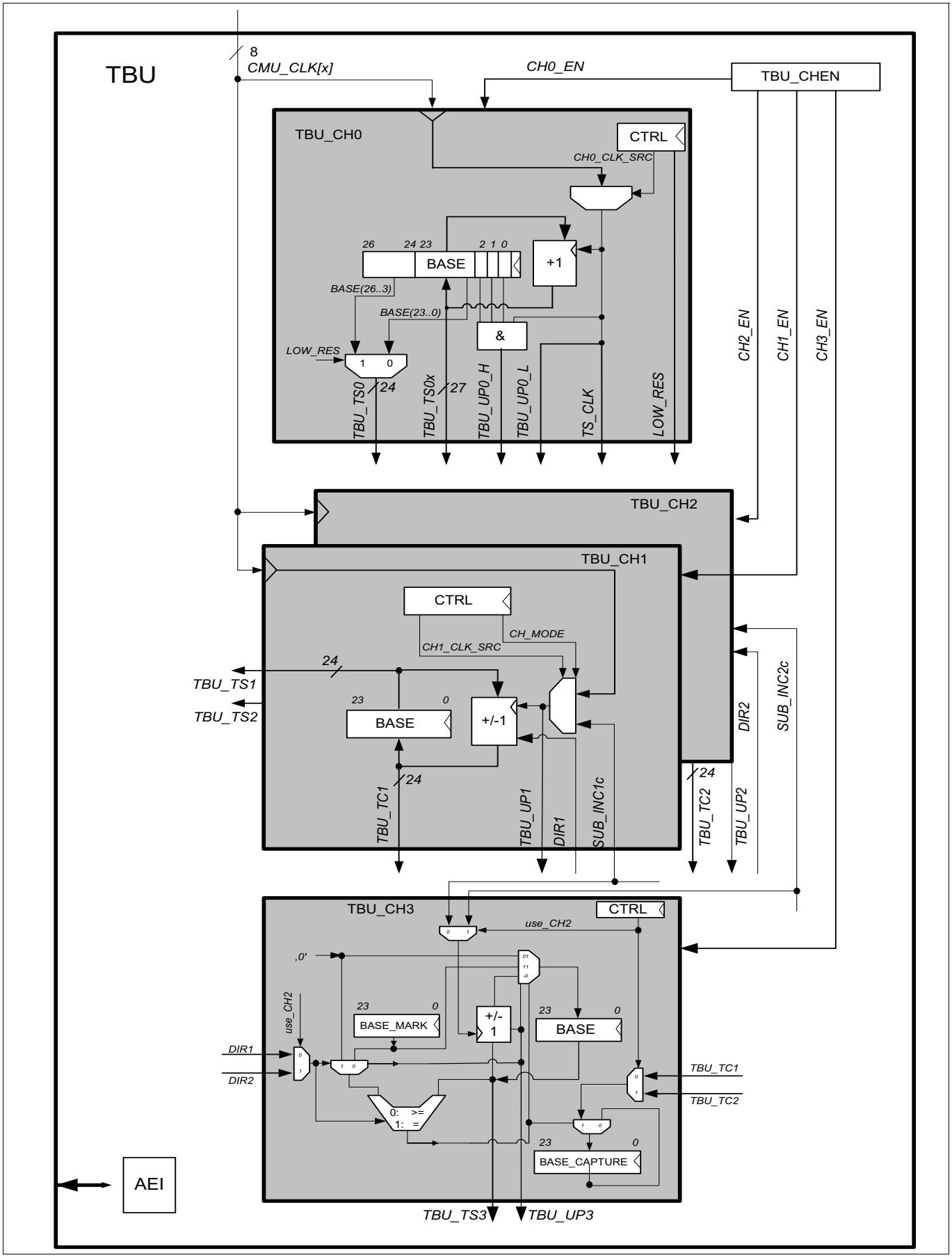


Figure 30 TBU Block Diagram

Dependent on the device a third TBU channel exists which offers the same functionality as time base channel 1.

The configuration of the independent time base channels TBU_CH[z]_BASE is done via the AEI interface. The TBU channel 0 to 2 may select one of the eight *CMU_CLK[x]* (x: 0…7) signals coming from the CMU sub-module.

For TBU channels 1 and 2 an additional clock signal *SUB_INC[y]c* (y: 1, 2) coming from the DPLL can be selected as input clock for the TBU_CH[y]_BASE. This clock in combination with the *DIR[y]* signals determines the counter direction of the TBU_CH[y]_BASE.

The selected time stamp clock signal for the TBU_CH0 sub-unit is served via the *TS_CLK* signal line to the DPLL sub-module. The *TS_CLK* signal equals the signal *TBU_UP0*.

28.12.2 TBU Channels

The time base values are generated within the TBU time base channels in two independent and one dependent operation modes.

In all modes, the time base register **TBU_CH[z]_BASE** (z: 0…3) can be initialized with a start value just before enabling the corresponding TBU channel.

Moreover, the time base register **TBU_CH[z]_BASE** (z: 0…3) can always be read in order to determine the actual value of the counter.

28.12.2.1 Independent Modes

28.12.2.1.1 Free Running Counter Mode

TBU channel 0 provides a 27 bit counter in a free running counter mode. Dependent on the bit field **LOW_RES** of register **TBU_CH0_CTRL,** the lower 24 bits (bit 0 to 23) or the upper 24 bits (bits 3 to 26) are provided to the GTM sub-modules.

TBU channel 1 and 2 provides a 24 bit counter in a free running counter mode enabled by reset CH_MODE of register **TBU_CH[y]_CTRL** (y:1…2).

In TBU Free running counter mode, the time base register **TBU_CH[v]_BASE** (v:0…2) is updated on every specified incoming clock event by the selected signal *CMU_CLK[x]* (x: 0…8) (dependent on **TBU_CH[v]_CTRL** (v:0…2) register). In general the time base register **TBU_CH[v]_BASE** is incremented on every *CMU_CLK[x]* clock tick.

28.12.2.1.2 Forward/Backward Counter Mode

TBU channel 1 and 2 provides a 24 bit forward/backward counter enabled by set CH_MODE of register **TBU_CH[y]_CTRL** (y:1…2). In this mode the *DIR[y]* signal provided by the DPLL is taken into account.

The value of the time base register **TBU_CH[y]_BASE** is incremented in case when the *DIR[y]* signal equals ‘0’ and decremented in case when the *DIR[y]* signal is ‘1’.

28.12.2.2 Dependent Mode

28.12.2.2.1 Modulo Counter Mode

TBU channel 3 provides a 24 bit forward/backward modulo counter. The clock SUB_INC[y]c and counter direction DIR[y] provided by DPLL is selected by use_CH2 of register TBU_CH3_CTRL.

The modulo value is defined in TBU_CH3_BASE_MARK. In forward counter mode if TBU_CH3_BASE value is reaching TBU_CH3_BASE_MARK TBU_CH3_BASE is reset and TBU_TS[y] is captured in TBU_CH3_BASE_CAPTURE. In backward counter mode if TBU_CH3_BASE value is reaching '0' TBU_CH3_BASE is set to TBU_CH3_BASE_MARK and TBU_TS[y] is captured in TBU_CH3_BASE_CAPTURE.

28.12.3 TBU Configuration Register Overview

Table 27 TBU Configuration Register Overview

Register Name	Description	see Page
TBU_CHEN	TBU global channel enable	135
TBU_CH0_CTRL	TBU channel 0 control	135
TBU_CH0_BASE	TBU channel 0 base	136
TBU_CH1_CTRL	TBU channel 1 control	137
TBU_CH[y]_BASE	TBU channel y base	139
TBU_CH2_CTRL	TBU channel 2 control	138
TBU_CH3_CTRL	TBU channel 3 control	140
TBU_CH3_BASE	TBU channel 3 base	140
TBU_CH3_BASE_MARK	TBU channel 3 modulo value	141
TBU_CH3_BASE_CAPTURE	TBU channel 3 base captured	141

In a typical application the Time Base Unit (TBU) considers channels 0, 1 and 3 only. In this case register addresses 0x20…0x2C are reserved and shall be read as zero. Channel 2 can be additionally implemented on special high-end application requirements.

28.12.4 TBU Register description

28.12.4.1 Register TBU_CHEN

TBU Global Channel Enable

TBU_CHEN
TBU Global Channel Enable **(000100_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								ENDIS_CH3		ENDIS_CH2		ENDIS_CH1		ENDIS_CH0	
r								rw		rw		rw		rw	

Field	Bits	Type	Description
ENDIS_CHx (x=0-3)	2*x+1:2*x	rw	**TBU channel x enable/disable control** Write / Read : 00_B Don't care, bits 1:0 will not be changed / channel disabled 01_B Channel disabled: is read as 00 (see below) / -- 10_B Channel enabled: is read as 11 (see below) / -- 11_B Don't care, bits 1:0 will not be changed / channel enabled
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.12.4.2 Register TBU_CH0_CTRL

TBU Channel 0 Control Register

TBU_CH0_CTRL
TBU Channel 0 Control Register **(000104_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												CH_CLK_SRC			LOW_RES
r												rw			rw

Field	Bits	Type	Description
LOW_RES	0	rw	**TBU_CH0_BASE register resolution** The two resolutions for the TBU channel 0 can be used in the TIM channel 0 and the DPLL sub-modules. This value can only be modified if channel 0 is disabled. 0_B TBU channel uses lower counter bits (bit 0 to 23) 1_B TBU channel uses upper counter bits (bit 3 to 26)
CH_CLK_SRC	3:1	rw	**Clock source for channel x (x:0...2) time base counter** This value can only be modified if channel 0 is disabled. 000_B CMU_CLK0 selected 001_B CMU_CLK1 selected 010_B CMU_CLK2 selected 011_B CMU_CLK3 selected 100_B CMU_CLK4 selected 101_B CMU_CLK5 selected 110_B CMU_CLK6 selected 111_B CMU_CLK7 selected
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.12.4.3 Register TBU_CH0_BASE

TBU Channel 0 Base Register

TBU_CH0_BASE
TBU Channel 0 Base Register **(000108_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					**BASE**										
r					rw										

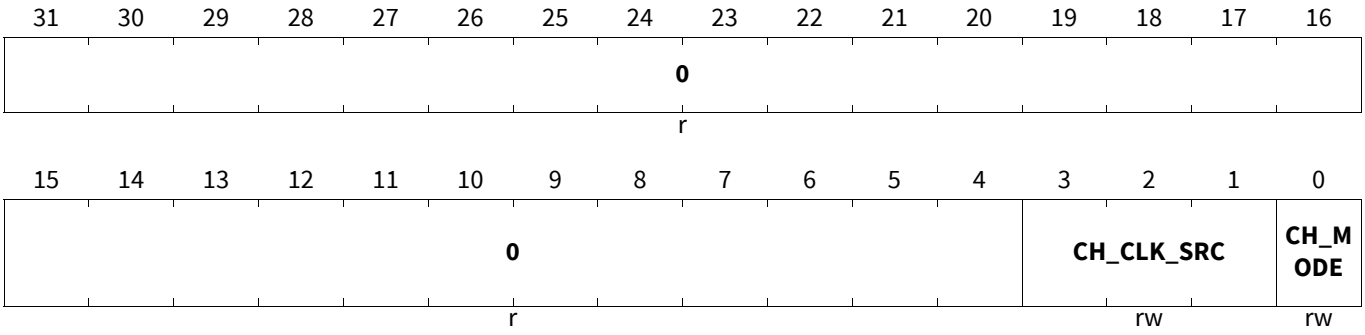
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE															
rw															

Field	Bits	Type	Description
BASE	26:0	rw	**Time base value for channel 0** The value of **BASE** can only be written if the TBU channel 0 is disabled. If channel 0 is enabled, a read access to this register provides the current value of the underlying 27 bit counter.
0	31:27	r	**Reserved** Read as zero, shall be written as zero

28.12.4.4 Register TBU_CH1_CTRL

TBU Channel 1 Control Register

TBU_CH1_CTRL
TBU Channel 1 Control Register **(00010C$_H$)** **Application Reset Value: 0000 0000$_H$**

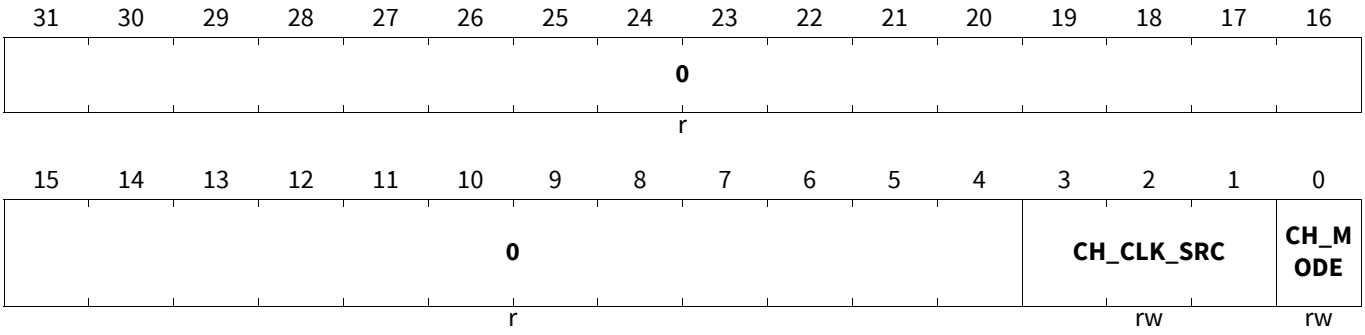


Field	Bits	Type	Description
CH_MODE	0	rw	**Channel mode** This value can only be modified if channel 1 is disabled. In Free running counter mode the CMU clock source specified by **CH_CLK_SRC** is used for the counter. In Forward/Backward counter mode the *SUB_INC1c* clock signal in combination with the *DIR1* input signal is used to determine the counter direction and clock frequency. 0_B Free running counter mode 1_B Forward/backward counter mode
CH_CLK_SRC	3:1	rw	**Clock source for channel 1 time base counter** This value can only be modified if channel 1 was disabled 000_B CMU_CLK0 selected 001_B CMU_CLK1 selected 010_B CMU_CLK2 selected 011_B CMU_CLK3 selected 100_B CMU_CLK4 selected 101_B CMU_CLK5 selected 110_B CMU_CLK6 selected 111_B CMU_CLK7 selected
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.12.4.5 Register TBU_CH2_CTRL

TBU Channel 2 Control Register

TBU_CH2_CTRL
TBU Channel 2 Control Register **(000114_H)** **Application Reset Value: $0000\ 0000_H$**



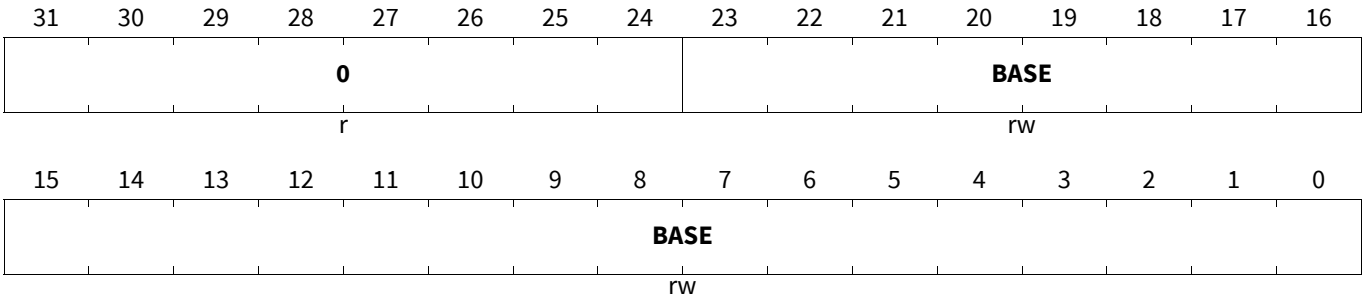
Field	Bits	Type	Description
CH_MODE	0	rw	**Channel mode** This value can only be modified if channel 2 is disabled. In Free running counter mode the CMU clock source specified by **CH_CLK_SRC** is used for the counter. In Forward/Backward counter mode the *SUB_INC2c* clock signal in combination with the *DIR2* input signal is used to determine the counter direction and clock frequency. 0_B Free running counter mode 1_B Forward/backward counter mode
CH_CLK_SRC	3:1	rw	**Clock source for channel 2 time base counter** This value can only be modified if channel 2 was disabled 000_B CMU_CLK0 selected 001_B CMU_CLK1 selected 010_B CMU_CLK2 selected 011_B CMU_CLK3 selected 100_B CMU_CLK4 selected 101_B CMU_CLK5 selected 110_B CMU_CLK6 selected 111_B CMU_CLK7 selected
0	31:4	r	**Reserved** Read as zero, shall be written as zero.

28.12.4.6 Register TBU_CH[y]_BASE

TBU Channel 1 Base Register

TBU_CH1_BASE

TBU Channel 1 Base Register **(000110_H)** **Application Reset Value: $0000\ 0000_H$**

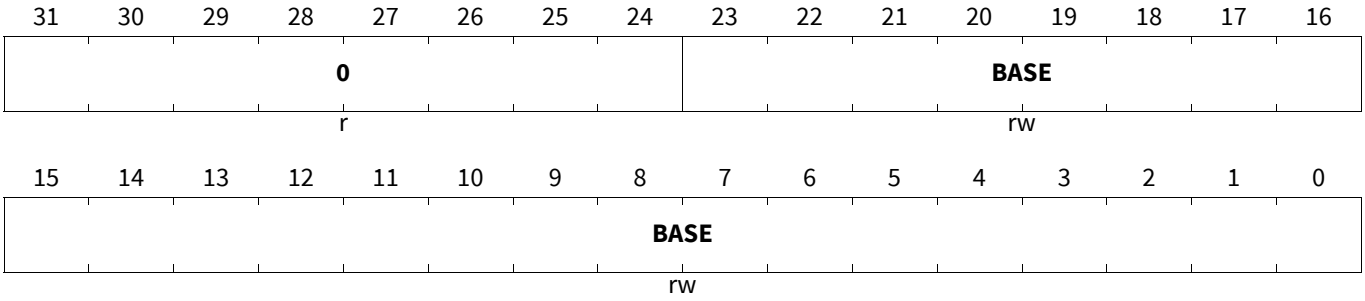


Field	Bits	Type	Description
BASE	23:0	rw	**Time base value for channel y (y: 1, 2)** The value of **BASE** can only be written if the corresponding TBU channel y is disabled. If the corresponding channel y is enabled, a read access to this register provides the current value of the underlying counter.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

TBU Channel 2 Base Register

TBU_CH2_BASE

TBU Channel 2 Base Register **(000118_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
BASE	23:0	rw	**Time base value for channel y (y: 1, 2)** The value of **BASE** can only be written if the corresponding TBU channel y is disabled. If the corresponding channel y is enabled, a read access to this register provides the current value of the underlying counter.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.12.4.7 Register TBU_CH3_CTRL

TBU Channel 3 Control Register

TBU_CH3_CTRL
TBU Channel 3 Control Register **(00011C$_H$)** **Application Reset Value: 0000 0001$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											USE_CH2	0			CH_MODE
r											rw	r			r

Field	Bits	Type	Description
CH_MODE	0	r	**Channel mode** 1 = Forward/backward counter mode
USE_CH2	4	rw	**Channel selector for modulo counter** This value can only be modified if channel 3 was disabled 0_B TBU_CH1 values used (SUB_INC1c for clock, DIR1 for counter direction, TBU_TS1 for capturing) 1_B TBU_CH2 values used (SUB_INC2c for clock, DIR2 for counter direction, TBU_TS2 for capturing)
0	3:1, 31:5	r	**Reserved** Read as zero, shall be written as zero.

28.12.4.8 Register TBU_CH3_BASE

TBU Channel 3 Base Register

TBU_CH3_BASE
TBU Channel 3 Base Register **(000120$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								BASE							
r								rw							

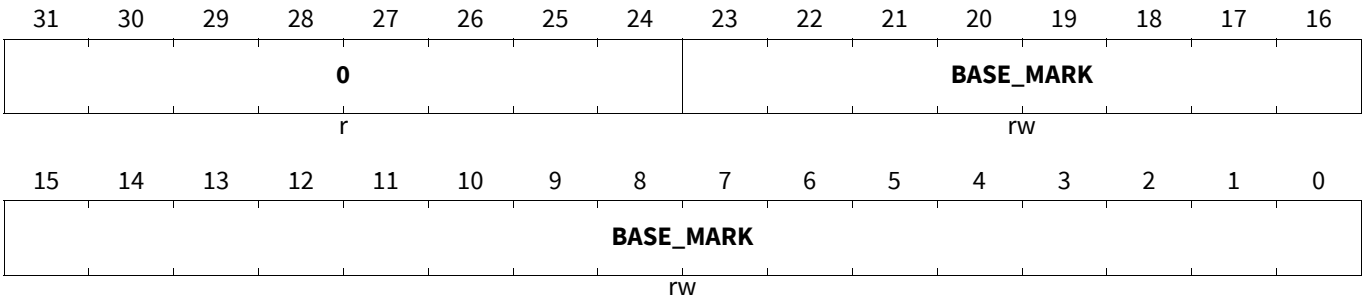
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE															
rw															

Field	Bits	Type	Description
BASE	23:0	rw	**Time base value for channel 3** The value of **BASE** can only be written if the corresponding TBU channel 3 is disabled. If the corresponding channel 3 is enabled, a read access to this register provides the current value of the underlying counter.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.12.4.9 Register TBU_CH3_BASE_MARK

TBU Channel 3 Modulo Value Register

TBU_CH3_BASE_MARK
TBU Channel 3 Modulo Value Register **(000124_H)** **Application Reset Value: 0000 0000$_H$**

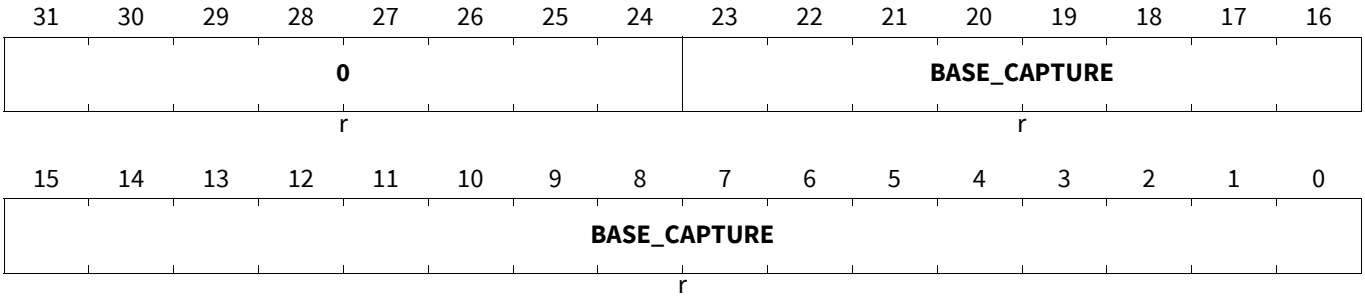


Field	Bits	Type	Description
BASE_MARK	23:0	rw	**Modulo value for channel 3** The value of **BASE_MARK** can only be written if the corresponding TBU channel 3 is disabled.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.12.4.10Register TBU_CH3_BASE_CAPTURE

TBU Channel 3 Base Captured Register

TBU_CH3_BASE_CAPTURE
TBU Channel 3 Base Captured Register **(000128_H)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BASE_CAPTURE	23:0	r	**Captured value of time base channel 1 or 2** When USE_CH2=0, TBU_TS1 is captured, and if USE_CH2 is set TBU_TS2 is captured.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.13 Timer Input Module (TIM)

28.13.1 Overview

The Timer Input Module (TIM) is responsible for filtering and capturing input signals of the GTM. Several characteristics of the input signals can be measured inside the TIM channels. For advanced data processing the detected input characteristics of the TIM module can be routed through the ARU to subsequent processing units of the GTM.

Input characteristics mean either time stamp values of detected input rising or falling edges together with the new signal level or the number of edges received since channel enable together with the actual time stamp or PWM signal duration for a whole PWM period.

The architecture of TIM is shown in **Figure 31**.

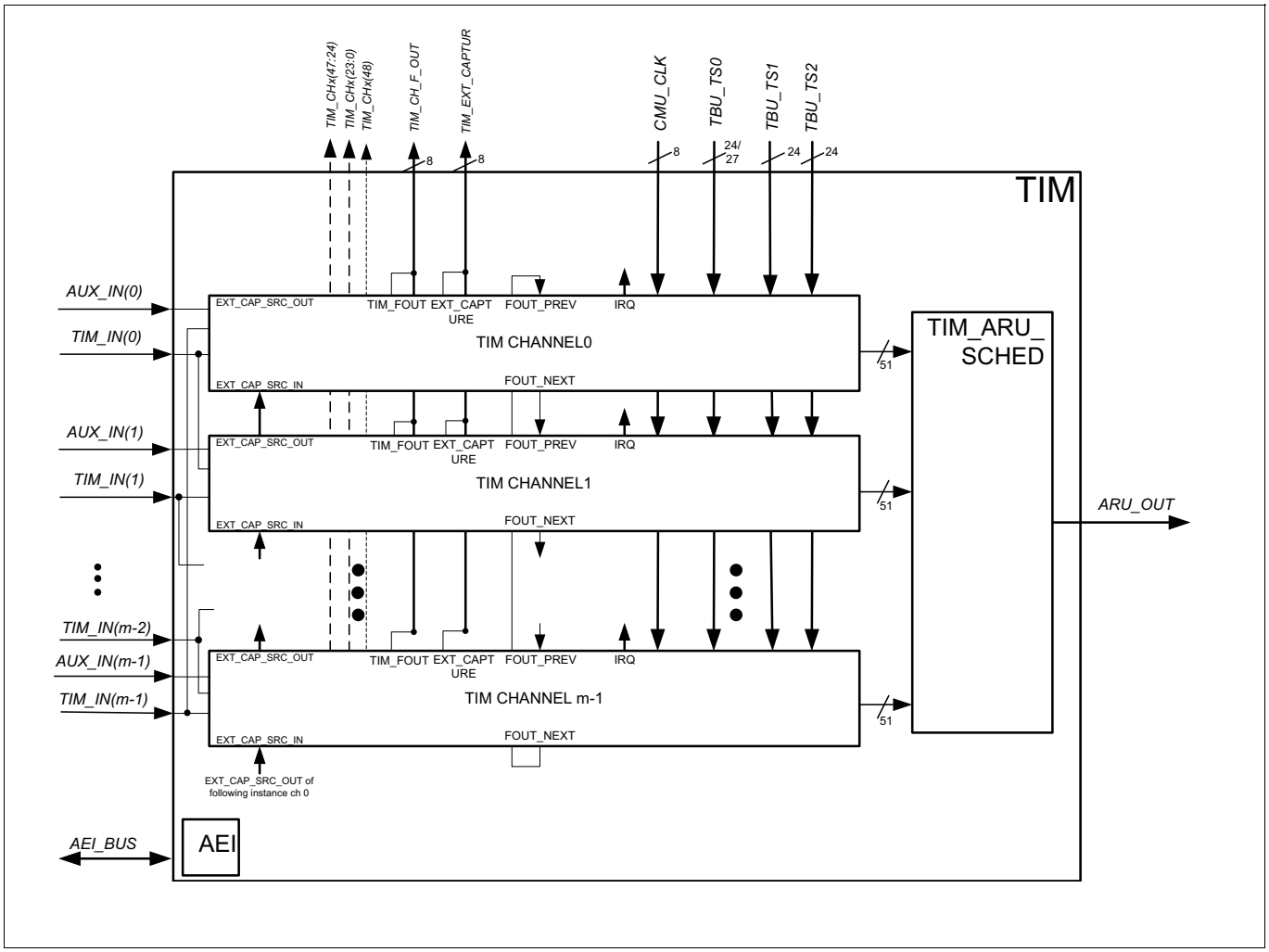


Figure 31 TIM Block Diagram

The number of channels *m* inside a TIM sub-module depends on the device.

Each of the *m* dedicated input signals are filtered inside the FLTx sub-unit of the TIM Module. It should be noted that the incoming input signals are synchronized to the clock SYS_CLK, resulting in a delay of two SYS_CLK periods for the incoming signals.

The measurement values can be read by the CPU directly via the AEI-Bus or they can be routed through the ARU to other sub-modules of the GTM.

For the GTM TIM0 sub-module only, the dashed signal outputs *TIM[i]_CH[x](23:0)*, *TIM[i]_CH[x](47:24)* and *TIM[i]_CH[x](48)* come from the TIM0 sub-module channels zero (0) to five (5) and are connected to MAP sub-module. There, they are used for further processing and for routing to the DPLL.

The two (three) time bases coming from the TBU are connected to the TIM channels to annotate time stamps to incoming signals. For TIM0 the extended 27 bit width time base TBU_TS0 is connected to the TIM channels, and the user has to select if the lower 24 bits (*TBU_TS0(23…0)*) or the higher 24 bits (*TBU_TS0(26…3)*) are stored inside the **GPR0** and **GPR1** registers.

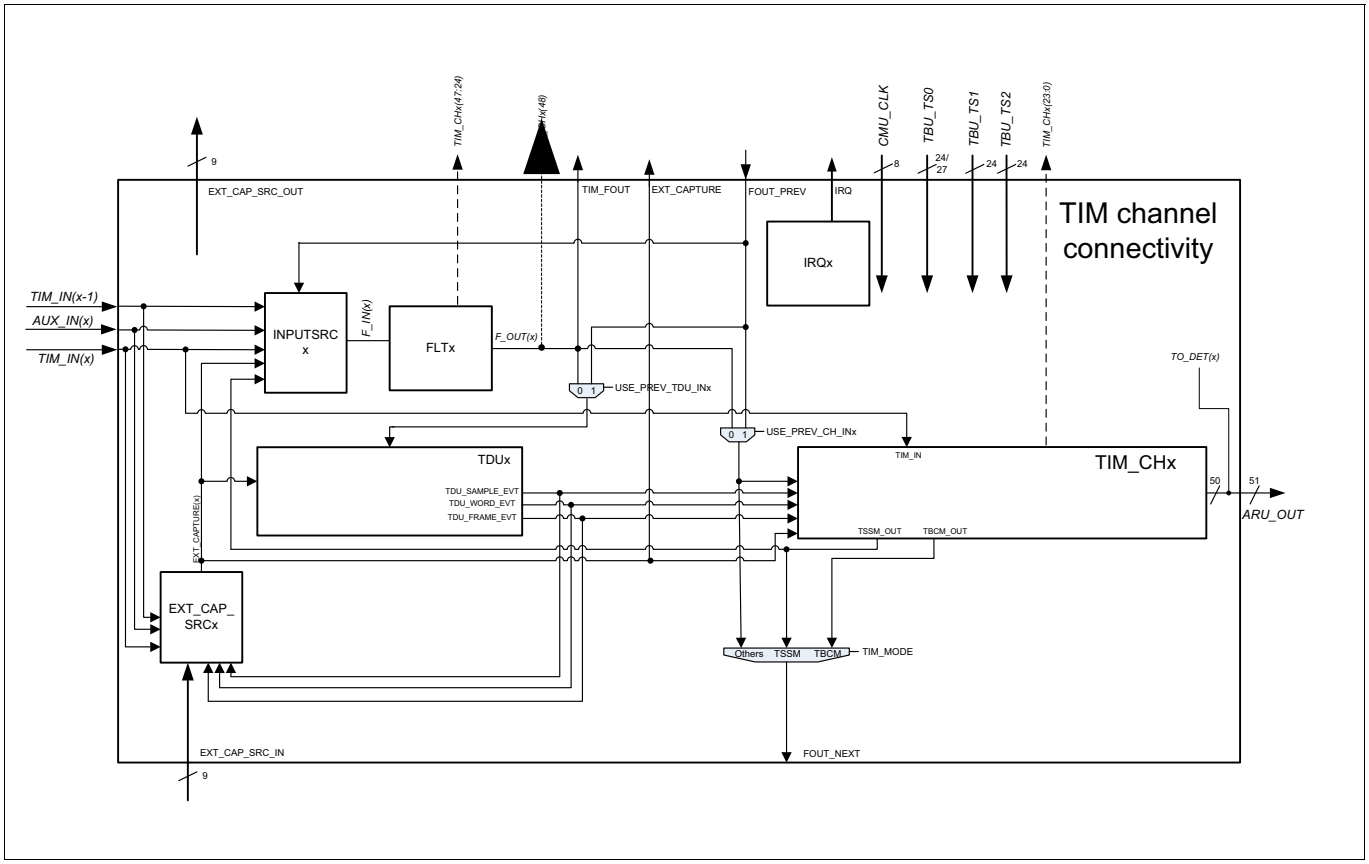


Figure 32 TIM channel internal connectivity

Above figure gives an overview of the channel internal connectivity of the sub units. The sub units with the major functionality are listed next:

- INPUT_SRCx: Select signal for processing by the Filter unit FLTx
- FLTx: The filter unit provides different filter mechanisms described in more detail in **Section 28.13.2**.
- TDUx: Timeout detection unit (no subsequent edge detected during a specified duration
- TIM_CHx: Measurement unit; different measurements strategies configurable on the filtered signal
- IRQx: Local interrupt controller (enabling, status, …)
- EXT_CAP_SRCx: Selects a local signal ext_capture(x) which is needed by certain functions

Details are given in the next chapters.

Depending on the values of the configuration bit fields **USE_PREV_TDU_INx**, **USE_PREV_CH_INx** it is possible to operate on the signal of the local channel x or the previous channel x-1.

Depending on the value of the configuration bit field **TIM_MODEx** it is possible to provide different signals (via FOUT_NEXT) to the next channel.

In TBCM mode each capture event selected by the sensitive edges (CNTS) will be forwarded with the value of ECNT[0] to the following channel (via FOUT_NEXT).

28.13.1.1 Input source selection INPUTSRCx

It can be configured which source shall be used for processing in the FLT,TDU,TIM_CH units. It can be selected by the bit fields **CICTRL** and **MODE_x**, **VAL_x** in the register **TIM[i]_IN_SRC** which source is in use.

Alternatively the signal **F_IN(x)** can be generated by a 8 bit lookup table, which allows to define any function of 3 input sources.

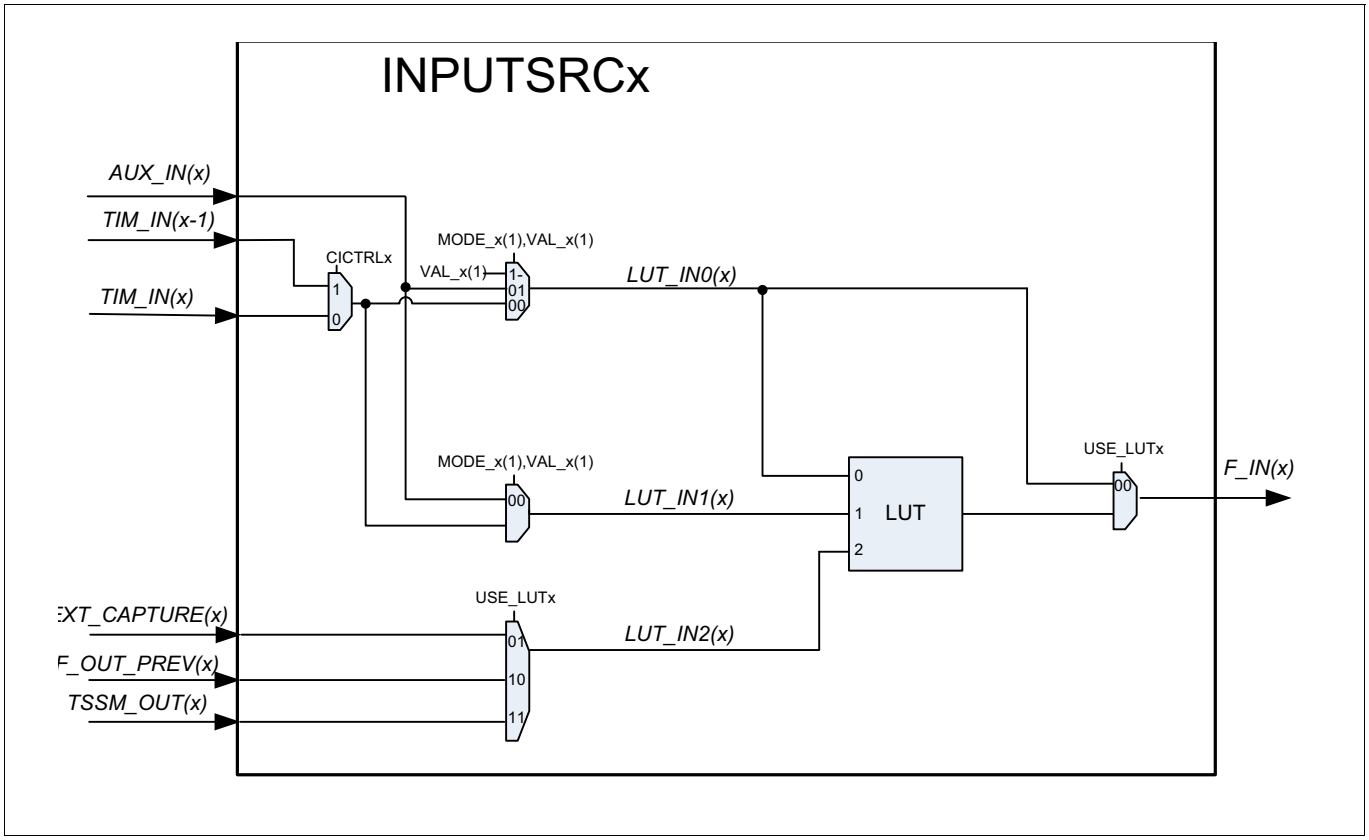


Figure 33 INPUTSRC Block Diagram

If **USE_LUT**=b00 is set the lookup table signal generation is bypassed and the signal selection is performed as follows:

In a certain **MODE_x**, **VAL_x** combination the input signal *F_IN(x)* can be driven by **VAL_x(1)** with 0 or 1 directly.

Due to the fact that all 8 channels are bundled in the register **TIM[i]_IN_SRC** a synchronous control of all 8 input channels is possible.

Two adjacent channels can be combined by setting the **CICTRL** bit field in the corresponding **TIM[i]_CH[x]_CTRL** register. This allows for a combination of complex measurements on one input signal with two TIM channels.

The additional signal **AUX_IN[x]** can be selected as an input signal. The source of this signal is defined in the subchapter “TIM auxiliary input multiplexing”.

If **USE_LUT** !=0b00 is set, the lookup table signal generation with following inputs is in use. See **Figure 33**:

Input **LUT_INO(x)** selection:

TIM_IN(x) if **CICTRLx**=0 and **MODE_x**(1)=0 and **VAL_x**(1)=0
TIM_IN(x-1) if **CICTRLx**=1 and MODE_x(1)=0 and VAL_x(1)=0

AUX_IN(x) if MODE_x(1)=0 and VAL_x(1)=1
VAL_x(1) if MODE_x(1)=1

Input **LUT_IN1(x)** selection:

AUX_IN(x) if **MODE_x**(1)=0 and **VAL_x**(1)=0
TIM_IN(x) if **CICTRLx**=0
TIM_IN(x-1) if **CICTRLx**=1

Input **LUT_IN2(x)** selection:
EXT_CAPTURE(x) if **USE_LUT**=0b01
FOUT_PREV(x) if **USE_LUT**=0b10
TSSM_OUT(x) if **USE_LUT**=0b11

The lookup table is defined by the contents of the bit field **TO_CNT2x**. The lookup_table_index is defined by **LUT_IN2**(x) & **LUT_IN1**(x) & **LUT_IN0**(x). The signal **F_IN**(x) is generated by **TO_CNT2x**[lookup_table_index].

If **USE_LUT** !=0b00 is set, only limited functionality is available in the TDU. See bit field Slicing (**SLICING**) in the register TIM[i]_CH[x]_TDUV.

28.13.1.2 Input observation

It is possible to observe for all channels of one instance by reading **TIM_INP_VAL** the actual signal values of the following processing stages:

- TIM_IN(7:0) signals after TIM input synchronization
- TIM F_IN(7:0) signals after TIM INPUTSRC selection (input to TIM_FLT)
- TIM F_OUT(7:0) signals after TIM filter functionality (output of TIM_FLT)

28.13.1.3 External capture source selection EXTCAPSRCx

Each channel can operate on an external capture signal **EXT_CAPTURE**. The source to use for this signal can be configured by the bit field **EXT_CAP_SRCx** in the register TIM[i]_CH[x]_ECTRL

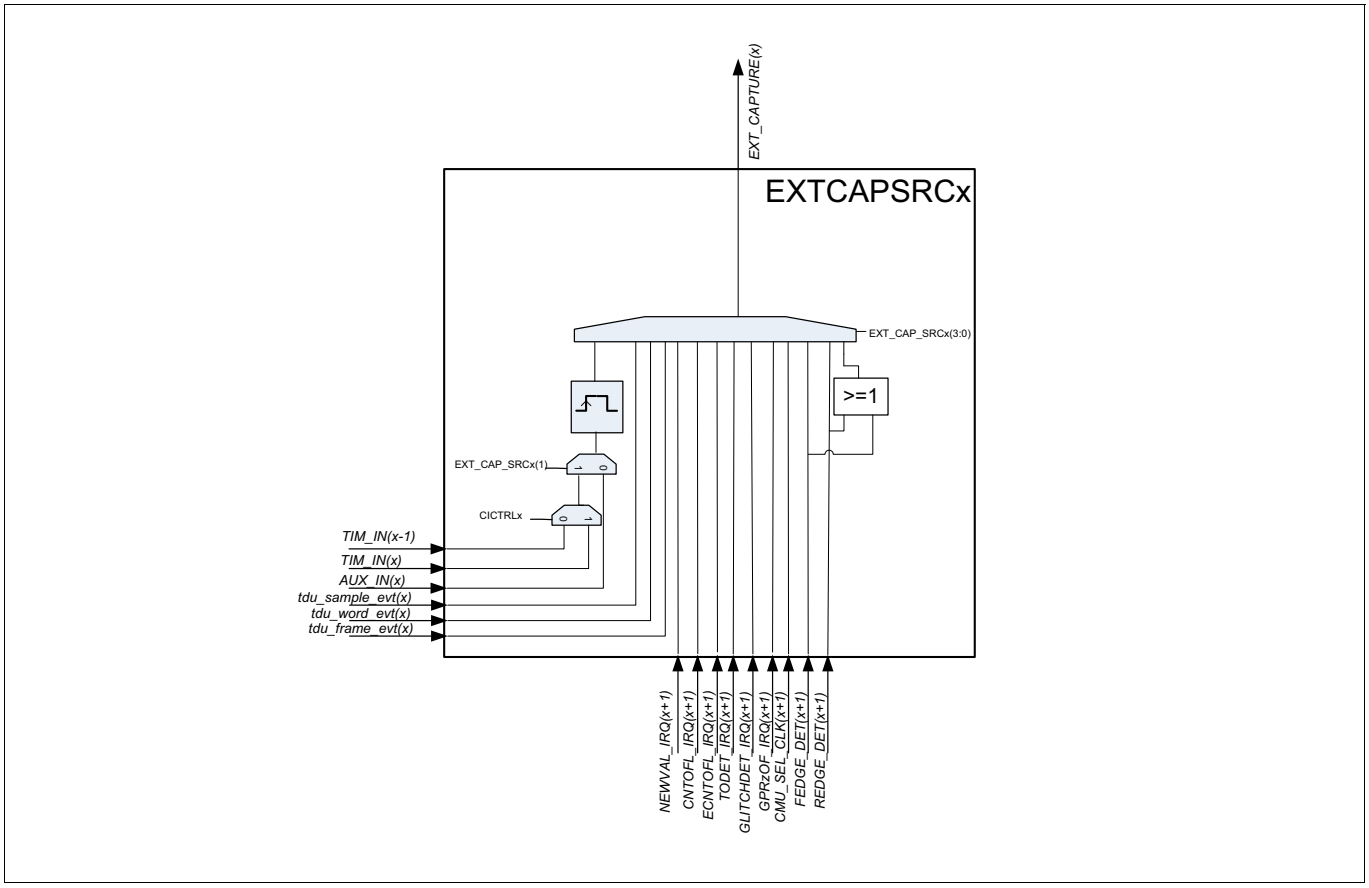


Figure 34 EXTCAPSRC Block Diagram

The external capture functionality can be enabled for the TIM channel **x** with the bit **EXT_CAP_EN** in the register **TIM[i]_CH[x]_CTRL**, it will trigger on each rising edge. A pulse generation for each rising edge of the selected input signal **TIM_IN[x]** and **AUX_IN[x]** is applied.

The six TIM channel interrupt sources can be triggered by the operation in the certain TIM channel modes. Alternatively they can be issued by a soft trigger using the corresponding bits in the register **TIM[i]_CH[x+1]_FORCINT**.

28.13.2 TIM Filter Functionality (FLT)

28.13.2.1 Overview

The TIM sub-module provides a configurable filter mechanism for each input signal. These filter mechanism is provided inside the FLT sub-unit. The FLT architecture is shown in **Figure 35**.

The filter includes a clock synchronization unit (CSU), an edge detection unit (EDU), and a filter counter associated to the filter unit (FLTU).

The CSU is synchronizing the incoming signal *F_IN* to the selected filter clock frequency, which is controlled with the bit field **FLT_CNT_FRQ** of register **TIM[i]_CH[x]_CTRL**.

The synchronized input signal *F_IN_SYNC* is used for further processing within the filter.

It should be noted that glitches with a duration go less than the selected CMU clock period is lost.

The filter modes can be applied individually to the falling and rising edges of an input signal. The following filter modes are available:

- immediate edge propagation mode,
- individual de-glitch time mode (up/down counter), and
- individual de-glitch time mode (hold counter).
- individual de-glitch time mode (reset counter).

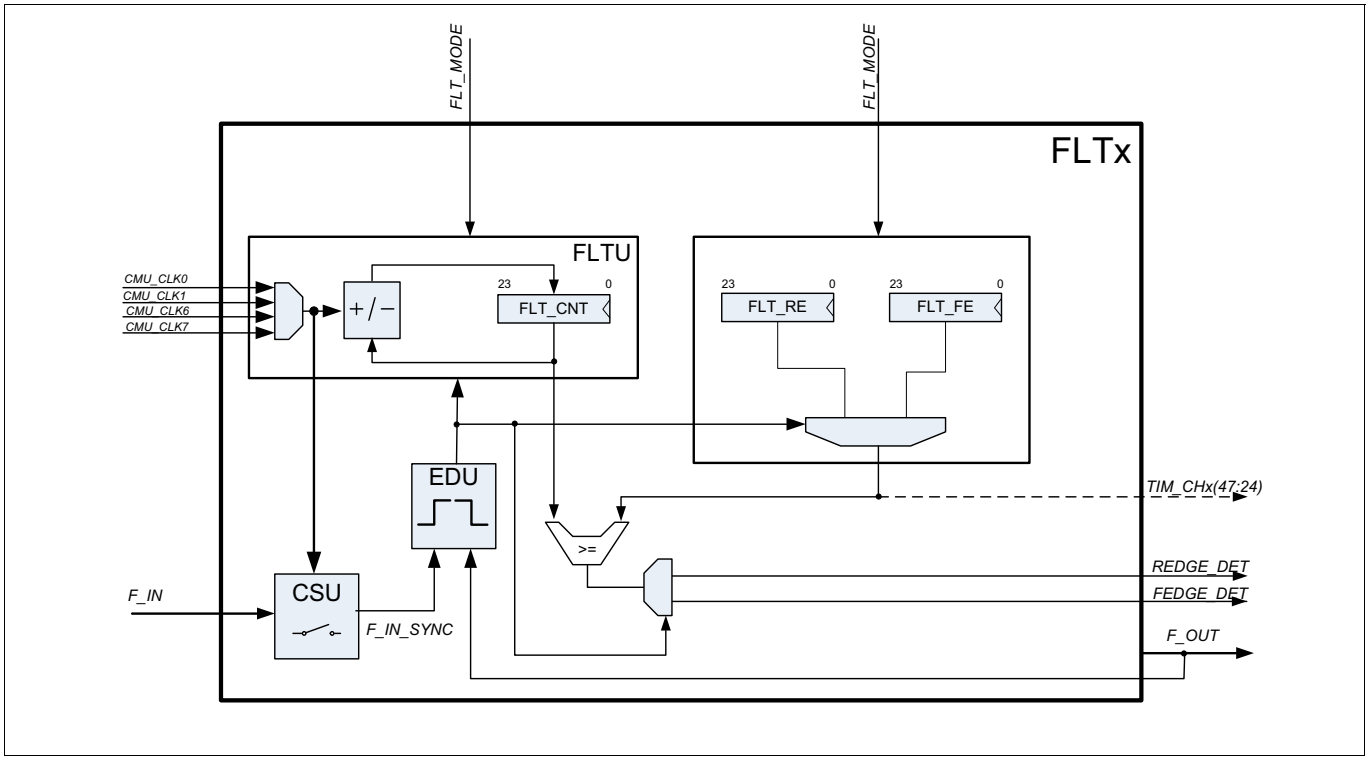


Figure 35 FLT Architecture

The filter parameters (deglitching and acceptance time) for the rising and falling edge can be configured inside the two filter parameter registers **FLT_RE** (rising edge) and **FLT_FE** (falling edge). The exact meaning of the parameter depends on the filter mode.

However the delay time T of both filter parameters **FLT_xE** can always be determined by:

T=(FLT_xE+1)*T_{FLT_CLK},

When a glitch is detected on an input signal a status flag **GLITCHDET** is set inside the **TIM[i]_CH[x]_IRQ_NOTIFY** register.

Table 28 gives an overview about the meanings for the registers **FLT_RE** and **FLT_FE**. In the individual deglitching time modes, the actual filter threshold for a detected regular edge is provided on the *TIM[i]_CH[x](47:24)* output line. In the case of immediate edge propagation mode, a value of zero is provided on the *TIM[i]_CH[x](47:24)* output line.

The *TIM[i]_CH[x](47:24)* output line is used by the MAP sub-module for further processing (please see chapter “TIM0 Input Mapping Module (MAP)”).

Table 28 Filter Parameter summary for the different Filter Modes

Filter mode	Meaning of FLT_RE	Meaning of FLT_FE
Immediate edge propagation	Acceptance time for rising edge	Acceptance time for falling edge
Individual de-glitch time (up/down counter)	De-glitch time for rising edge	De-glitch time for falling edge

Table 28 Filter Parameter summary for the different Filter Modes (cont'd)

Filter mode	Meaning of FLT_RE	Meaning of FLT_FE
Individual de-glitch time (hold counter)	De-glitch time for rising edge	De-glitch time for falling edge
Individual de-glitch time (reset counter)	De-glitch time for rising edge	De-glitch time for falling edge

A counter **FLT_CNT** is used to measure the glitch and acceptance times.

The frequency of the **FLT_CNT** counter is configurable in bit field **FLT_CNT_FRQ** of register **TIM[i]_CH[x]_CTRL**.

The counter **FLT_CNT** can either be clock with the *CMU_CLK0, CMU_CLK1, CMU_CLK6* or the *CMU_CLK7* signal. These signals are coming from the CMU sub-module.

The **FLT_CNT**, **FLT_FE** and **FLT_RE** registers are 24-bit width. For example, when the resolution of the *CMU_CLK0* signal is 50ns this allows maximal de-glitch and acceptance times of about 838ms for the filter.

28.13.2.2 TIM Filter Modes

28.13.2.2.1 Immediate Edge Propagation Mode

In immediate edge propagation mode after detection of an edge the new signal level on *F_IN_SYNC* is propagated to *F_OUT* with a delay of one T_{period} and the new signal level remains unchanged until the configured acceptance time expires.

For each edge type the acceptance time can be specified separately in the **FLT_RE** and **FLT_FE** registers.

Each signal change on the input *F_IN_SYNC* during the duration of the acceptance time has no effect on the output signal level *F_OUT* of the filter but it sets the glitch **GLITCHDET** bit in the **TIM[i]_CH[x]_IRQ_NOTIFY** register.

After it expires an acceptance time the input signal *F_IN_SYNC* is observed and on signal level change the filter raises a new detected edge and the new signal level is propagated to *F_OUT*.

Independent of a signal level change the value of *F_OUT* is always set to *F_IN_SYNC,* when the acceptance time expires (see also **Figure 37**).

Figure 36 shows an example for the immediate edge propagation mode, in the case of rising edge detection. Both, the signal before filtering (*F_IN*) and after filtering (*F_OUT*) are shown. The acceptance time *at1* is specified in the register **FLT_RE**.

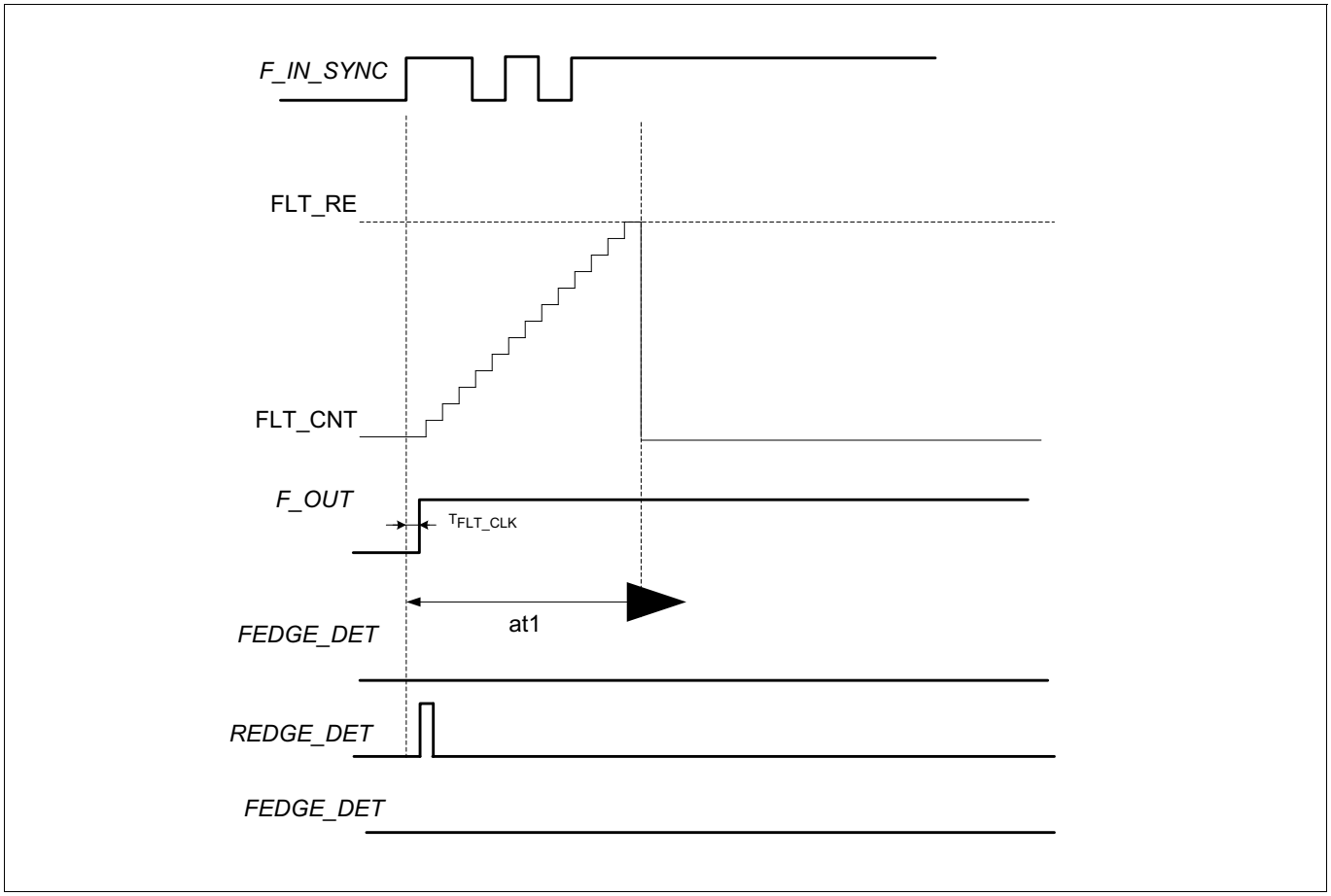


Figure 36 Immediate Edge Propagation Mode in the case of a rising edge

In immediate edge propagation mode the glitch measurement mechanism is not applied to the edge detection. Detected edges on *F_IN_SYNC* are transferred directly to *F_OUT*.

The counter **FLT_CNT** is incremented until acceptance time threshold is reached.

Figure 37 shows a more complex example of the TIM filter, in which both, rising and falling edges are configured in immediate edge propagation mode.

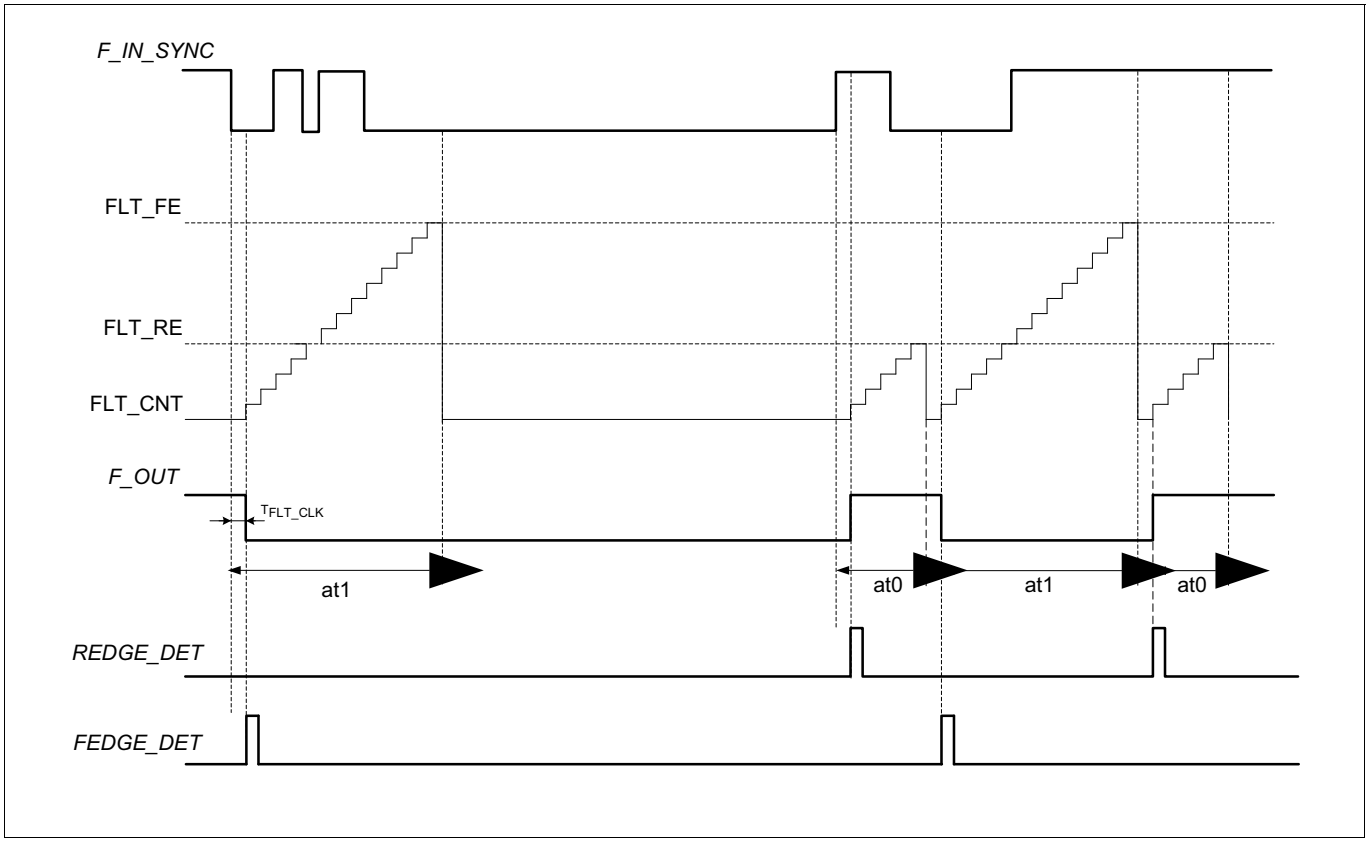


Figure 37 Immediate Edge Propagation Mode in the case of a rising and falling edge

If the **FLT_CNT** has reached the acceptance time for a specific signal edge and the signal *F_IN_SYNC* has already changed to the opposite level of *F_OUT*, the opposite signal level is set to *F_OUT* and the acceptance time measurement is started immediately. **Figure 37** shows this scenario at the detection of the first rising edge and the second falling edge.

28.13.2.2.2 Individual De-glitch Time Mode (up/down counter)

In individual de-glitch time mode (up/down counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers **FLT_RE** and **FLT_FE,** respectively.

The filter counter register **FLT_CNT** is incremented when the signal level on *F_IN_SYNC* is unequal to the signal level on *F_OUT* and decremented if *F_IN_SYNC* equals *F_OUT*.

After **FLT_CNT** has reached a value of zero during decrementing the counter is stopped immediately.

If a glitch is detected a glitch detection bit **GLITCHDET** is set in the **TIM[i]_CH[x]_IRQ_NOTIFY** register.

The detected edge signal together with the new signal level is propagated to *F_OUT* after the individual de-glitch threshold is reached. **Figure 38** shows the behavior of the filter in individual de-glitch time (up/down counter) mode in the case of the rising edge detection.

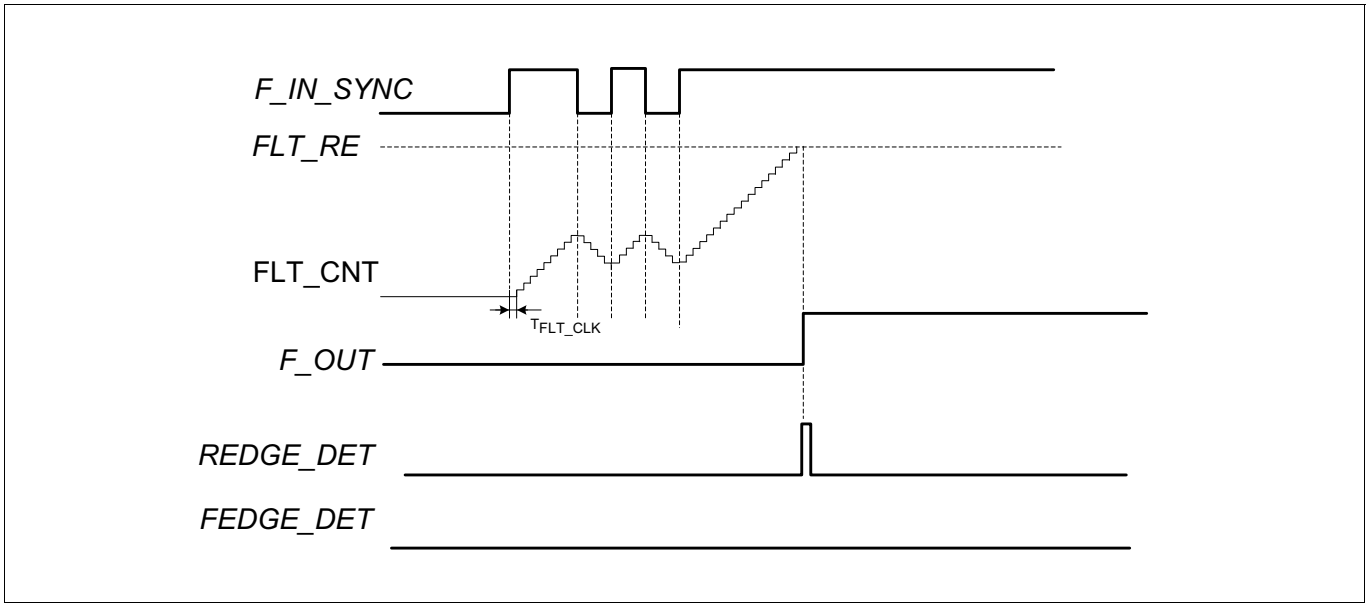


Figure 38 Individual De-glitch Time Mode (up/down counter) in the case of a rising edge

28.13.2.2.3 Individual De-glitch Time Mode (hold counter)

In individual de-glitch time mode (hold counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers **FLT_RE** and **FLT_FE,** respectively.

The filter counter register **FLT_CNT** is incremented when the signal level on *F_IN_SYNC* is unequal to the signal level on *F_OUT* and the counter value of **FLT_CNT** is hold if *F_IN* equals *F_OUT*.

If a glitch is detected the glitch detection bit **GLITCHDET** is set in the **TIM[i]_CH[x]_IRQ_NOTIFY** register.

The detected edge signal together with the new signal level is propagated to *F_OUT* after the individual de-glitch threshold is reached. **Figure 39** shows the behavior of the filter in individual de-glitch time (hold counter) mode in the case of the rising edge detection.

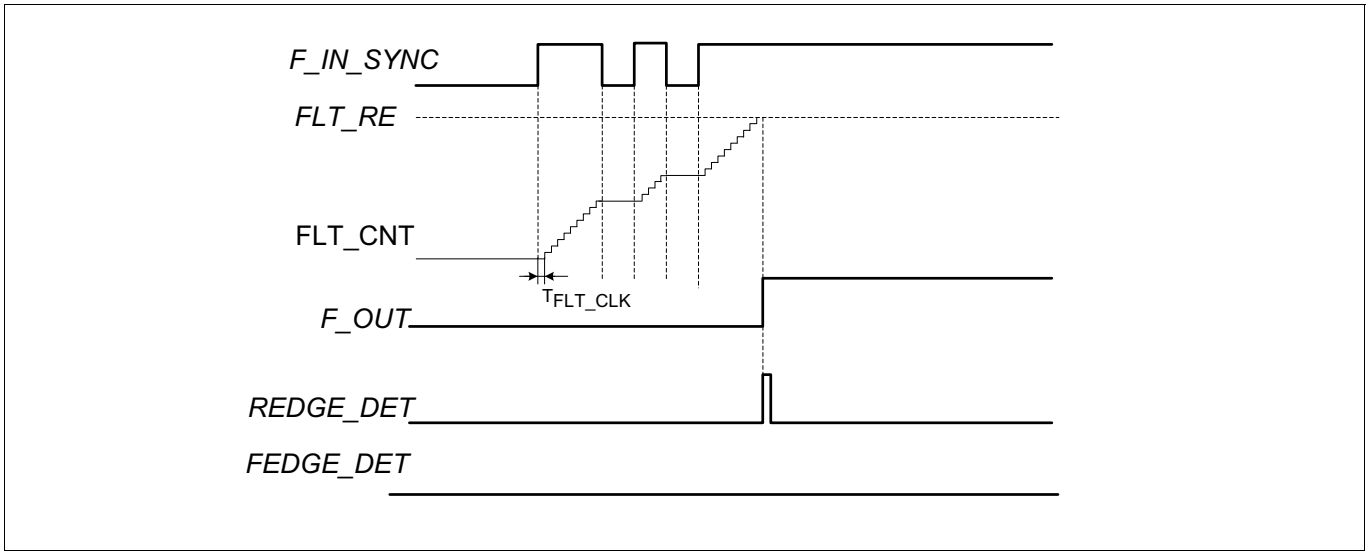


Figure 39 Individual De-glitch Time Mode (hold counter) in the case of a rising edge

28.13.2.2.4 Individual De-glitch Time Mode (reset counter)

In individual de-glitch time mode (reset counter) each edge of an input signal can be filtered with an individual de-glitch threshold filter value mentioned in the registers **FLT_RE** and **FLT_FE,** respectively.

The filter counter register **FLT_CNT** is incremented when the signal level on *F_IN_SYNC* is unequal to the signal level on *F_OUT* and the counter value of **FLT_CNT** is reset to 0x000000 if *F_IN* equals *F_OUT*.

If a glitch is detected the glitch detection bit **GLITCHDET** is set in the **TIM[i]_CH[x]_IRQ_NOTIFY** register.

The detected edge signal together with the new signal level is propagated to *F_OUT* after the individual de-glitch threshold is reached. **Figure 40** shows the behavior of the filter in individual de-glitch time (reset counter) mode in the case of the rising edge detection.

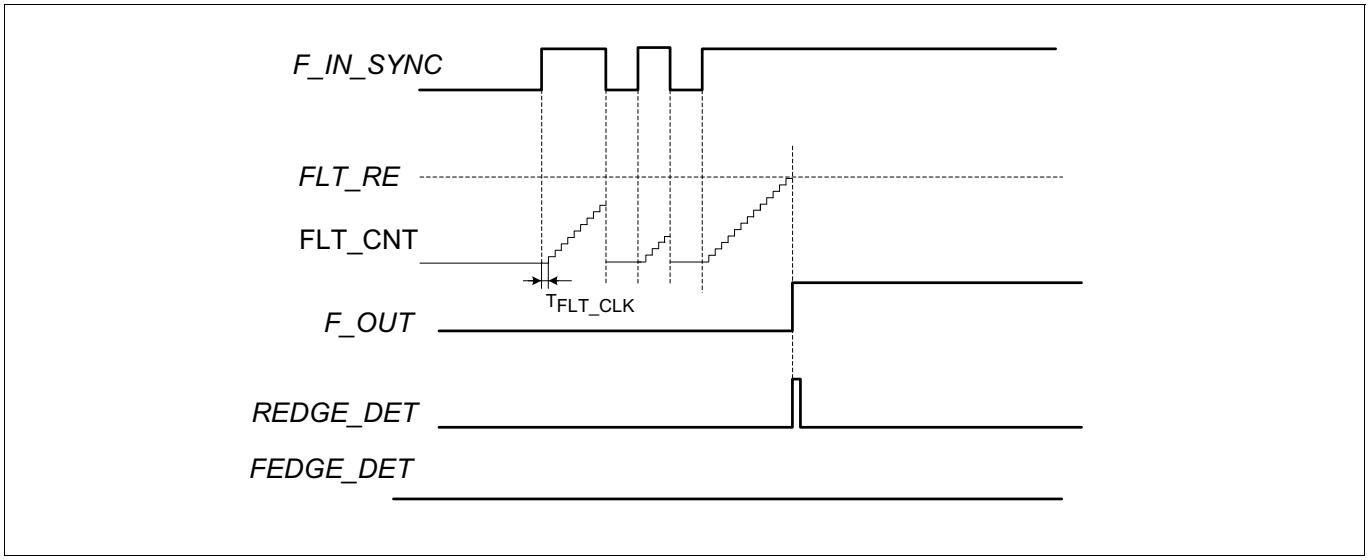


Figure 40 Individual De-glitch Time Mode (reset counter) in the case of a rising edge

28.13.2.2.5 Immediate Edge Propagation and Individual De-glitch Mode

As already mentioned, the four different filter modes can be applied individually to each edge of the measured signal.

However, if one edge is configured with immediate edge propagation and the other edge with an individual deglitching mode (whether up/down counter, hold counter or reset counter) a special consideration has to be applied.

Assume that the rising edge is configured for immediate edge propagation and the falling edge with individual deglitching mode (up/down counter) as shown in **Figure 41**.

If the falling edge of the incoming signal already occurs during the measuring of the acceptance time of the rising edge, the measurement of the deglitching time on the falling edge is started delayed, but immediately after the acceptance time measurement phase of the rising edge has finished.

Consequently, the deglitching counter cannot measure the time T_{ERROR}, as shown in **Figure 41**.

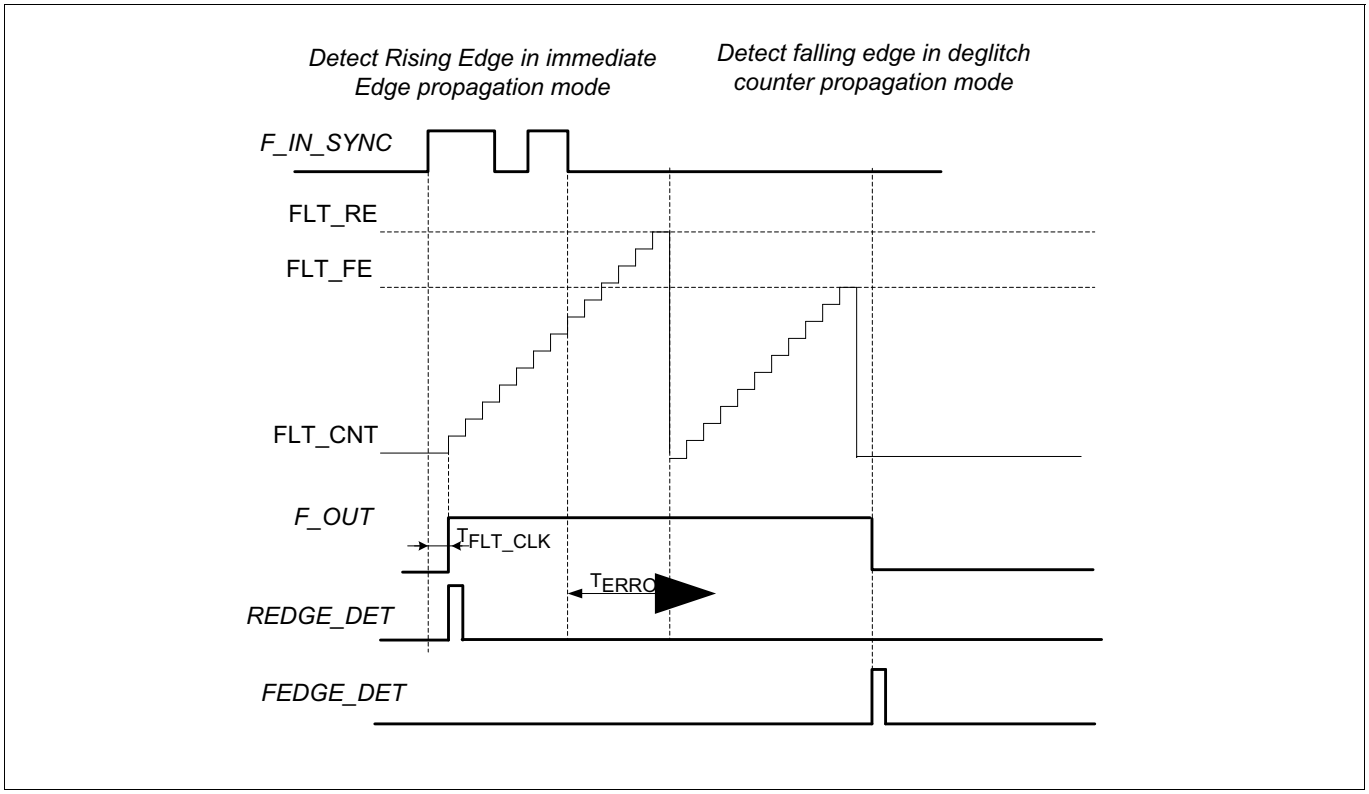


Figure 41 Mixed mode measurement

28.13.2.3 TIM Filter re-configuration

If **FLT_EN**=1 a change of or **FLT_FE** will take place immediately.

If **FLT_EN**=1 a change of **FLT_MODE_RE** or **FLT_MODE_FE** will be used with the next occurring corresponding edge. If the mode is changed while the filter unit is processing a certain mode, it will end this edge filtering in the mode as started.

If **FLT_EN**=1 a change of **FLT_CTR_RE**, **FLT_CTR_FE**, **EFLT_CTR_RE** or **EFLT_CTR_FE** will take place immediately.

28.13.3 Timeout Detection Unit (TDU)

The Timeout Detection Unit (TDU) is responsible for timeout detection of the TIM input signals.

Each channel of the TIM sub-module has its own Timeout Detection Unit (TDU) where a timeout event can be set up on the filtered input signal of the corresponding channel.

In each timeout unit exist 3 8 bit counter/comparator slices. A counter/comparator slice is shown below. The counter TO_CNT will increment by signal *INC*. The counter can be loaded with the value *LOAD_VAL* if *LOAD_VAL*= 1. GT_EVT will be 1 if TO_CNT > TOV is fulfilled. EQ_EVT will be 1 if TO_CNT = TOV is fulfilled.

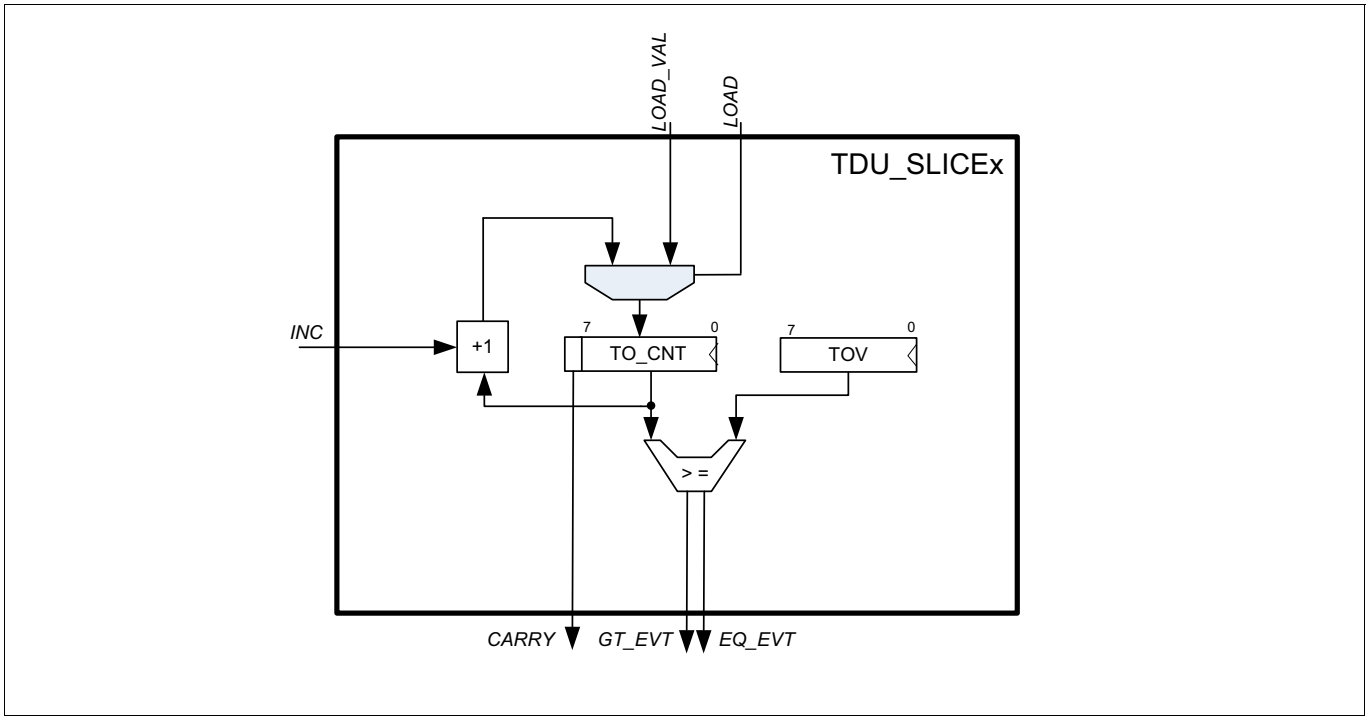


Figure 42 Counter/comparator slice

The counter/comparator slices can be cascaded depending on the application needs to operate as:

- 3x 8-bit counter
- 1x 16-bit counter and 1x 8-bit counter
- 1x 24-bit counter
- 2x 8-bit counter

This allows the user to use the functions:

- timeout on input signals
- local CMU clock prescaler 8 bit
- trigger event generation 8-bit (external capture, todet_irq)

in parallel.

With usage of the 3x 8-bit counter it is possible to define different timeout values for the 2 signal levels.

Following table shows which functions can be used in parallel.

28.13.3.1 Used parallel functions

Table 29 Used parallel functions)

Counter type	Timeout functionality	Generate local TIM CMU clk	Source for external capture to previous channel	Source for TODET_IRQ
24 bit	24 bit	no	tdu_timeout_evt tdu_sample_evt	tdu_timeout_evt tdu_sample_evt
1 x 8 bit 1 x 16 bit	16 bit local clk tdu_sample_evt usable	yes	tdu_timeout_evt, tdu_frame_evt, tdu_sample_evt	tdu_timeout_evt, tdu_frame_evt, tdu_sample_evt
3x 8 bit	8 bit local clk tdu_sample_evt usable	yes	tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt	tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt
3x 8 bit	no	yes	tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt	tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt
2x 8 bit	no	no	tdu_timeout_evt, tdu_word_evt, tdu_frame_evt	tdu_timeout_evt, tdu_word_evt, tdu_frame_evt

Next table shows which of the available 8 bit resources are cascaded with a chosen **SLICING**.

28.13.3.2 Which of the available 8 bit resources are cascaded with a chosen SLICING

Table 30 Which of the available 8 bit resources are cascaded with a chosen SLICING

Counter type	Counters count on	Counter resource generates	CLK selection
24 bit	CNT on TCS	CNT= TO_CNT2 & TO_CNT1 & TO_CNT; TCMP = TOV2 & TOV1 & TOV; CNT >= TCMP generates tdu_sample_evt tdu_timeout_evt = tdu_sample_evt tdu_frame_evt = 0 tdu_word_evt = 0	TCS selected
3x 8 bit	TO_CNT2 on TCS TO_CNT on tdu_sample_evt TO_CNT1 on tdu_word_evt	TO_CNT2 >= TOV2 generates tdu_sample_evt TO_CNT >= TOV generates tdu_word_evt TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt	TO_CNT2: TCS selected TO_CNT: tdu_sample_evt selected with TCS_USE_SAMPLE_EVT=1 TO_CNT1: tdu_word_evt selected with TDU_SAME_CNT_CLK=0

Table 30 Which of the available 8 bit resources are cascaded with a chosen SLICING (cont'd)

Counter type	Counters count on	Counter resource generates	CLK selection
3x 8 bit	TO_CNT2 on TCS TO_CNT on tdu_sample_evt TO_CNT1 on tdu_sample_evt	TO_CNT2 >= TOV2 generates tdu_sample_evt TO_CNT >= TOV generates tdu_word_evt TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt or tdu_frame_evt	TO_CNT2: TCS selected TO_CNT: tdu_sample_evt selected with TCS_USE_SAMPLE_EVT=1 TO_CNT1: tdu_sample_evt selected with TDU_SAME_CNT_CLK=1
3x 8 bit	TO_CNT2 on TCS TO_CNT on TCS TO_CNT1 on tdu_word_evt	TO_CNT2 >= TOV2 generates tdu_sample_evt TO_CNT >= TOV generates tdu_word_evt TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt	TO_CNT2: TCS selected TO_CNT: TCS selected with TCS_USE_SAMPLE_EVT=0 TO_CNT1: tdu_word_evt selected with TDU_SAME_CNT_CLK=0
3x 8 bit	TO_CNT2 on TCS TO_CNT on TCS TO_CNT1 on TCS	TO_CNT2 >= TOV2 generates tdu_sample_evt TO_CNT >= TOV generates tdu_word_evt TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt or tdu_frame_evt	TO_CNT2: TCS selected TO_CNT: TCS selected with TCS_USE_SAMPLE_EVT=0 TO_CNT1: TCS selected with TDU_SAME_CNT_CLK=1
2x 8 bit	TO_CNT on TCS TO_CNT1 on tdu_word_evt	TO_CNT >= TOV generates tdu_word_evt TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt tdu_sample_evt = 0	TO_CNT: TCS selected TO_CNT1: tdu_word_evt selected with TDU_SAME_CNT_CLK=0
2x 8 bit	TO_CNT on TCS TO_CNT1 on TCS	TO_CNT >= TOV generates tdu_word_evt TO_CNT1 >= TOV1 generates tdu_frame_evt tdu_timeout_evt = tdu_word_evt tdu_sample_evt = 0	TO_CNT: TCS selected TO_CNT1: TCS selected with TDU_SAME_CNT_CLK=1

Table 30 Which of the available 8 bit resources are cascaded with a chosen SLICING (cont'd)

Counter type	Counters count on	Counter resource generates	CLK selection
1 x 8 bit 1 x 16 bit	TO_CNT2 on TCS CNT on TCS	TO_CNT2 >= TOV2 generates tdu_sample_evt CNT = TO_CNT1 & TO_CNT; TCMP = TOV1 & TOV; CNT >= TCMP generates tdu_frame_evt tdu_timeout_evt = tdu_frame_evt tdu_word_evt = 0	TO_CNT2: TCS selected CNT: TCS selected with TCS_USE_SAMPLE_EVT=0
1 x 8 bit 1 x 16 bit	TO_CNT2 on TCS CNT on tdu_sample_evt	[CDATA]TO_CNT2 >= TOV2 generates tdu_sample_evt CNT = TO_CNT1 & TO_CNT; TCMP = TOV1 & TOV; CNT >= TCMP generates tdu_frame_evt tdu_timeout_evt = tdu_frame_evt tdu_word_evt = 0	TO_CNT2: TCS selected CNT: tdu_sample_evt selected with TCS_USE_SAMPLE_EVT=1

Based on a chosen counter configuration by **SLICING** it is possible to control the start behavior of the counters by **TDU_START** in multiple ways. In addition the stopping of the counters can be controlled by **TDU_STOP**. Depending on the application needs it can be decided how the individual counter slices can be reset/reloaded by the configuration field **TDU_RESYNC**.

Depending on the counter configuration, up to 4 internal compare events tdu_timeout_evt, tdu_sample_evt, tdu_word_evt, tdu_frame_evt out of the 3 comparator slices can be generated. It can be chosen by **TODET_IRQ_SRC** which shall be used as *TIM_TODETx_IRQ* signal which will be accessible by the **TODET** bit inside the **TIM[i]_CH[x]_IRQ_NOTIFY** register

The TDU architecture is shown in **Figure 43**.

28.13.3.3 Architecture of the TDU Sub-unit

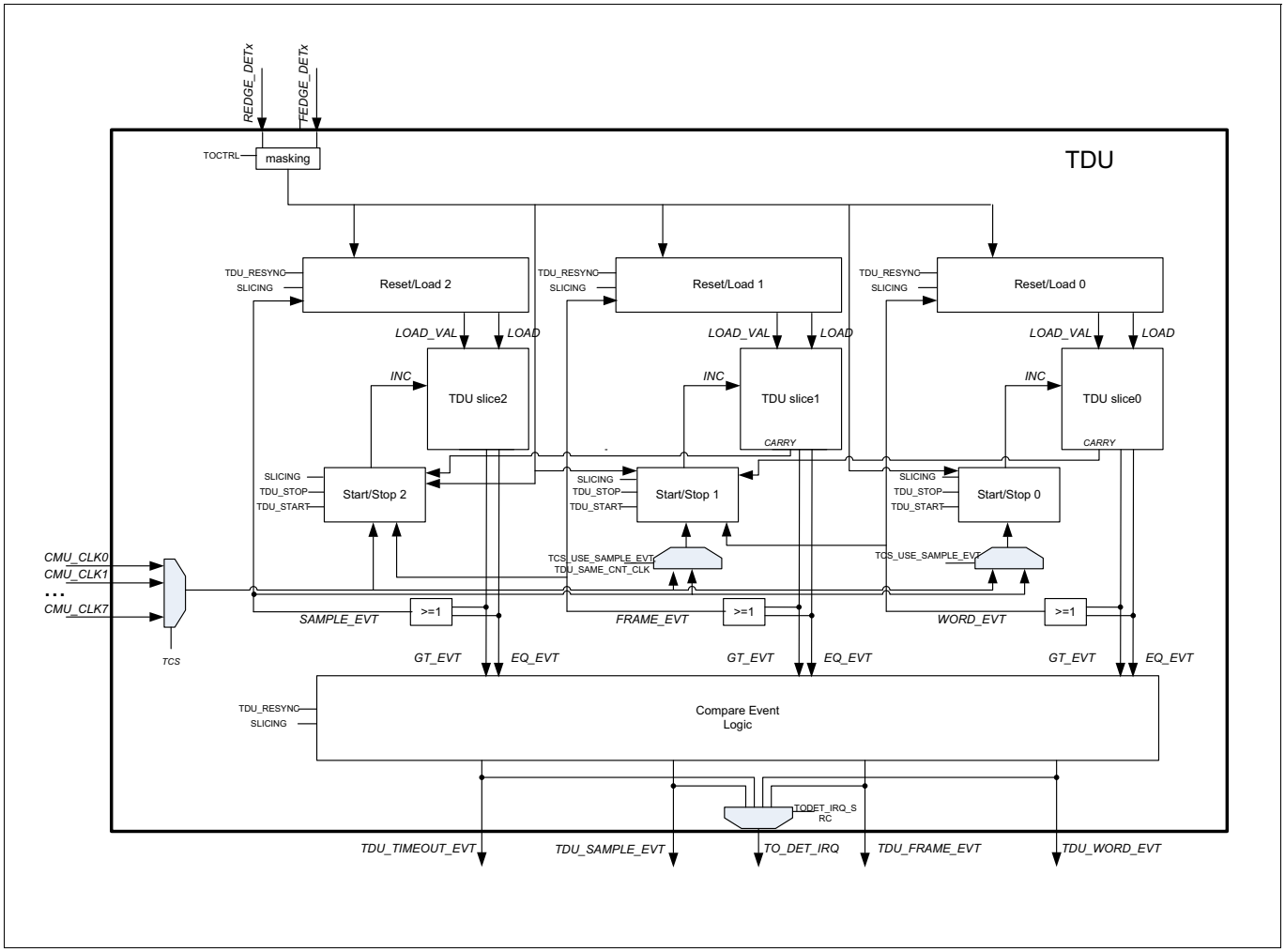


Figure 43 Architecture of the TDU Sub-unit

Each TDU_slice has its own start/stop control, based on the chosen configuration it will decide if the counter inside the TDU slice will increment on the resolution of the applied clock/event. The reset/load control decides based on the configuration settings and the compare result of the TDU slices those the counters **TO_CNT**,**TO_CNT1**,**TO_CNT2** have to be reloaded. Depending on the chosen counter/compare configuration the compare event logic will generate based on the compare results of the 3 TDU slices and the chosen resolution the events *tdu_sample_evt*, *tdu_word_evt*, *tdu_frame_evt*.

The primary resolution on which the TDU is working can be specified with the bit field **TCS** of the register **TIM[i]_CH[x]_TDUV**. The corresponding input signal *CMU_CLKx* will be used to clock the TDU. The individual timeout/counter values have to be specified in number of ticks of the selected input clock signal in the fields **TOV**, **TOV1**, **TOV2** of the timeout value register **TIM[i]_CH[x]_TDUV** of the TIM channel x.

In case of cascading the bit slices by usage of **SLICING** and **TCS_USE_SAMPLE_EVT** and **TDU_SAME_CLK** the resolution for counting can be switched to the events tdu_sample_evt or tdu_word_evt. More details see table above.

The counter compare units start operation on occurrence of the first "start event" configured by **TDU_START**. They continue their operation until the first "stop event" configured by **TDU_STOP** occurs.

In case of occurrence of a start event and a compare/count resolution event in the same clock cycle, the counters will increment or reload/reset based on **TDU_RESYNC** immediately. No *tdu_sample_evt*, *tdu_word_evt*, *tdu_frame_evt* will be generated.

In case of occurrence of a stop event the counters will not change their values. In case of occurrence of a stop event and a compare/count resolution event in the same clock cycle the corresponding events *tdu_sample_evt*, *tdu_word_evt*, *tdu_frame_evt* will be generated.

In case of occurrence of a start event and a stop event in the same clock cycle the counters will not change their values. No *tdu_sample_evt*, *tdu_word_evt*, *tdu_frame_evt* will be generated.

The function of the timeout unit (configured to **TDU_RESYNC**=0000,**TDU_START**=000) can be started or stopped inside the **TIM[i]_CH[x]_CTRL** register by setting/resetting the **TOCTRL** bit.

Timeout detection can be enabled to be sensitive to falling, rising or both edges of the input signal by writing the corresponding values to the bit field **TOCTRL**.

The TDU generates an interrupt signal *TIM_TODETx_IRQ* whenever a timeout is detected for an individual input signal, and the **TODET** bit is set inside the **TIM[i]_CH[x]_IRQ_NOTIFY** register.

In addition, when the ARU access is enabled with the **ARU_EN** bit inside the **TIM[i]_CH[x]_CTRL** register, the actual values stored inside the registers **TIM[i]_CH[x]_GPR0** and **TIM[i]_CH[x]_GPR1** are sent together with the last stored signal level to the ARU if a timeout event *TDU_TIMEOUT_EVT* occurs.

To signal that a timeout occurred, the ARU_OUT(50) bit (ACB(2)) is set. The bit ACB(0) will be updated with the timeout event to the signal level on which the timeout was detected. Timeout signaling with ACB(2) is only possible with **TODET_IRQ_SRC**= 0000.

Thus, a destination could determine if a timeout occurred at the TIM input by evaluating ACB bit 2.

Since the TIM channel still monitors its input pin although the timeout happened, a valid edge could occur at the input pin while the timeout information is still valid at the ARU. In that case, the new edge associated data is stored inside the registers **TIM[i]_CH[x]_GPR0** and **TIM[i]_CH[x]_GPR1**, the GPR overflow detected bit is set together in the ACB field (ACB(1)) with the timeout bit (ACB(2)) and the values are marked as valid to the ARU.

The ACB bit 2 is cleared, when a successful ARU write access by the TIM channel took place.

The ACB bit 1 is cleared, when a successful ARU write access by the TIM channel took place.

When a valid edge initiates an ARU write access which has not ended while a new timeout occurs the GPR overflow detected bit (ACB(1)) is set. The bit ACB(0) will be updated to the level on which the timeout occurred.

When a timeout occurred and initiates an ARU write access which has not ended while a new timeout occurs the GPR overflow detected bit (ACB(1)) is not set.

The following table clarifies the meaning of the ACB Bits for valid data provided by a TIM channel:

Table 31 ACB Bits for valid data provided by a TIM channel

ACB4/3	ACB2	ACB1	ACB0	Description
dc	0	0	SL	Valid edge detected
dc	0	1	SL	Input edge overwritten by subsequent edge
dc	1	0	SL	Timeout detected without valid edge
dc	1	1	SL	Timeout detected with subsequent valid edge detected

28.13.4 TIM Channel Architecture

28.13.4.1 Overview

Each TIM channel consist of an input edge counter **ECNT**, a Signal Measurement Unit (SMU) with a counter **CNT**, a counter shadow register **CNTS** for SMU counter and two general purpose registers **GPR0** and **GPR1** for value storage.

The value **TOV** of the timeout register **TIM[i]_CH[x]_TDU** is provided to TDU sub-unit of each individual channel for timeout measurement. The architecture of the TIM channel is depicted in **Figure 44**.

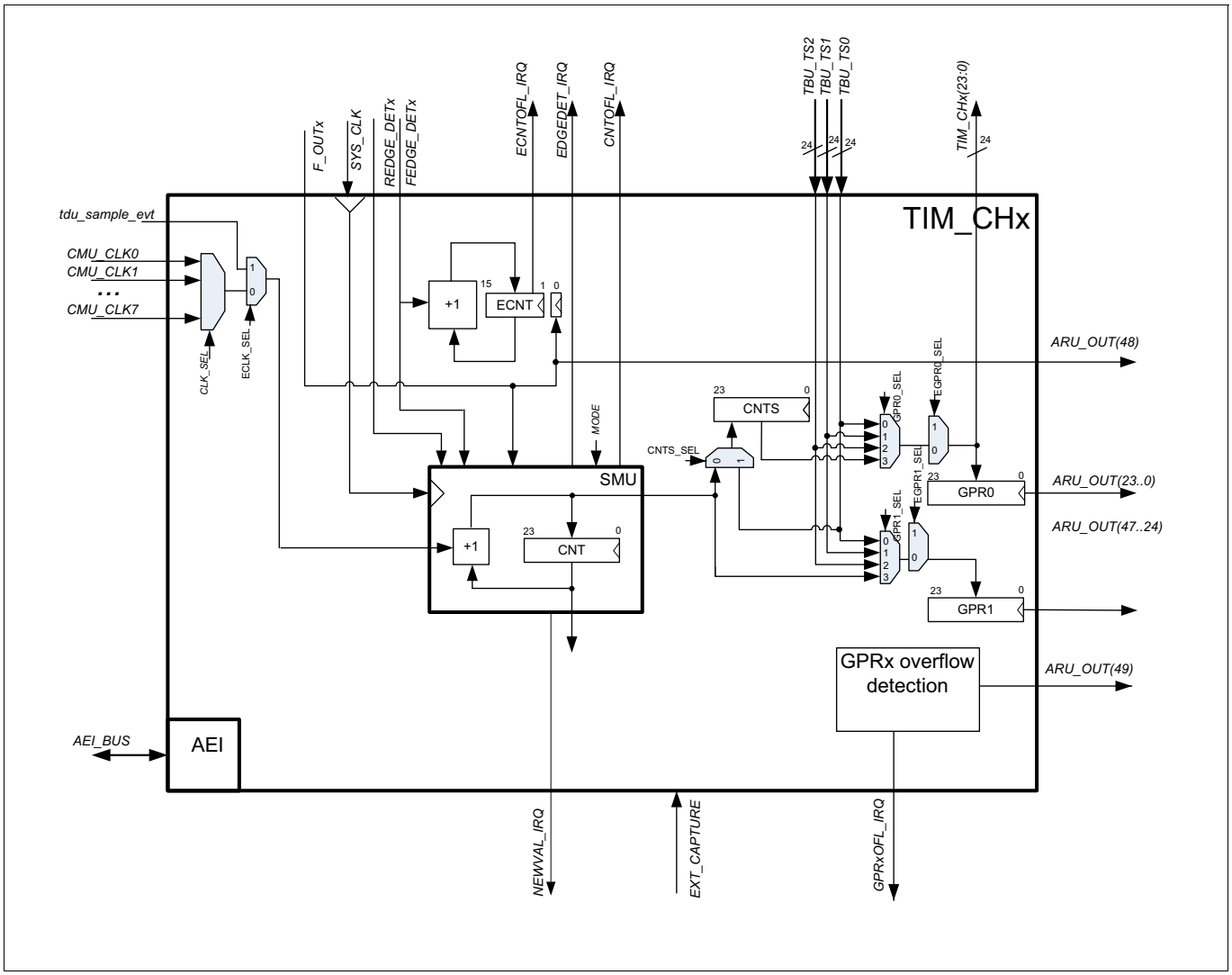


Figure 44 TIM Channel Architecture

Each TIM channel receives both input trigger signals *REDGE_DETx* and *FEDGE_DETx,* generated by the corresponding filter module in order to signalize a detected echo of the input signal *F_INx*. The signal *F_OUTx* shows the filtered signal of the channel's input signal *F_INx*.

The edge counter **ECNT** counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of **ECNT**. (However, the actual counter implementation counts only falling edges on ECNT[n:1] bits. It generates **ECNT** by composing the ECNT[n:1] bits with F_OUTx as bit 0).

Thus, the whole ECNT counter value is always odd, when a positive edge was received and always even, when a negative edge was received.

The current **ECNT[7:0]** register content is made visible on the bits 31 down to 24 of the registers **GPR0**, **GPR1,** and **CNTS**. This allows the software to detect inconsistent read accesses to registers **GPR0**, **GPR1**, and **CNTS**. However, the update strategy of these registers depends on the selected TIM modes, and thus the consistency check has to be adapted carefully.

It can be chosen with the bit field **FR_ECNT_OFL** when an **ECNT** overflow is signaled on **ECNTOFL**. An ECNT overflow can be signaled on 8 bit or full range resolution.

While reading the register **TIM[i]_CH[x]_ECNT** the bit **ECNT[0]** shows the input signal value F_OUTx independent of the state (enabled / disabled) of the channel. If a channel gets disabled (OSM mode or resetting TIM_EN) the content of **TIM[i]_CH[x]_ECNT** will be frozen until a read of the register takes place. This read will reset the **ECNT** counter. Continuing reads will show the input signal value in bit **ECNT[0]** again.

When new data is written into **GPR0** and **GPR1** the **NEWVAL** bit is set in **TIM[i]_CH[x]_IRQ_NOTIFY** register and depending on corresponding enable bit value the *NEWVALx_IRQ* interrupt is raised.

Each TIM input channel has an ARU connection for providing data via the ARU to the other GTM sub-modules. The data provided to the ARU depends on the TIM channel mode and its corresponding adjustments (e.g. multiplexer configuration).

The bit **ARU_EN** of register **TIM[i]_CH[x]_CTRL** decides, whether the measurement results of registers **GPR0** and **GPR1** are consumed by another sub-module via ARU (**ARU_EN** = 1) or the CPU via AEI (**ARU_EN** = 0).

To guarantee a consistent delivery of data from the **GPR0** and **GPR1** registers to the ARU or the CPU each TIM channel has to ensure that the data is consumed before it is overwritten with new values.

If new data was produced by the TIM channel (bit **NEWVAL** is set inside **TIM[i]_CH[x]_IRQ_NOTIFY** register) while the old data is not consumed by the ARU (**ARU_EN** = 1) or CPU (**ARU_EN** = 0), the TIM channel sets the **GPROFL** bit inside the status register **TIM[i]_CH[x]_IRQ_NOTIFY** and it overwrites the data inside the registers **GPR0** and **GPR1**. In addition when **ARU_EN**=1 the bit ACB(1) is set to 1 to indicate the overflow in the ARU data.

If the CPU is selected as consumer for the registers **GPR0** and **GPR1** (**ARU_EN** = 0), the acknowledge for reading out data is performed by a read access to the register **GPR0**. Thus, register **GPR1** should be read always before **GPR0**.

If the ARU is selected as consumer for the registers **GPR0** and **GPR1** (**ARU_EN** = 1), the acknowledge for reading out data is performed by the ARU itself. However, the registers **GPR0** and **GPR1** could be read by CPU without giving an acknowledge.

28.13.4.2 TIM Channel Modes

The TIM provides seven different measurement modes that can be configured with the bit field **TIM_MODE** of register **TIM[i]_CH[x]_CTRL**. The measurement modes are described in the following subsections. Besides these different basic measurement modes, there exist distinct configuration bits in the register **TIM[i]_CH[x]_CTRL** for a more detailed controlling of each mode. The meanings of these bits are as follows:

- **DSL**: control the signal level for the measurement modes (e.g. if a measurement is started with rising edge or falling edge, or if high level pulses or low level pulses are measured.
- **EGPR0_SEL, GPR0_SEL** and **EGPR1_SEL, GPR1_SEL**: control the actual content of the registers **GPR0** and **GPR1** after a measurement has finished.
- **CNTS_SEL**: control the content of the registers **CNTS.** The actual time for updating the **CNTS** register is mode dependent.
- **OSM**: activate measurement in one-shot mode or continuous mode. In one-shot mode only one measurement cycle is performed and after that the channel is disabled.
- **NEWVAL**: The NEWVAL IRQ interrupt is triggered at the end of a measurement cycle, signaling that the registers GPR0 and GPR1 are updated.
- **ARU_EN**: enables sending of the registers **GPR0** and **GPR1** together with the actual signal level (in bit 48) and the overflow signal GPROFL (in bit 49), and the timeout status information (bit 50) to the ARU.
- **EXT_CAP_EN**: forces an update of the registers **GPR0** and **GPR1** and **CNTS** (TIM channel mode dependent) only on each rising edge of the EXT_CAPTURE signal and triggers a NEWVAL IRQ interrupt. If this mode is disabled the NEWVAL IRQ interrupt is triggered at the end of each measurement cycle.

For each channel the source of the EXT_CAPTURE signal can be configured with the bit fields **EXT_CAP_SRC** in the register **TIM[i]_CH[x]_ECTRL.**

28.13.4.2.1 TIM PWM Measurement Mode (TPWM)

In TIM PWM Measurement Mode the TIM channel measures duty cycle and period of an incoming PWM signal. The **DSL** bit defines the polarity of the PWM signal to be measured.

When measurement of pulse high time and period is requested (PWM with a high level duty cycle, **DSL**=1) and **IMM_START**=0, the channel starts measuring after the first rising edge is detected by the filter.

If **IMM_START**=1 the measurement starts immediately after activating the channel by **TIM_EN**=1.

Measurement is done with the **CNT** register counting with the configured clock coming from **CMU_CLKx** until a falling edge is detected.

Assume: **SWAP_CAPTURE**=0, **ECNT_RESET**=0

Then the counter value is stored inside the shadow register **CNTS** (if **CNTS_SEL** = 0) and the counter **CNT** counts continuously until the next rising edge is reached.

On this following rising edge the content of the **CNTS** register is transferred to **GPR0** and the content of **CNT** register is transferred to **GPR1**, assuming settings for the selectors **EGPR0_SEL**=0,**GPR0_SEL**=11 and **EGPR1_SEL**=0,**GPR1_SEL**=11. By this, **GPR0** contains the duty cycle length and **GPR1** contains the period. It should be noted, that the bits 1 to 7 of the **ECNT** may be used to check data consistency of the registers **GPR0** and **GPR1**.

In addition the **CNT** register is cleared **NEWVAL** status bit inside of **TIM[i]_CH[x]_IRQ_NOTIFY** status register and depending on corresponding interrupt enable condition *TIM_NEWVALx_IRQ* interrupt is raised.

The CNTS register update is not performed until the measurement is started. Afterwards each edge leaving the level defined by DSL is performing a CNTS register update.

If a PWM with a low level duty cycle should be measured (**DSL** = 0) and **IMM_START**=0, the channel waits for a falling edge until measurement is started. On this edge the low level duty cycle time is stored first in **CNTS** and then finally in **GPR0** and the period is stored in **GPR1**.

When a PWM period was successfully measured, the data in the registers **GPR0** and **GPR1** is marked as valid for reading by the ARU when the **ARU_EN** bit is set inside **TIM[i]_CH[x]_CTRL** register, the **NEWVAL** bit is set inside the **TIM[i]_CH[x]_IRQ_NOTIFY** register, and a new measurement is started.

If the preceding PWM values were not consumed by a reader attached to the ARU (**ARU_EN** bit enabled) or by the CPU the TIM channel set **GPROFL** status bit in **TIM[i]_CH[x]_IRQ_NOTIFY** and depending on corresponding interrupt enable bit value raises a *GPROFL_IRQ* and overwrites the old values in **GPR0** and **GPR1**. A new measurement is started afterwards.

If the register **CNT** produces an overflow during the measurement, the bit **CNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_CNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_ECNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

If **ECNT_RESET**=0 the counter **CNT** will be reset to 0 on active edge (defined by **DSL**) of the input signal. If **ECNT_RESET**=1 the counter **CNT** will be reset to 0 on each edge of the input signal.

Assume **EXT_CAP_EN**=0 and **SWAP_CAPTURE**=0:On every input edge to the active level defined by **DSL** will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1**. Every edge to the inactive level will capture the data selected by **CNTS_SEL** to the registers **CNTS**.

Assume **EXT_CAP_EN**=0 and **SWAP_CAPTURE**=1:On every input edge to the inactive level defined by **DSL** will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1**. Every edge to the active level will capture the data selected by **CNTS_SEL** to the registers **CNTS**.

External capture TIM PWM Measurement Mode (TPWM)

If external capture is enabled **EXT_CAP_EN**=1, the PWM measurement is done continuously. The actual measurement values are captured to GPRx if an external capture event occurs.

On every external capture event the data selected by **CNTS_SEL, EGPR0_SEL**, **GPR0_SEL** will be captured to the registers **CNTS, GPR0.**

If **SWAP_CAPTURE**=0 every external capture event will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1**. Every input edge to the level != **DSL** will capture the data selected by **CNTS_SEL** to the registers **CNTS**.

If **SWAP_CAPTURE**=1 every input edge to the inactive level != **DSL** will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1**.

Assume **SWAP_CAPTURE**=0:

Operation is done depending on CMU clock, **ISL, DSL** bit and the input signal value defined in next table (Assume CNTS_SEL= 0):

Table 32 Operation depending on CMU clock, ISL, DSL and the input signal value (Assume CNTS_SEL= 0)

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
0	1	0	-	0	CNT++
1	1	0	-	0	no
rising edge	-	0	0	0	capture CNT value in CNTS
falling edge	-	0	0	0	CNT=0
rising edge	-	0	1	0	no
falling edge	-	0	1	0	capture CNT value in CNTS; CNT=0
1	1	0	-	1	CNT++
0	1	0	-	1	no
falling edge	-	0	0	1	capture CNT value in CNTS
rising edge	-	0	0	1	CNT=0
falling edge	-	0	1	1	no
rising edge	-	0	1	1	capture CNT value in CNTS; CNT=0
-	-	rising edge	-	-	do GPRx capture; issue NEWVAL_IRQ
-	0	0	-	-	no

The CNTS register update is not performed until the measurement is started (first edge defined by DSL is detected). Afterwards the update of the CNTS register is defined by ISL,DSL combinations in the table above.

28.13.4.2.2 TIM Pulse Integration Mode (TPIM)

In TIM Pulse Integration Mode each TIM channel is able to measure a sum of pulse high or low times on an input signal, depending on the selected signal level bit **DSL** of register **TIM[i]_CH[x]_CTRL** register.

If **IMM_START**=0 the pulse integration measurement is started with occurrence of the first edge defined by DSL on the input signal. If **IMM_START**=1 the measurement starts immediately after activating the channel by **TIM_EN**=1.

The pulse times are measured by incrementing the TIM channel counter **CNT** until the counter is stopped with occurrence of a input signal edge to the opposite signal level defined by **DSL**.

The counter **CNT** counts with the *CMU_CLKx* clock specified by the *CLK_SEL* bit field of the **TIM[i]_CH[x]_CTRL** register.

The **CNT** register is reset at the time the channel is activated (enabling via AEI write access) and it accumulates pulses while the channel is staying enabled.

Assume **EXT_CAP_EN**=0 and **SWAP_CAPTURE**=0:After measurement is started, every falling(DSL=1) or rising(DSL=0) input edge will issue a *TIM_NEWVALx_IRQ* interrupt, and the registers **CNTS**, **GPR0** and **GPR1** are updated according to settings of its corresponding input multiplexers, using the bits **EGPR0_SEL**, **EGPR1_SEL**, **GPR0_SEL**, **GPR1_SEL** and **CNTS_SEL**. It should be noted, that the bits 1 to 7 of the **ECNT** may be used to check data consistency of the registers **GPR0** and **GPR1**.

Assume **EXT_CAP_EN**=0 and **SWAP_CAPTURE**=1:After measurement is started, every falling(DSL=1) or rising(DSL=0) input edge will issue a *TIM_NEWVALx_IRQ* interrupt, and the registers **CNTS**, **GPR0** are updated according to settings of its corresponding input multiplexers, using the bits **EGPR0_SEL**, **GPR0_SEL** and **CNTS_SEL**.

Every input edge to active level defined by **DSL** (rising DSL=1;falling DSL=0) will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1**.

When the **ARU_EN** bit is set inside the **TIM[i]_CH[x]_CTRL** register the measurement results of the registers **GPR0** and **GPR1** can be send to subsequent sub-modules attached to the ARU.

External capture TIM Pulse Integration Mode (TPIM)

If external capture is enabled **EXT_CAP_EN**=1, the pulse integration is done until next external capture event occurs.

On every external capture event the data selected by **CNTS_SEL, EGPR0_SEL**, **GPR0_SEL** will be captured to the registers **CNTS, GPR0.**

If **SWAP_CAPTURE**=0 every external capture event will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1**.

If **SWAP_CAPTURE**=1 every input edge to the inactive level != **DSL** will capture the data selected by **EGPR1_SEL**, **GPR1_SEL** to the registers **GPR1**.

Assume **SWAP_CAPTURE**=0; **IMM_START**=0:

Operation is done depending on CMU clock, **DSL** bit and the input signal value defined in next table (inc_cnt = false if TIM channel is enabled):

Table 33 Operation depending on CMU clock, DSL and the input signal value (inc_cnt = false if TIM channel is enabled)

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
falling edge	-	0	-	0	inc_cnt = true
rising edge	-	0	-	0	if inc_cnt == true then {do capture GPRx, CNTS; issue NEWVAL_IRQ} inc_cnt = false
rising edge	-	0	-	1	inc_cnt = true
falling edge	-	0	-	1	if inc_cnt == true then {do capture GPRx, CNTS; issue NEWVAL_IRQ} inc_cnt = false

Table 33 Operation depending on CMU clock, DSL and the input signal value (inc_cnt = false if TIM channel is enabled) (cont'd)

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
-	1	0	-	-	if inc_cnt == true then CNT++;
-	-	rising edge	-	-	do capture GPRx, CNTS; issue NEWVAL_IRQ; CNT=0
-	0	0	-	-	no

28.13.4.2.3 TIM Input Event Mode (TIEM)

In TIM Input Event Mode the TIM channel is able to count edges.

It is configurable if rising, falling or both edges should be counted. This can be done with the bit fields **DSL** and **ISL** in **TIM[i]_CH[x]_CTRL** register.

In addition, a *TIM[i]_NEWVAL[x]_IRQ* interrupt is raised when the configured edge was received and this interrupt was enabled.

The counter register **CNT** is used to count the number of edges, and the bit fields **EGPR0_SE**L, **EGPR1_SE**L,**GPR0_SEL**, **GPR1_SEL**, and **CNTS_SEL** can be used to configure the desired update values for the registers **GPR0**, **GPR1** and **CNTS**. These register are updated whenever the edge counter **CNT** is incremented due to the arrival of a desired edge.

If the preceding data was not consumed by a reader attached to the ARU or by the CPU the TIM channel sets **GPROFL** status bit and raises a *GPROFL[x]_IRQ* if it was enabled in **TIM[i]_CH[x]_IRQ_EN** register and overwrites the old values in **GPR0** and **GPR1** with the new ones.

If the register **CNT** produces an overflow during the measurement, the bit **CNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_CNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_ECNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

The TIM Input Event Mode does not depend on the bit field **CLK_SEL** of register **TIM[i]_CH[x]_CTRL**.

External capture TIM Input Event Mode (TIEM)

If external capture is enabled, capturing is done depending on the **DSL**, **ISL** bit and the input signal value defined in next table:

Table 34 Capturing depended on the DSL, ISL and the input signal value, if external capture is enabled

Input signal F_OUTx	External capture	ISL	DSL	Action description
-	rising edge	1	-	do capture; issue NEWVAL_IRQ; CNT++
-	0	1	-	no
1	rising edge	0	1	do capture; issue NEWVAL_IRQ; CNT++
0	-	0	1	no
0	rising edge	0	0	do capture; issue NEWVAL_IRQ; CNT++
1	-	0	0	no

28.13.4.2.4 TIM Input Prescaler Mode (TIPM)

In the TIM Input Prescaler Mode the number of edges which should be detected before a *TIM[i]_NEWVAL[x]_IRQ* is raised is programmable. In this mode it must be specified in the **CNTS** register after how many edges the interrupt has to be raised.

A value of 0 in **CNTS** means that after one edge an interrupt is raised and a value of 1 means that after two edges an interrupt is raised, and so on.

The edges to be counted can be selected by the bit fields **DSL** and **ISL** of register **TIM[i]_CH[x]_CTRL**.

With each triggered interrupt, the registers **GPR0** and **GPR1** are updated according to bits **EGPR0_SE**L, **EGPR1_SE**L,**GPR0_SEL** and **GPR1_SEL**.

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_ECNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

The TIM Input Prescaler Mode does not depend on the bit field **CLK_SEL** of register **TIM[i]_CH[x]_CTRL**.

External capture TIM Input Prescaler Mode (TIPM)

If external capture is enabled, the external capture events are counted instead of the input signal edges.

Operation is done depending on the external capture signal, **DSL**, **ISL** bit and the input signal value defined in next table:

Table 35 Operation depending on the external capture signal, DSL, ISL and the input signal value

Input signal F_OUTx	External capture	ISL	DSL	Action description
-	rising edge	1	-	if CNT >= CNTS then do capture; issue NEWVAL_IRQ; CNT=0 else CNT++ endif
-	0	1	-	no
1	rising edge	0	1	if CNT >= CNTS then do capture; issue NEWVAL_IRQ; CNT=0 else CNT++ endif
0	-	0	1	no
0	rising edge	0	0	if CNT >= CNTS then do capture; issue NEWVAL_IRQ; CNT=0 else CNT++ endif
1	-	0	0	no

28.13.4.2.5 TIM Bit Compression Mode (TBCM)

The TIM Bit Compression Mode can be used to combine all filtered input signals of a TIM sub-module to a parallel *m* bit data word, which can be routed to the ARU, where *m* is the number of channels available in the TIM sub-module.

Figure 45 gives an overview of the TIM bit compression mode.

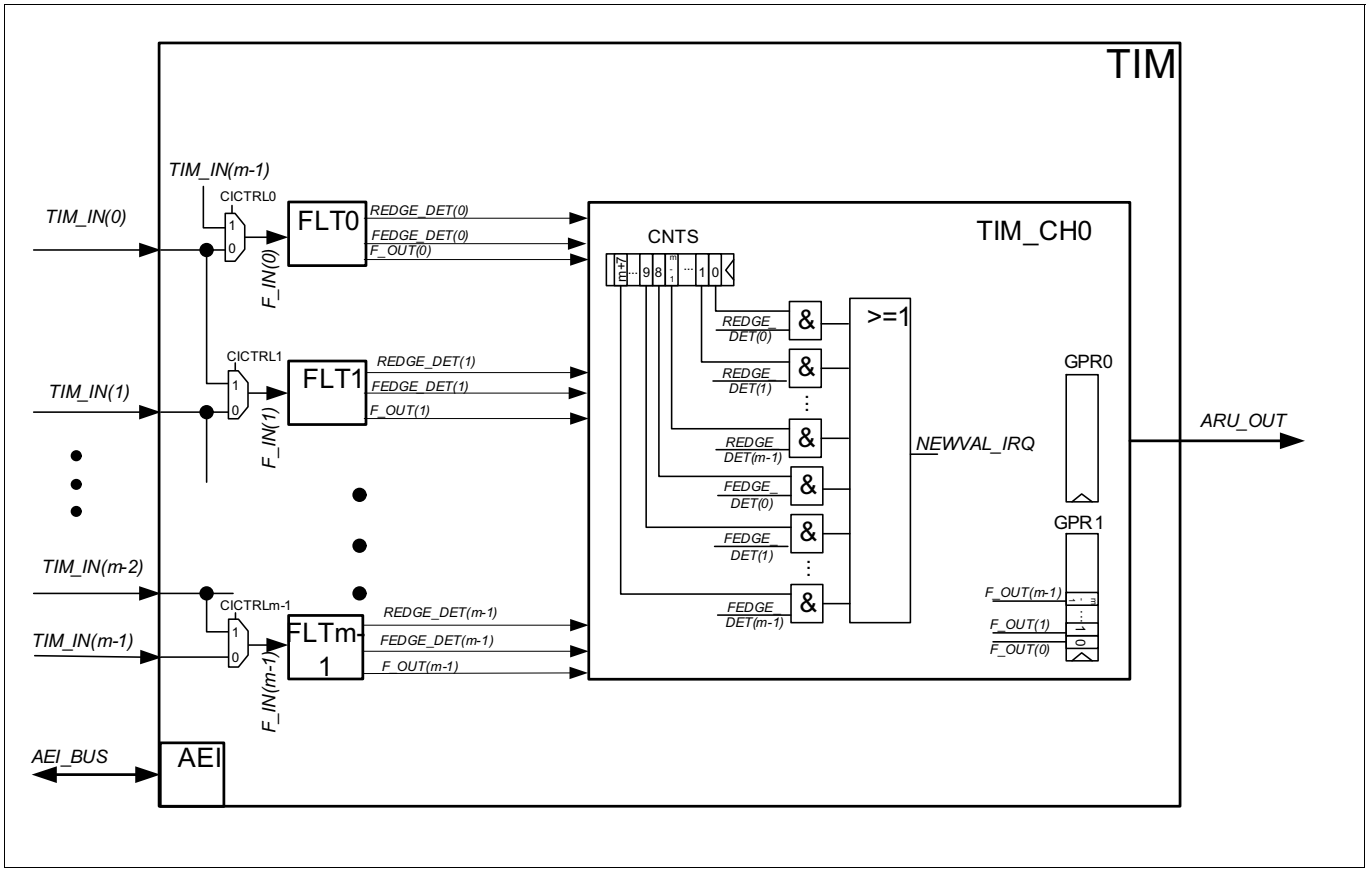


Figure 45 TIM Bit Compression Mode

The register **CNTS** of a channel is used to configure the event that releases the *NEWVAL_IRQ* and samples the input signals F_IN(0) to F_IN(*m*-1) in ascending order as a parallel data word in **GPR1**.

The bits 0 to *m-1* of the **CNTS** register are used to select the *REDGE_DET* signals of the TIM filters 0 to *m-1* as a sampling event, and the bits 8 to (7+*m*) are used to select the *FEDGE_DET* signals of the TIM filters 0 to *m-1*, respectively. If multiple events are selected, the events are OR-combined (see also **Figure 45**).

EGPR0_SEL,**GPR0_SEL** selects the timestamp value, which is routed through the ARU. **GPR1_SEL** is not applicable in TBCM mode.

If the bit **ARU_EN** of register **TIM[i]_CH[x]_CTRL** is set, the sampled data of register **GPR1** is routed together with a time stamp of register **GPR0** to the ARU, whenever the NEWVAL_IRQ is released.

In TIM Bit compression mode, the register **ECNT** increments with each *NEWVAL_IRQ*, which means that the value of ECNT may depend on all *m* input signals. Consequently, the LSB of **ECNT** does not reflect the actual level of the input signal TIM_IN(x).

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_ECNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

The TIM Bit Compression Mode does not depend on the bit field **CLK_SEL** of register **TIM[i]_CH[x]_CTRL**.

External capture Bit Compression Mode (TBCM)

If external capture is enabled, capturing is done depending on the **DSL**, **ISL** bit and the input signal value defined in next table:

Table 36 Capturing depended on the DSL, ISL and the input signal value, if external capture is enabled

Input signal F_OUTx	External capture	ISL	DSL	Action description
-	rising edge	1	-	do capture; issue NEWVAL_IRQ; CNT++
-	0	1	-	no
1	rising edge	0	1	do capture; issue NEWVAL_IRQ; CNT++
0	-	0	1	no
0	rising edge	0	0	do capture; issue NEWVAL_IRQ; CNT++
1	-	0	0	no

28.13.4.2.6 TIM Gated Periodic Sampling Mode (TGPS)

In the TIM Gated Periodic Sampling Mode the number of CMU clock cycles which should elapse before capturing and raising *TIM[i]_NEWVAL[x]_IRQ* is programmable. In this mode it must be specified in the **CNTS** register after how many CMU clock cycles the interrupt has to be raised.

A value of 0 in **TIM[i]_CH[x]_CNTS** means that after one **CLK_SEL** edge a trigger/interrupt is raised, and a value of 1 means that after two edges a trigger/interrupt is raised, and so on.

In the **TIM[i]_CH[x]_CNT** register the elapsed cycles were incremented and compared against **TIM[i]_CH[x]_CNTS**. If **TIM[i]_CH[x]_CNT** is greater or equal to **TIM[i]_CH[x]_CNTS** a trigger will be raised. This allows by writing a value to **TIM[i]_CH[x]_CNTS** that the actual period time can be changed on the fly.

Operation is done depending on CMU clock, **DSL**, **ISL** bit and the input signal value defined in next table:

Table 37 Operation depending on CMU clock, DSL, ISL and the input signal value

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description

-	1	0	1	-	if CNT >= CNTS then do capture
					  issue NEWVAL_IRQ;
					  CNT=0
					else
					  CNT++
					endif

0	0	0	0	1	no

1	1	0	0	1	if CNT >= CNTS then do capture;
					  issue NEWVAL_IRQ;
					  CNT=0
					else
					  CNT++
					endif

0	0	-	0	1	no

0	1	0	0	0	if CNT >= CNTS then do capture;
					  issue NEWVAL_IRQ;
					  CNT=0
					else
					  CNT++
					endif

Table 37 Operation depending on CMU clock, DSL, ISL and the input signal value (cont'd)

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description
1	0	0	0	0	no
-	0	0	-	-	no

In this mode the **TIM[i]_CH[x]_GPR1** operates as a shadow register for **TIM[i]_CH[x]_CNTS**. This would allow that the period for the next sampling period could be specified. The update of **TIM[i]_CH[x]_CNTS** will only take place once on a trigger if the **TIM[i]_CH[x]_GPR1** was written by the CPU. This means that the captured value from the previous trigger can be read by the CPU from **TIM[i]_CH[x]_GPR1** and afterwards the new sampling period for the next sampling period (the one after the actual sampling period) could be written.

With each triggered interrupt, the registers **GPR0** and **GPR1** are updated according to bits **GPR0_SEL**, **GPR1_SEL, EGPR0_SEL** and **EGPR1_SEL**.

When selecting **ECNT** as a source for the capture registers, GPRx will show the edge count and the input signal value at point of capture. Selecting **GPR0_SEL** = '11' and **EGPR0_SEL** = '0' for TIM channel 0 all 8 TIM input signals will be captured to **GPR0[7:0]**.

In the TGPS Mode the bit field **CLK_SEL** of register **TIM[i]_CH[x]_CTRL** will define the selected CMU clock which will be used.

The behavior of the **ECNT** counter is configurable by **ECNT_RESET**. If set to 1 on each interrupt (period expired) the **ECNT** will be reset. Otherwise it operates in wrap around mode.

If the register **ECNT** produces an overflow during the measurement, the bit **ECNTOFL** is set inside the register **TIM[i]_CH[x]_IRQ_NOTIFY** and interrupt *TIM_ECNTOFL[x]_IRQ* is raised depending on corresponding interrupt enable condition.

External capture TIM Gated Periodic Sampling Mode (TGPS)

If external capture is enabled, the external capture events will capture the GPRx, reset the counter **CNT** and issue a *NEWVAL_IRQ*.

Operation is done depending on the CMU clock, external capture signal, **DSL**, **ISL** bit and the input signal value defined in next table:

Table 38 Operation depending on the CMU clock, external capture signal, DSL, ISL and the input signal value

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description

-	1	0	1	-	if CNT >= CNTS then do
					capture;
					issue NEWVAL_IRQ;
					CNT=0
					else
					CNT++
					endif

0	0	0	0	1	no

Table 38 Operation depending on the CMU clock, external capture signal, DSL, ISL and the input signal value (cont'd)

Input signal F_OUTx	selected CMU Clock	External capture	ISL	DSL	Action description

1	1	0	0	1	if CNT >= CNTS then do
					capture;
					issue NEWVAL_IRQ;
					CNT=0
					else
					CNT++
					endif

0	0	-	0	1	no

0	1	0	0	0	if CNT >= CNTS then do
					capture;
					issue NEWVAL_IRQ;
					CNT=0
					else
					CNT++
					endif

1	0	0	0	0	no
-	0	0	-	-	no
-	-	rising edge	-	-	do capture; issue NEWVAL_IRQ; CNT =0

28.13.4.2.7 TIM Serial Shift Mode (TSSM)

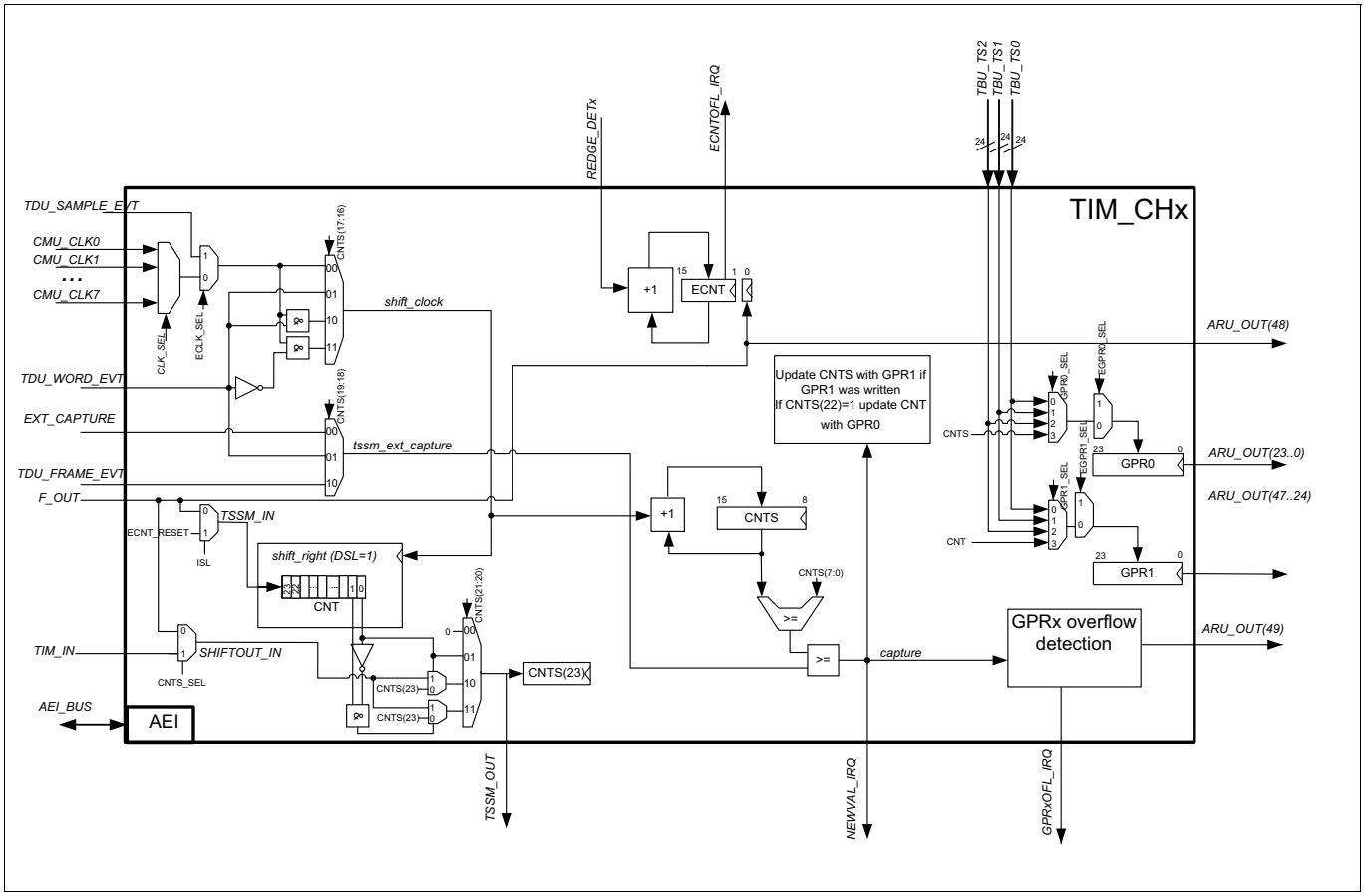


Figure 46 TIM Serial Shift Mode (TSSM)

In the TIM Serial Shift Mode on each shift clock event the actual value of the input signal **TSSM_INx** will be registered in dependence of **DSL** n the register **TIM[i]_CH[x]_CNT**.

If **ISL**=0 is set **FOUTx** will be used as shift in value **TSSM_INx**, with **ISL**=1 the bit field **ECNT_RESET** defines the value for **TSSM_INx**.

With **DSL**=0 **TSSM_INx** will be stored in **TIM[i]_CH[x]_CNT[0]** and **TIM[i]_CH[x]_CNT[22:0]** will be shifted left. With **DSL**</ b>=1 **TSSM_OUTx** will be stored in **TIM[i]_CH[x]_CNT[23]** and **TIM[i]_CH[x]_CNT[23:1]** will be shifted right.

Operation is done depending on the shift clock, external capture signal, **DSL**, **ISL** bit and the input signal value defined in next table:

Table 39 Operation depending on the shift clock, external capture signal, DSL, ISL and the input signal value

Input signal TSSM_INx	shift clock	tssm_ext_capture	ISL	DSL	Action description
-	0	0	-	-	no
-	-	1	-	-	if EXT_CAP_EN=1 then see function table in next chapter else no endif

Table 39 Operation depending on the shift clock, external capture signal, DSL, ISL and the input signal value (cont'd)

Input signal TSSM_INx	shift clock	tssm_ext_capture	ISL	DSL	Action description

value	1	0	0	0	CNT[23:1]= CNT[22:0];
					CNT[0]= value
					if CNTS[15:8] >= CNTS[7:0]
					then
					do capture;
					issue NEWVAL_IRQ;
					CNTS[15:8]=0
					else
					CNTS[15:8]++
					endif

value	1	0	0	1	CNT[22:0]= CNT[23:1];
					CNT[23]= value
					if CNTS[15:8] >= CNTS[7:0]
					then
					do capture;
					issue NEWVAL_IRQ;
					CNTS[15:8]=0
					else
					CNTS[15:8]++
					endif

value	1	0	1	0	CNT[23:1]= CNT[22:0]; CNT[0]= value if CNTS[15:8] >= CNTS[7:0] then do capture; issue NEWVAL_IRQ; CNTS[15:8]=0 CNT[23:0]=ECNT_RESET else CNTS[15:8]++ endif

value	1	0	1	1	CNT[22:0]= CNT[23:1];
					CNT[23]= value
					if CNTS[15:8] >= CNTS[7:0]
					then
					do capture;
					issue NEWVAL_IRQ;
					CNTS[15:8]=0
					CNT[23:0]=ECNT_RESET
					else
					CNTS[15:8]++
					endif

The register **TIM[i]_CH[x]_CNTS[7:0]** define the amount of bits which will be stored inside **TIM[i]_CH[x]_CNT.**

Each shift clock will increment the register **TIM[i]_CH[x]_CNTS[15:8].** If the condition **TIM[i]_CH[x]_CNTS[15:8] >= TIM[i]_CH[x]_CNTS[7:0]** is met a capture event is raised and *TIM[i]_NEWVAL[x]_IRQ* is asserted.

With each capture event the registers **GPR0** and **GPR1** are updated according to bits **GPR0_SEL**, **GPR1_SEL, EGPR0_SEL** and **EGPR1_SEL**.

If the bit field **ISL** is set to 1 the register bits **TIM[i]_CH[x]_CNT** are set to the value defined by **ECNT_RESET** in case of a capture event.

In this mode the **TIM[i]_CH[x]_GPR1** operates as a shadow register for **TIM[i]_CH[x]_CNTS**. This allows that the amount of bits to sample can be specified. The update of **TIM[i]_CH[x]_CNTS** will only take place once on a trigger if the **TIM[i]_CH[x]_GPR1** was written by the CPU. This means that the captured value from the previous trigger can be read by the CPU from **TIM[i]_CH[x]_GPR1** and afterwards the new amount of bits to sample for the next sampling period (the one after the actual sampling period) could be written.

The shift clock which will be in use is selectable by **TIM[i]_CH[x]_CNTS[17:16]**:

$\mathbf{00_B}$: source selection by **USE_TDU_CLK_SRC** is in use. It can be set to any CMU_CLK source or to the local TDU sample clock tdu_sample_evt.

$\mathbf{01_B}$: the tdu_word_evt signal will be used as shift clock source.

$\mathbf{10_B}$: the clk source selected by **USE_TDU_CLK_SRC** is used and gated with tdu_word_evt. If tdu_word_evt=0 then shift clock will be 0.

$\mathbf{11_B}$: the clk source selected by **USE_TDU_CLK_SRC** is used and gated with tdu_word_evt. If tdu_word_evt=1 then shift clock will be 0.

Signal Generation with TIM Serial Shift Mode

If **TIM[i]_CH[x]_CNTS[22]** is 1 the **TIM[i]_CH[x]_GPR0** operates as a shadow register for **TIM[i]_CH[x]_CNT**. This allows that the bits for shifting out can be specified. The update of **TIM[i]_CH[x]_CNT** will only take place once on a trigger if the **TIM[i]_CH[x]_GPR0** was written by the CPU. This means that the captured value from the previous trigger can be read by the CPU from **TIM[i]_CH[x]_GPR0** and afterwards the new bits to shift out could be written.

In addition the TIM Serial Shift Mode is able to generate a signal **TSSM_OUT** which can be used internally to the TIM channel.

On each system clock the value for **TSSM_OUT** is generated as defined next. The actual value can be read by the register bit **TIM[i]_CH[x]_CNTS[23]**.

Following functionality for **TSSM_OUTx** is selectable by **TIM[i]_CH[x]_CNTS[21:20]:**

$\mathbf{00_B}$: Constant output; **TSSM_OUTx** = 0.

$\mathbf{10_B}$:Shift output; If **DSL**=0 (shift left) then **TSSM_OUTx** = **TIM[i]_CH[x]_CNT[23]** else (shift right) **TSSM_OUTx** = **TIM[i]_CH[x]_CNT[0]**.

$\mathbf{10_B}$: Latched output; If **DSL**=0 and **TIM[i]_CH[x]_CNT[23]**=1 then **TSSM_OUTx** = **SHIFTOUT_INx** elsif **DSL**=1 and **TIM[i]_CH[x]_CNT[0]**=1 then **TSSM_OUTx** = **SHIFTOUT_INx**.

$\mathbf{11_B}$: Registered output; If **DSL**=0 and **TIM[i]_CH[x]_CNT[23:22]**=b01 then **TSSM_OUTx** = **SHIFTOUT_INx** elsif **DSL**=1 and **TIM[i]_CH[x]_CNT[1:0]**=b10 then **TSSM_OUTx** = **SHIFTOUT_INx**.

In case of registered or latched output mode the signal **SHIFTOUT_INx** is selectable by **CNTS_SEL**.

If **CNTS_SEL**=0 is set **FOUTx** will be used for **SHIFTOUT_INx**, with **CNTS_SEL**=1 the signal **TIM_INx** is in us for **SHIFTOUT_INx**.

External capture TIM Serial Shift Mode (TSSM)

If external capture is enabled (**EXT_CAP_EN**=1), the external capture events will capture the GPRx, reset the counter **CNT** depending on ISL and issue a *NEWVAL_IRQ*. Functionality from previous table will be applied.

The source which will be used as external capture event for TSSM mode is selectable by **TIM[i]_CH[x]_CNTS[19:18]**:

00_B: source selection by **EXT_CAP_SRC** is in use.

01_B: tdu_word_evt signal will be used as source.

10_B: tdu_frame_evt signal will be used as source.

11_B: reserved

Operation is done depending on the shift clock, external capture signal, **DSL**, **ISL** bit and the input signal value defined in next table:

Table 40 Operation depending on the shift clock, external capture signal, DSL, ISL and the input signal value

Input signal F_OUTx	shift clock	tssm_ext_capture	ISL	DSL	Action description
-	0	1	1	-	do capture; issue NEWVAL_IRQ; CNTS[15:8]=0 CNT[23:0]=ECNT_RESET
-	0	1	0	-	do capture; issue NEWVAL_IRQ; CNTS[15:8]=0
value	1	1	1	0	CNT[23:1]= CNT[22:0]; CNT[0]= value do capture; issue NEWVAL_IRQ; CNTS[15:8]=0 CNT[23:0]=ECNT_RESET
value	1	1	1	1	CNT[22:0]= CNT[23:1]; CNT[23]= value do capture; issue NEWVAL_IRQ; CNTS[15:8]=0 CNT[23:0]=ECNT_RESET
value	1	1	0	0	CNT[23:1]= CNT[22:0]; CNT[0]= value do capture; issue NEWVAL_IRQ; CNTS[15:8]=0
value	1	1	0	1	CNT[22:0]= CNT[23:1]; CNT[23]= value do capture; issue NEWVAL_IRQ; CNTS[15:8]=0

28.13.5 MAP Submodule Interface

The GTM provides one dedicated TIM sub-module TIM0 where channels zero (0) to five (5) are connected to the MAP sub-module described in chapter “TIM0 Input Mapping Module ”. There, the TIM0 sub-module channels provide the input signal level together with the actual filter value and the annotated time stamp for the edge

together in a 49 bit wide signal to the MAP sub-module. This 49 bit wide data signal is marked as valid with a separate valid signal tim0_map_dval[x] (x: 0…5).

28.13.5.1 Structure of map data

Table 41 MAP Submodule Interface

tim0_map_data[x](48)	Signal level bit from tim0_ch[x]
tim0_map_data[x](47:24)	actual filter value TIM0_CH[x]_FLT_RE/ TIM0_CH[x]_FLT_FE if corresponding channel x bit field FLT_MODE_RE/ FLT_MODE_FE is 1 else 0 is assigned.
tim0_map_data[x](23:0)	time stamp value selected by TBU0_SEL, GRP0_SEL, EGPR0_SEL, CNTS_SEL of channel x if bit field TIM_EN= 1
tim0_map_dval[x]	mark tim0_map_data[x] valid for one clock cycle

*Note: With **TIM_EN**=1 the MAP interface starts operation, it is not dependent on the setting of the bit fields **TIM_MODE, ISL, DSL**.*

Note: While the MAP interface is in use the following guidelines have to be fulfilled, otherwise inconsistent filter values can be transferred.

Change **TIM0_CH[x]_FLT_RE** only between occurrence of rising and falling edge.

Change **TIM0_CH[x]_FLT_FE** only between occurrence of falling and rising edge.

28.13.6 TIM Interrupt Signals

Table 42 TIM Interrupt Signals

Signal	Description
TIM[i]_NEWVAL[x]_IRQ	New measurement value detected by SMU of channel x (x: 0…m-1)
TIM[i]_ECNTOFL[x]_IRQ	ECNT counter overflow of channel x (x: 0…m-1)
TIM[i]_CNTOFL[x]_IRQ	SMU CNT counter overflow of channel x (x: 0…m-1)
TIM[i]_GPROFL[x]_IRQ	GPR0 and GPR1 data overflow, old data was not read out before new data has arrived at input pin of channel x (x: 0…m-1)
TIM[i]_TODET[x]_IRQ	Time out reached for input signal of channel x (x: 0…m-1)
TIM[i]_GLITCHDET[x]_IRQ	A glitch was detected by the TIM filter of channel x (x: 0…m-1)

28.13.7 TIM Configuration Register Overview

Table 43 TIM Configuration Register Overview

Register Name	Description	see Page
TIM[i]_CH[x]_CTRL	TIMi channel x control register	178
TIM[i]_CH[x]_ECTRL	TIMi channel x extended control register	194
TIM[i]_CH[x]_FLT_RE	TIMi channel x filter parameter 0 register	182

Table 43 TIM Configuration Register Overview (cont'd)

Register Name	Description	see Page
TIM[i]_CH[x]_FLT_FE	TIMi channel x filter parameter 1 register	183
TIM[i]_CH[x]_TDUV	TIMi channel x TDU control register	192
TIM[i]_CH[x]_TDUC	TIMi channel x TDU counter register	193
TIM[i]_CH[x]_GPR0	TIMi channel x general purpose 0 register	183
TIM[i]_CH[x]_GPR1	TIMi channel x general purpose 1 register	184
TIM[i]_CH[x]_CNT	TIMi channel x SMU counter register	185
TIM[i]_CH[x]_ECNT	TIMi channel x SMU edge counter register	194
TIM[i]_CH[x]_CNTS	TIMi channel x SMU shadow counter register	185
TIM[i]_CH[x]_IRQ_NOTIFY	TIMi channel x interrupt notification register	186
TIM[i]_CH[x]_IRQ_EN	TIMi channel x interrupt enable register	187
TIM[i]_CH[x]_EIRQ_EN	TIMi channel x error interrupt enable register	191
TIM[i]_CH[x]_IRQ_FORCINT	TIMi channel x force interrupt register	188
TIM[i]_CH[x]_IRQ_MODE	TIMi interrupt mode configuration register	189
TIM[i]_RST	TIMi global software reset register	188
TIM[i]_IN_SRC	TIMi AUX IN source selection register	190
TIM[i]_INP_VAL	TIMi input value observation register	198

28.13.8 TIM Configuration Registers Description

28.13.8.1 Register TIM[i]_CH[x]_CTRL

TIMi Channel x Control Register

Table 44 Filter Modes for Rising Edge

FLT_MODE_RE		EFLT_CTR_RE[1)]	FLT_CTR_RE	Coding
0	Immediate edge propagation mode for rising edge	0	0	Immediate edge propagation mode
		0	1	Immediate edge propagation mode
		1	0	Reserved
		1	1	Reserved
1	Individual de-glitch mode for rising edge	0	0	Up-Down Counter individual de-glitch mode
		0	1	Hold Counter individual de-glitch mode
		1	0	Reset Counter individual de-glitch mode
		1	1	Reserved

1) Bit is located in register TIM[i]_CH[x]_ECTRL.

Table 45 Filter Modes for Falling Edge

FLT_MODE_FE		EFLT_CTR_FE[1)]	FLT_CTR_FE	Coding
0	Immediate edge propagation mode for rising edge	0	0	Immediate edge propagation mode
		0	1	Immediate edge propagation mode
		1	0	Reserved
		1	1	Reserved
1	Individual de-glitch mode for rising edge	0	0	Up-Down Counter individual de-glitch mode
		0	1	Hold Counter individual de-glitch mode
		1	0	Reset Counter individual de-glitch mode
		1	1	Reserved

1) Bit is located in register TIM[i]_CH[x]_ECTRL.

TIMi_CHx_CTRL (i=0-7;x=0-7)

TIMi Channel x Control Register **($001024_H+i*800_H+x*80_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOCTRL		EGPR1_SEL	EGPR0_SEL	FR_ECNT_OFL		CLK_SEL		FLT_CTR_FE	FLT_MODE_FE	FLT_CTR_RE	FLT_MODE_RE	EXT_CAP_EN	FLT_CNT_FRQ		FLT_EN
rw		rw	rw	rw		rw		rw	rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECNT_RESET	ISL	DSL	CNTS_SEL	GPR1_SEL		GPR0_SEL		TBU0_SEL	CICTRL	ARU_EN	OSM	TIM_MODE			TIM_EN
rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw			rw

Field	Bits	Type	Description
TIM_EN	0	rw	**TIM channel x enable** Enabling of the channel resets the registers **ECNT, TIM[i]_CH[x]_CNT, TIM[i]_CH[x]_GPR0,** and **TIM[i]_CH[x]_GPR1** to their reset values. After finishing the action in one-shot mode the **TIM_EN** bit is cleared automatically. Otherwise, the bit must be cleared manually. 0_B Channel disabled 1_B Channel enabled
TIM_MODE	3:1	rw	**TIM channel x mode** If an undefined value is written to the TIM_MODE register, the hardware switches automatically to TIM_MODE = 0b000 (TPWM mode). The TIM_MODE register should not be changed while the TIM channel is enabled. If the TIM channel is enabled and operating in TPWM or TPIM mode after the first valid edge defined by DSL has occurred, a reconfiguration of DSL,ISL,TIM_MODE will not change the channel behavior. Reading these bit fields after reconfiguration will show the newly configured settings but the initial channel behavior will not change. Only a disabling of the TIM channel by setting TIM_EN= 0 and reenabling with TIM_EN= 1 will change the channel operation mode. 000_B PWM Measurement Mode (TPWM) 001_B Pulse Integration Mode (TPIM) 010_B Input Event Mode (TIEM) 011_B Input Prescaler Mode (TIPM) 100_B Bit Compression Mode (TBCM) 101_B Gated Periodic Sampling Mode (TGPS) 110_B Serial Shift Mode (TSSM)
OSM	4	rw	**One-shot mode** After finishing the action in one-shot mode the **TIM_EN** bit is cleared automatically. 0_B Continuous operation mode 1_B One-shot mode
ARU_EN	5	rw	**GPR0 and GPR1 register values routed to ARU** 0_B Registers content not routed 1_B Registers content routed
CICTRL	6	rw	**Channel Input Control** 0_B Use signal TIM_IN(x) as input for channel x 1_B Use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0)
TBU0_SEL	7	rw	**TBU_TS0 bits input select for TIM0_CH[x]_GPRz (z: 0, 1)** This bit is only applicable for TIM0. 0_B Use TBU_TS0(23..0) to store in TIM0_CH[x]_GPR0/TIM0_CH[x]_GPR1 1_B Use TBU_TS0(26..3) to store in TIM0_CH[x]_GPR0/TIM0_CH[x]_GPR1

Field	Bits	Type	Description
GPR0_SEL	9:8	rw	**Selection for GPR0 register** If EGPR0_SEL =0 / EGPR0_SEL =1 : If a reserved value is written to the EGPR0_SEL, GPR0_SEL bit fields, the hardware will use TBU_TS0 input. 00_B Use TBU_TS0 as input / use ECNT as input 01_B Use TBU_TS1 as input / use TIM_INP_VAL as input 10_B Use TBU_TS2 as input / reserved 11_B Use CNTS as input; if TGPS mode in channel = 0 is selected, use TIM Filter F_OUT as input / reserved
GPR1_SEL	11:10	rw	**Selection for GPR1 register** If EGPR1_SEL =0 / EGPR1_SEL =1: If a reserved value is written to the EGPR1_SEL, GPR1_SEL bit fields, the hardware will use TBU_TS0 input. Note: In TBCM mode: EGPR1_SEL=1, GPR1_SEL=01 selects TIM_INP_VAL as input; in all other cases, TIM Filter F_OUT is used. 00_B Use TBU_TS0 as input / use ECNT as input 01_B Use TBU_TS1 as input / use TIM_INP_VAL as input 10_B Use TBU_TS2 as input / reserved 11_B Use CNT as input / reserved
CNTS_SEL	12	rw	**Selection for CNTS register** The functionality of the **CNTS_SEL** is disabled in the modes TIPM,TGPS and TBCM. **CNTS_SEL** in TSSM mode selects the source signal for registered or latched shift out operation. 0_B use F_OUTx 1_B use TIM_INx 0_B Use CNT register as input 1_B Use TBU_TS0 as input
DSL	13	rw	**Signal level control** In TIM_MODE=0b110 (TSSM), the bit field DSL defines the shift direction. 0_B Shift left 1_B Shift right 0_B Measurement starts with falling edge (low level measurement) 1_B Measurement starts with rising edge (high level measurement)
ISL	14	rw	**Ignore signal level** This bit is mode dependent and will have different meanings (see details in the TIM Channel mode description). 0_B Use DSL bit for selecting active signal level (TIEM) 1_B Ignore DSL and treat both edges as active edge (TIEM)

Field	Bits	Type	Description
ECNT_RESET	15	rw	**Enables resetting of counter in certain modes** If TIM_MODE=0b101 (TGPS) / TIM_MODE=0b000 (TPWM) else ECNT counter operating in wrap around mode; In TIM_MODE=0b110 (TSSM), the bit field ECNT_RESET defines the initial polarity for the shift register. 0_B ECNT counter operating in wrap around mode / ECNT counter operating in wrap around mode, CNT is reset on active input edge defined by DSL 1_B ECNT counter is reset with periodic sampling / ECNT counter operating in wrap around mode, CNT is reset on active and inactive input edge
FLT_EN	16	rw	**Filter enable for channel x** If the filter is disabled, all filter related units (including CSU) are bypassed, which means that the signal *F_IN* is directly routed to signal *F_OUT*. 0_B Filter disabled and internal states are reset 1_B Filter enabled
FLT_CNT_FRQ	18:17	rw	**Filter counter frequency select** 00_B FLT_CNT counts with CMU_CLK0 01_B FLT_CNT counts with CMU_CLK1 10_B FLT_CNT counts with CMU_CLK6 11_B FLT_CNT counts with CMU_CLK7
EXT_CAP_EN	19	rw	**Enables external capture mode** The selected TIM mode is only sensitive to external capture pulses the input event changes are ignored. 0_B External capture disabled 1_B External capture enabled
FLT_MODE_RE	20	rw	**Filter mode for rising edge** Coding see **Table 44**.
FLT_CTR_RE	21	rw	**Filter counter mode for rising edge** Coding see **Table 44**.
FLT_MODE_FE	22	rw	**Filter mode for falling edge** Coding see **Table 45**.
FLT_CTR_FE	23	rw	**Filter counter mode for falling edge** Coding see **Table 45**.
CLK_SEL	26:24	rw	**CMU clock source select for channel** If ECLK_SEL =0 / ECLK_SEL =1: 000_B CMU_CLK0 selected / tdu_sample_evt of TDU selected 001_B CMU_CLK1 selected / reserved 010_B CMU_CLK2 selected / reserved 011_B CMU_CLK3 selected / reserved 100_B CMU_CLK4 selected / reserved 101_B CMU_CLK5 selected / reserved 110_B CMU_CLK6 selected / reserved 111_B CMU_CLK7 selected / reserved

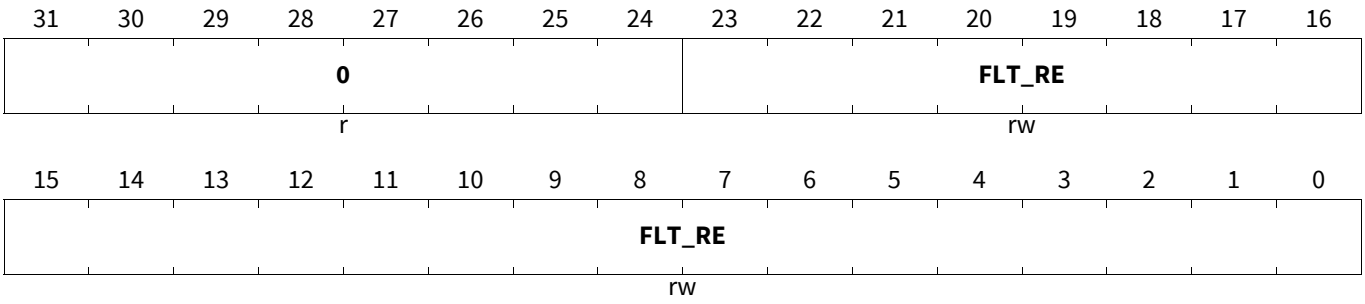
Field	Bits	Type	Description
FR_ECNT_OFL	27	rw	**Extended Edge counter overflow behavior** 0_B Overflow will be signaled on ECNT bit width = 8 1_B Overflow will be signaled on EECNT bit width (full range)
EGPR0_SEL	28	rw	**Extension of GPR0_SEL bit field** Details described in GPR0_SEL bit field.
EGPR1_SEL	29	rw	**Extension of GPR1_SEL bit field** Details described in GPR1_SEL bit field.
TOCTRL	31:30	rw	**Timeout control** It has to be mentioned that writing of TOCTRL= 0 will every time stop the TDU, independent of the previous state of TOCTRL. 00_B Timeout feature disabled 01_B Timeout feature enabled for rising edge only 10_B Timeout feature enabled for falling edge only 11_B Timeout feature enabled for both edges

28.13.8.2 Register TIM[i]_CH[x]_FLT_RE

TIMi Channel x Filter Parameter 0 Register

TIMi_CHx_FLT_RE (i=0-7;x=0-7)

TIMi Channel x Filter Parameter 0 Register($00101C_H$+i*800_H+x*80_H) Application Reset Value: 0000 0000_H



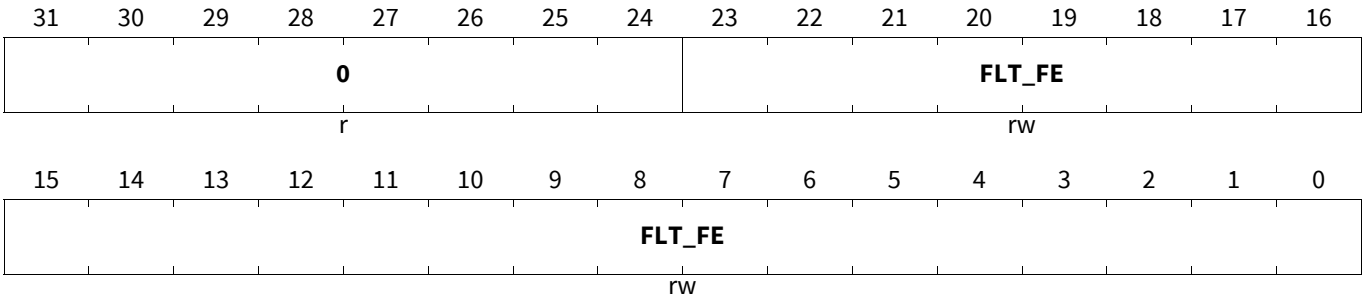
Field	Bits	Type	Description
FLT_RE	23:0	rw	**Filter parameter for rising edge** FLT_RE has different meanings in the various filter modes.Immediate edge propagation mode = acceptance time for rising edgeIndividual deglitch time mode = deglitch time for rising edge.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.13.8.3 Register TIM[i]_CH[x]_FLT_FE

TIMi Channel x Filter Parameter 1 Register

TIMi_CHx_FLT_FE (i=0-7;x=0-7)

TIMi Channel x Filter Parameter 1 Register(001020_H+i*800_H+x*80_H) Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
FLT_FE	23:0	rw	**Filter parameter for falling edge** FLT_FE has different meanings in the various filter modes.Immediate edge propagation mode = acceptance time for falling edge Individual deglitch time mode = deglitch time for falling edge.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

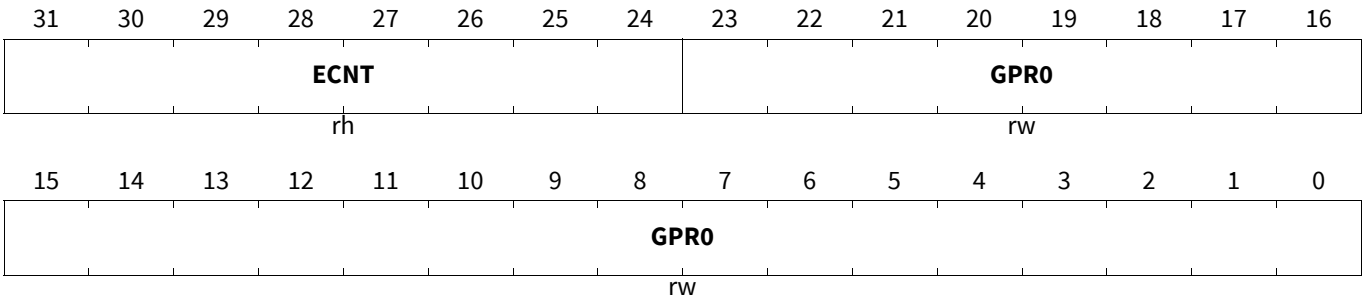
28.13.8.4 Register TIM[i]_CH[x]_GPR0

TIMi Channel x General Purpose 0 Register

Note: The ECNT register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.

TIMi_CHx_GPR0 (i=0-7;x=0-7)

TIMi Channel x General Purpose 0 Register(001000_H+i*800_H+x*80_H) Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
GPR0	23:0	rw	**Input signal characteristic parameter 0** The content of this register has different meaning for the TIM channels modes. The content directly depends on the bit fields **EGPR0_SEL**, **GPR0_SEL** of register **TIM[i]_CH[x]_CTRL**. **Note:** The content of this register can only be written in TIM channel mode TSSM.
ECNT	31:24	rh	**Edge counter** The **ECNT** counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of **ECNT**.

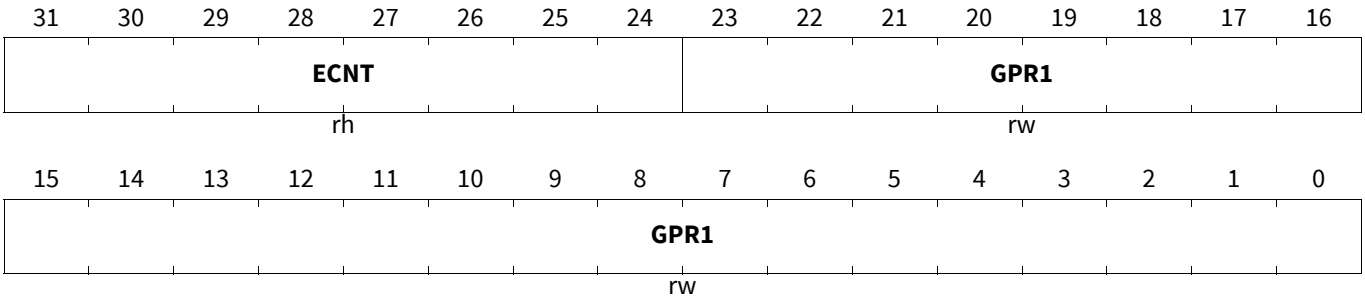
28.13.8.5 Register TIM[i]_CH[x]_GPR1

TIMi Channel x General Purpose 1 Register

Note: The ECNT register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.

TIMi_CHx_GPR1 (i=0-7;x=0-7)

TIMi Channel x General Purpose 1 Register(001004_H+i*800_H+x*80_H) Application Reset Value: 0000 0000_H

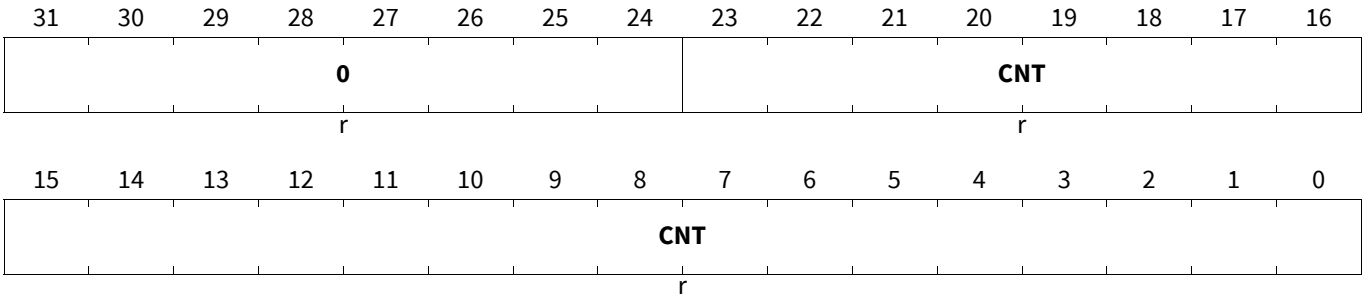


Field	Bits	Type	Description
GPR1	23:0	rw	**Input signal characteristic parameter 1** The content of this register has different meaning for the TIM channels modes. The content directly depends on the bit fields **EGPR1_SEL**, **GPR1_SEL** of register **TIM[i]_CH[x]_CTRL**. In TBCM mode if EGPR1_SEL=1, GPR1_SEL=01 then TIM_INP_VAL is used as input in all other cases TIM Filter F_OUT is used as input and Bits GPR1(23:8) = 0 The content of this register can only be written in TIM channel mode TGPS and TSSM.
ECNT	31:24	rh	**Edge counter** The **ECNT** counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of **ECNT**.

28.13.8.6 Register TIM[i]_CH[x]_CNT

TIMi Channel x SMU Counter Register

TIMi_CHx_CNT (i=0-7;x=0-7)
TIMi Channel x SMU Counter Register (001008_H+i*800_H+x*80_H) Application Reset Value: 0000 0000_H



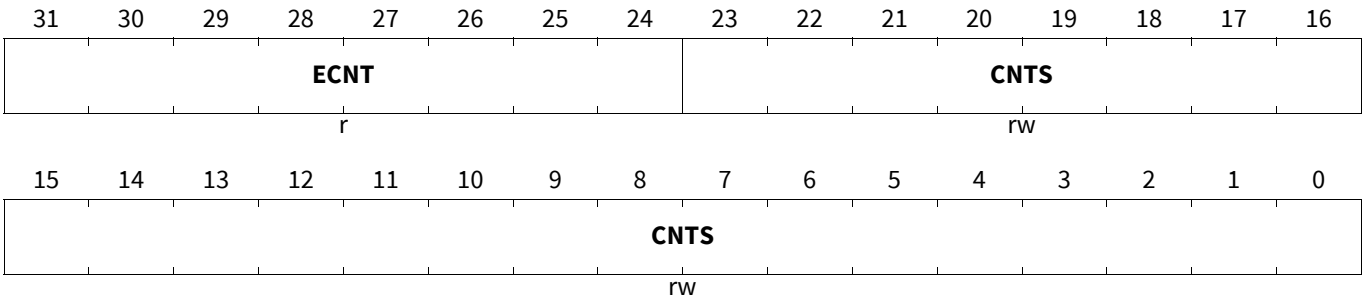
Field	Bits	Type	Description
CNT	23:0	r	**Actual SMU counter value** The meaning of this value depends on the configured mode: TPWM = actual duration of PWM signal.TPIM = actual duration of all pulses (sum of pulses).TIEM = actual number of received edges.TIPM = actual number of received edges.TGPS = elapsed time for periodic sampling.TSSM = shift data.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.13.8.7 Register TIM[i]_CH[x]_CNTS

TIMi Channel x SMU Shadow Counter Register

Note: The ECNT register is reset to its initial value when the channel is enabled. Please note, that bit 0 depends on the input level coming from the filter unit and defines the reset value immediately.

TIMi_CHx_CNTS (i=0-7;x=0-7)
TIMi Channel x SMU Shadow Counter Register(001010_H+i*800_H+x*80_H)Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
CNTS	23:0	rw	**Counter shadow register** The content of this register has different meaning for the TIM channels modes. The content depends directly on the bit field **CNTS_SEL** of register **TIM[i]_CH[x]_CTRL**. The register **TIM[i]_CH[x]_CNTS** is only writable in TIPM,TBCM ,TGPS and TSSM mode.
ECNT	31:24	r	**Edge counter** The **ECNT** counts every incoming filtered edge (rising and falling). The counter value is uneven in case of detected rising, and even in case of detected falling edge. Thus, the input signal level is part of the counter and can be obtained by bit 0 of **ECNT**.

28.13.8.8 Register TIM[i]_CH[x]_IRQ_NOTIFY

TIMi Channel x Interrupt Notification Register

TIMi_CHx_IRQ_NOTIFY (i=0-7;x=0-7)
TIMi Channel x Interrupt Notification Register($00102C_H$+i*800_H+x*80_H)Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										GLITCHDET	TODET	GPROFL	CNTOFL	ECNTOFL	NEWVAL
r										rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NEWVAL	0	rw	**New measurement value detected by in channel x** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No event has occurred 1_B NEWVAL has occurred on the TIM channel
ECNTOFL	1	rw	**counter overflow of channel x** See bit 0.
CNTOFL	2	rw	**SMU CNT counter overflow of channel x** See bit 0.
GPROFL	3	rw	**GPR0 and GPR1 data overflow** Old data not read out before new data has arrived at input pin. See bit 0.
TODET	4	rw	**Timeout reached for input signal of channel x** See bit 0.

Field	Bits	Type	Description
GLITCHDET	5	rw	**Glitch detected on channel x, (x:0...m-1)** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No glitch detected for last edge 1_B Glitch detected for last edge
0	31:6	r	**Reserved** Read as zero, shall be written as zero.

28.13.8.9 Register TIM[i]_CH[x]_IRQ_EN

TIMi Channel x Interrupt Enable Register

TIMi_CHx_IRQ_EN (i=0-7;x=0-7)

TIMi Channel x Interrupt Enable Register(001030_H+i*800_H+x*80_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										GLITCHDET_IRQ_EN	TODET_IRQ_EN	GPROFL_IRQ_EN	CNTOFL_IRQ_EN	ECNTOFL_IRQ_EN	NEWVAL_IRQ_EN
r										rw	rw	rw	rw	rw	rw

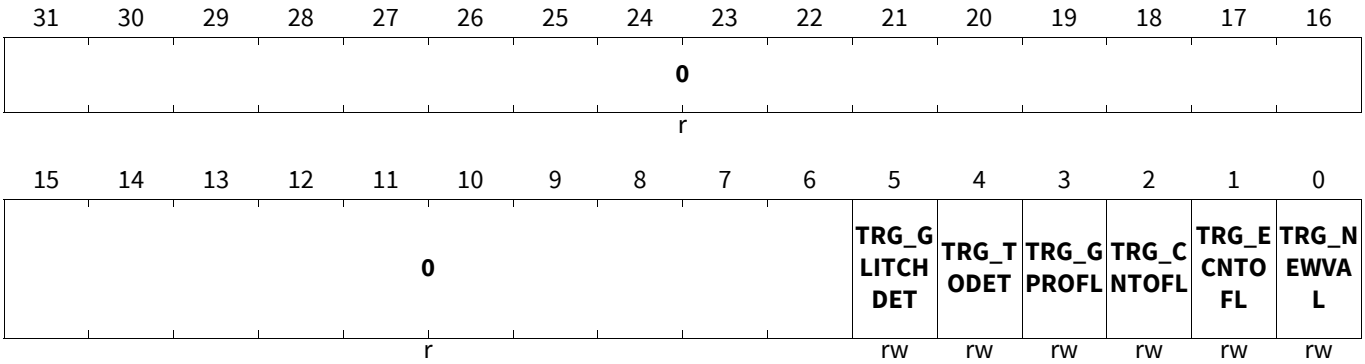
Field	Bits	Type	Description
NEWVAL_IRQ_EN	0	rw	**TIM_NEWVALx_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
ECNTOFL_IRQ_EN	1	rw	**TIM_ECNTOFLx_IRQ interrupt enable** Coding see bit 0.
CNTOFL_IRQ_EN	2	rw	**TIM_CNTOFLx_IRQ interrupt enable** Coding see bit 0.
GPROFL_IRQ_EN	3	rw	**TIM_GPROFL_IRQ interrupt enable** Coding see bit 0.
TODET_IRQ_EN	4	rw	**TIM_TODETx_IRQ interrupt enable** Coding see bit 0.
GLITCHDET_IRQ_EN	5	rw	**TIM_GLITCHDETx_IRQ interrupt enable** Coding see bit 0.
0	31:6	r	**Reserved** Read as zero, shall be written as zero.

28.13.8.10Register TIM[i]_CH[x]_IRQ_FORCINT

TIMi Channel x Force Interrupt Register

TIMi_CHx_IRQ_FORCINT (i=0-7;x=0-7)

TIMi Channel x Force Interrupt Register(001034_H+i*800_H+x*80_H) **Application Reset Value: 0000 0000_H**

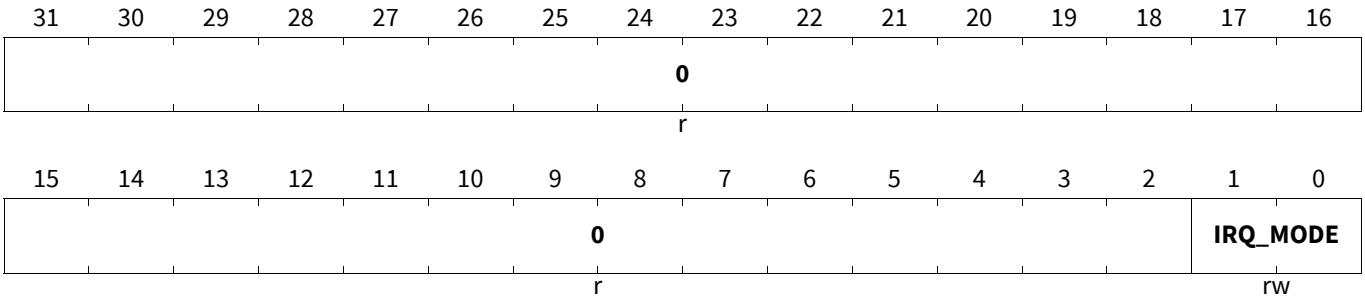


Field	Bits	Type	Description
TRG_NEWVAL	0	rw	**Trigger NEWVAL bit in TIM_CHx_IRQ_NOTIFY register by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B No interrupt triggering 1_B Assert corresponding field in TIM[i]_CH[x]_IRQ_NOTIFY register
TRG_ECNTOFL	1	rw	**Trigger ECNTOFL bit in TIM_CHx_IRQ_NOTIFY register by software** Coding see bit 0.
TRG_CNTOFL	2	rw	**Trigger CNTOFL bit in TIM_CHx_IRQ_NOTIFY register by software** Coding see bit 0.
TRG_GPROFL	3	rw	**Trigger GPROFL bit in TIM_CHx_IRQ_NOTIFY register by software** Coding see bit 0.
TRG_TODET	4	rw	**Trigger TODET bit in TIM_CHx_IRQ_NOTIFY register by software** Coding see bit 0.
TRG_GLITCHDET	5	rw	**Trigger GLITCHDET bit in TIM_CHx_IRQ_NOTIFY register by software** Coding see bit 0.
0	31:6	r	**Reserved** Read as zero, shall be written as zero.

28.13.8.11Register TIM[i]_CH[x]_IRQ_MODE

TIMi Channel x Interrupt Mode Configuration Register

TIMi_CHx_IRQ_MODE (i=0-7;x=0-7)
TIMi Channel x Interrupt Mode Configuration Register(001038_H+i*800_H+x*80_H) Application Reset Value: $0000\ 0000_H$

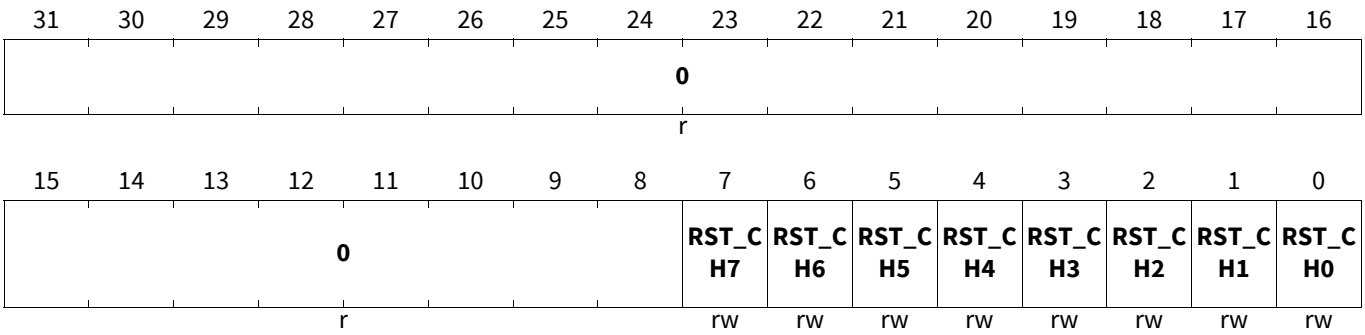


Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** The interrupt modes are described in **Section 28.4.5**. 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero

28.13.8.12Register TIM[i]_RST

TIMi Global Software Reset Register

TIMi_RST (i=0-7)
TIMi Global Software Reset Register ($00107C_H$+i*800_H) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RST_CHx (x=0-7)	x	rw	**Software reset of channel x** This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately. Please note, that the RST field width of this register depends on the number of implemented channels m within this sub-module. This register description represents a register layout for m = 8. 0_B No action 1_B Reset channel x
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.13.8.13Register TIM[i]_IN_SRC

TIMi AUX IN Source Selection Register

TIMi_IN_SRC (i=0-7)
TIMi AUX IN Source Selection Register **(001078_H+i*800_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE_7		**VAL_7**		**MODE_6**		**VAL_6**		**MODE_5**		**VAL_5**		**MODE_4**		**VAL_4**	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE_3		**VAL_3**		**MODE_2**		**VAL_2**		**MODE_1**		**VAL_1**		**MODE_0**		**VAL_0**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
VAL_x (x=0-7)	4*x+1:4*x	rw	**Value to be fed to Channel x** Multicore encoding in use (**VAL_x(1)** defines the state of the signal). Function depends on the combination of VAL_x(1) and MODE_x(1) see MODE_0 description. Any read access to a **VAL_x** bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored. 00_B State is 0 (ignore write access) 01_B Change state to 0 10_B Change state to 1 11_B State is 1 (ignore write access)

Field	Bits	Type	Description
MODE_x (x=0-7)	4*x+3:4*x+2	rw	**Input source to Channel x** Multicore encoding in use (**MODE_x(1)** defines the state of the signal). Function table: **MODE_x(1)=0** , **VAL_x(1)**=0: The input signal defined by bit field **CICTRL** of the TIM channel is used as input source. **MODE_x(1)=0** , **VAL_x(1)**=1: The signal TIM_AUX_IN of the TIM channel is used as input source. **MODE_x(1)=1** : The state **VAL_x(1)** defines the input level for the TIM channel. Any read access to a **MODE_x** bit field will always result in a value 00 or 11 indicating current state. A modification of the state is only performed with the values 01 and 10. Writing the values 00 and 11 is always ignored. 00_B State is 0 (ignore write access) 01_B Change state to 0 10_B Change state to 1 11_B State is 1 (ignore write access)

28.13.8.14Register TIM[i]_CH[x]_EIRQ_EN

TIMi Channel x Error Interrupt Enable Register

TIMi_CHx_EIRQ_EN (i=0-7;x=0-7)

TIMi Channel x Error Interrupt Enable Register($00103C_H$+i*800_H+x*80_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										GLITCHDET_EIRQ_EN	TODET_EIRQ_EN	GPROFL_EIRQ_EN	CNTOFL_EIRQ_EN	ECNTOFL_EIRQ_EN	NEWVAL_EIRQ_EN
r										rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NEWVAL_EIRQ_EN	0	rw	**TIM_NEWVALx_EIRQ error interrupt enable** 0_B Disable error interrupt, error interrupt is not visible outside GTM 1_B Enable error interrupt, error interrupt is visible outside GTM
ECNTOFL_EIRQ_EN	1	rw	**TIM_ECNTOFLx_IRQ interrupt enable** Coding see bit 0.
CNTOFL_EIRQ_EN	2	rw	**TIM_CNTOFLx_IRQ interrupt enable** Coding see bit 0.
GPROFL_EIRQ_EN	3	rw	**TIM_GPROFL_IRQ interrupt enable** Coding see bit 0.

Field	Bits	Type	Description
TODET_EIRQ_EN	4	rw	**TIM_TODETx_IRQ interrupt enable** Coding see bit 0.
GLITCHDET_EIRQ_EN	5	rw	**TIM_GLITCHDETx_IRQ interrupt enable** Coding see bit 0.
0	31:6	r	**Reserved** Read as zero, shall be written as zero.

28.13.8.15Register TIM[i]_CH[x]_TDUV

TIMi Channel x TDU Control Register

TIMi_CHx_TDUV (i=0-7;x=0-7)
TIMi Channel x TDU Control Register (001018$_H$+i*800$_H$+x*80$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		TCS		TDU_SAME_CNT_CLK	TCS_USE_SAMPLE_EVT	SLICING		TOV2							
r		rw		rw	rw	rw		rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOV1								TOV							
rw								rw							

Field	Bits	Type	Description
TOV	7:0	rw	**Time out compare value slice0 for channel x** Compare value for TO_CNT.
TOV1	15:8	rw	**Time out compare value slice1 for channel x** Compare value for TO_CNT1.
TOV2	23:16	rw	**Time out compare value slice2 for channel x** SLICING!=0b11: Compare value for TO_CNT2. SLICING= 0b11: TOV2 operate as a shadow register for TO_CNT.
SLICING	25:24	rw	**Cascading of counter slices** If USE_LUT=0b00 / USE_LUT !=0b00 00_B Combine slice2, slice1, slice0 to 1x24-bit counter / reserved 01_B Combine slice1, slice0 to 1x16-bit counter, use slice2 as 1x8-bit counter / combine slice1, slice0 to 1x16-bit; slice2 not usable 10_B Use slice2, slice1, slice0 as 3x8-bit counter / use slice1, slice0 as 2x8-bit counter; slice2 not usable 11_B Use slice1, slice0 as 2x8-bit counter / use slice1, slice0 as 2x8-bit counter
TCS_USE_SAMPLE_EVT	26	rw	**Use tdu_sample_evt as Timeout Clock** 0_B CMU_CLK selected by TCS is in use by TO_CNT, TO_CNT2 1_B CMU_CLK selected by TCS is in use by TO_CNT2; tdu_sample_evt is in use by TO_CNT

Field	Bits	Type	Description
TDU_SAME_CNT_CLK	27	rw	**Define clocking of TO_CNT, TO_CNT1** 0_B TO_CNT clock selected by (TCS, TCS_USE_SAMPLE_EVT); TO_CNT1 clocked on tdu_word_event 1_B TO_CNT1 uses same clock as TO_CNT
TCS	30:28	rw	**Timeout Clock selection** 000_B CMU_CLK0 selected 001_B CMU_CLK1 selected 010_B CMU_CLK2 selected 011_B CMU_CLK3 selected 100_B CMU_CLK4 selected 101_B CMU_CLK5 selected 110_B CMU_CLK6 selected 111_B CMU_CLK7 selected
0	31	r	**Reserved** Read as zero, shall be written as zero.

28.13.8.16Register TIM[i]_CH[x]_TDUC

TIMi Channel x TDU Counter Register

The register **TIM[i]_CH[x]_TDUC** is writable if Timeout unit is disabled (TOCTRL=0b00).

If USE_LUT != 0b00 (input signal generation by lookup table) the bit field TO_CNT2 is writable at any time, TO_CNT,TO_CNT1 will not be changed.

TIMi_CHx_TDUC (i=0-7;x=0-7)

TIMi Channel x TDU Counter Register (001014_H+i*800_H+x*80_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TO_CNT2							
r								rw							

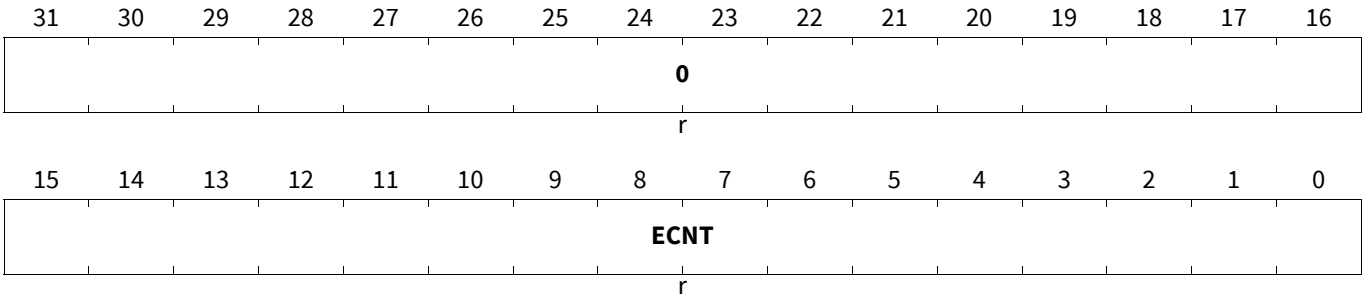
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TO_CNT1								TO_CNT							
rw								rw							

Field	Bits	Type	Description
TO_CNT	7:0	rw	**Current Timeout value slice0 for channel x** SLICING != 0b11: counter will be reset to 0X0 on TDU_RESYNC condition. SLICING = 0b11 : counter will be loaded with TOV2 on TDU_RESYNC condition.
TO_CNT1	15:8	rw	**Current Timeout value slice1 for channel x** Counter will be reset to 0x0 on TDU_RESYNC condition.
TO_CNT2	23:16	rw	**Current Timeout value slice2 for channel x** Counter will be reset to 0x0 on TDU_RESYNC condition.
0	31:24	r	**Reserved** Read as zero, shall be written as zero

28.13.8.17Register TIM[i]_CH[x]_ECNT

TIMi Channel x SMU Edge Counter Register

TIMi_CHx_ECNT (i=0-7;x=0-7)
TIMi Channel x SMU Edge Counter Register($00100C_H$+i*800_H+x*80_H) Application Reset Value: 0000 0000_H

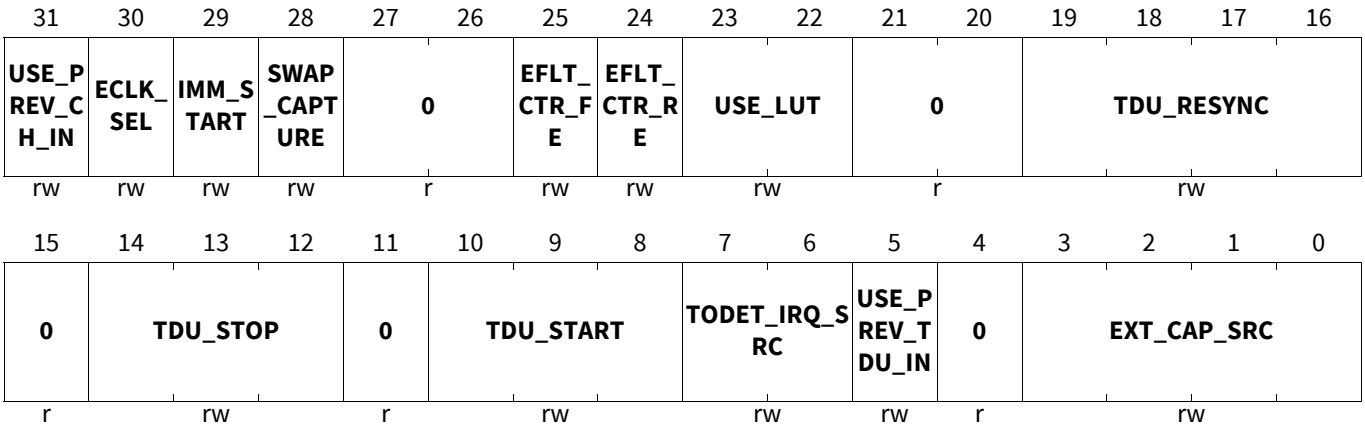


Field	Bits	Type	Description
ECNT	15:0	r	**Edge counter** If TIM channel is disabled the content of ECNT gets frozen. A read will auto clear the bits [15:1]. Further read accesses to ECNT will show on Bit 0 the actual input signal value of the channel.
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.13.8.18Register TIM[i]_CH[x]_ECTRL

TIMi Channel x Extended Control Register

TIMi_CHx_ECTRL (i=0-7;x=0-7)
TIMi Channel x Extended Control Register(001028_H+i*800_H+x*80_H) Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
EXT_CAP_SRC	3:0	rw	**Defines selected source for triggering the EXT_CAPTURE functionality** 0_H NEW_VAL_IRQ of following channel selected 1_H AUX_IN selected 2_H CNTOFL_IRQ of following channel selected 3_H CICTRL = 1: use signal TIM_IN(x) as input for channel x / CICTRL = 0: use signal TIM_IN(x-1) as input for channel x (or TIM_IN(m-1) if x is 0) 4_H ECNTOFL_IRQ of following channel selected 5_H TODET_IRQ of following channel selected 6_H GLITCHDET_IRQ of following channel selected 7_H GPROFL_IRQ of following channel selected 8_H cmu_clk selected by CLK_SEL of following channel 9_H REDGE_DET of following channel selected A_H FEDGE_DET of following channel selected B_H Logical OR of (FEDGE_DET, REDGE_DET) of following channel selected C_H tdu_sample_evt of local TDU selected D_H tdu_word_evt of local TDU selected E_H tdu_frame_evt of local TDU selected F_H Reserved
USE_PREV_TDU_IN	5	rw	**Select input data source for TDU** 0_B Use input data of local filter for TDU 1_B Use input data of previous channel (after filter unit) for TDU
TODET_IRQ_SRC	7:6	rw	**selection of source for TODET_IRQ** With TODET_IRQ_SRC=0b00 the ACB bit 2 will be driven by signal tdu_timeout_evt, if TODET_IRQ_SRC!=0b00 ACB2 will be 0. 00_B Use tdu_timeout_evt 01_B Use tdu_word_evt 10_B Use tdu_frame_evt 11_B Use tdu_sample_evt

Field	Bits	Type	Description
TDU_START	10:8	rw	**Defines condition which will start the TDU unit** In mode SLICING=0b11 every start/restart will load the TO_CNT with value TOV2. Note: tdu_start_000_event is defined as: Each writing of TOCTRL != 0 (independent of current TOCTRL) while TDU_START=0b000 and TDU is stopped (initially or stopped by TDU_STOP event). This event will last 1 system clock cycle. 000_B Start once immediate on tdu_start_000_event 001_B Start once with occurrence of first cmu_clk selected by CLK_SEL when measure unit is enabled by TIM_EN=1 010_B Start once with occurrence of first active edge selected by TOCTRL; restart on tdu_frame_evt if TDU is stopped 011_B Start once with occurrence of first active edge selected by TOCTRL 100_B Start/restart with occurrence of external capture event (if TDU is stopped, restart again) 101_B Start/restart with occurrence of first cmu_clk selected by CLK_SEL when measure unit is enabled by TIM_EN=1 (if TDU is stopped, restart again) 110_B Start once with occurrence of external capture event; restart on tdu_frame_evt if TDU is stopped 111_B Start/restart with occurrence of first active edge selected by TOCTRL (if TDU is stopped, restart again)
TDU_STOP	14:12	rw	**Defines condition which will stop the TDU unit** Note: tdu_toctrl_0_event is defined as: Each writing of TOCTRL = 0 (independent of current TOCTRL) while TDU is started. This event will last 1 system clock cycle. 000_B Immediate stop counting of TDU on tdu_toctrl_0_event (see Note) 001_B Stop counting of TDU on tdu_word_evt or on tdu_toctrl_0_event (see Note) 010_B Stop counting of TDU on tdu_frame_evt or on tdu_toctrl_0_event (see Note) 011_B Stop counting of TDU on tdu_timeout_evt or on tdu_toctrl_0_event (see Note) 100_B Stop counting of TDU on external capture event or on tdu_toctrl_0_event (see Note)
TDU_RESYNC	19:16	rw	**Defines condition which will resynchronize the TDU unit** Encoding see **Table 46** and **Table 47**.
USE_LUT	23:22	rw	**Generate Filter input by lookup table** 00_B Lookup table not in use, lut_in0(x) used as filter input 01_B Use 3-bit lookup table with index = ext_capture(x) & lut_in1(x) & lut_in0(x). Filter input is defined by TO_CNT2[index]. 10_B Use 3-bit lookup table with index = fout_prev(x) & lut_in1(x) & lut_in0(x). Filter input is defined by TO_CNT2[index]. 11_B Use 3-bit lookup table with index = tssm_out(x) & lut_in1(x) & lut_in0(x). Filter input is defined by TO_CNT2[index].
EFLT_CTR_RE	24	rw	**Extension of bit field FLT_CTR_RE** Details described in FLT_CTR_RE bit field of register TIM[i]_CH[x]_CTRL.

Field	Bits	Type	Description
EFLT_CTR_FE	25	rw	**Extension of bit field FLT_CTR_FE** Details described in FLT_CTR_FE bit field of register TIM[i]_CH[x]_CTRL.
SWAP_CAPTURE	28	rw	**Swap point of time of capturing CNTS and GPR1** This bit is only applicable in TPWM and TPIM mode. Set to 0 in all other modes. 0_B Inactive edge will capture data in CNTS; NEWVAL_IRQ event will capture data in GPR1 1_B Swap time of capture: inactive edge will capture data in GPR1; NEWVAL_IRQ event will capture data in CNTS
IMM_START	29	rw	**Start immediately the measurement** This bit is only applicable in TPWM and TPIM mode. Set to 0 in all other modes. 0_B Start with first active edge the measurement 1_B Start immediately after enable (TIM_EN=1) the measurement
ECLK_SEL	30	rw	**Extension of bit field CLK_SEL** Details described in CLK_SEL bit field of register TIM[i]_CH[x]_CTRL.
USE_PREV_CH_IN	31	rw	**Select input data source for TIM channel** 0_B Use input data of local filter unit for channel measurements 1_B Use input data of previous channel (after filter unit) for channel measurements
0	4, 11, 15, 21:20, 27:26	r	**Reserved** Read as zero, shall be written as zero.

Table 46 Behavior of TDU_RESYNC with SLICING != 0b11

TDU_RESYNC	Behavior
0000_B	reset counter TO_CNT2 on each active edge selected by TOCTRL or tdu_timeout_evt or on tdu_start_000_event [1]; reset counters TO_CNT,TO_CNT1 on tdu_timeout_evt or on tdu_start_000_event [1]; if SLICING=0b10 and TO_CTRL=0b-1 then reset TO_CNT on rising input edge; if SLICING=0b10 and TO_CTRL=0b1- then reset TO_CNT1 on falling input edge; if SLICING!=0b10 then reset counters TO_CNT,TO_CNT1 on each active edge selected by TOCTRL
$0\text{--}1_B$	if SLICING=0b10 and TO_CTRL=0bx1 then reset TO_CNT on rising input edge; if SLICING=0b10 and TO_CTRL=0b1- then reset TO_CNT1 on falling input edge; if SLICING!=0b10 then reset counters TO_CNT,TO_CNT1 on each active edge selected by TOCTRL; if SLICING=0b00 then reset TO_CNT2 on each active edge selected by TOCTRL
$0x1\text{-}_B$	reset counters TO_CNT on tdu_word_evt;
01--_B	reset counter TO_CNT1 on tdu_frame_evt; if SLICING=0b01 then reset TO_CNT on tdu_frame_evt
1000_B	reset counters TO_CNT,TO_CNT1,TO_CNT2 on event selected by EXT_CAP_SRC
1---_B	if SLICING!=0b00 then reset counter TO_CNT2 on tdu_sample_evt

Table 46 Behavior of TDU_RESYNC with SLICING != 0b11 (cont'd)

TDU_RESYNC	Behavior
1--1 $_B$	reset counter TO_CNT2 on each active edge selected by TOCTRL; if SLICING=0b10 and TO_CTRL=0b-1 then reset TO_CNT on rising input edge; if SLICING=0b10 and TO_CTRL=0b1- then reset TO_CNT1 on falling input edge; if SLICING!=0b10 then reset counters TO_CNT,TO_CNT1 on each active edge selected by TOCTRL
1-1-$_B$	reset counters TO_CNT on tdu_word_evt
11-- $_B$	reset counter TO_CNT1 on tdu_frame_evt; if SLICING=0b01 then reset TO_CNT on tdu_frame_evt

1) tdu_start_000_event is defined as: Each writing of TOCTRL != 0 (independent of current TOCTRL) while TDU_START=0b000 and TDU is stopped (initially or stopped by TDU_STOP event). This event will last 1 system clock cycle.

Table 47 Behavior of TDU_RESYNC with SLICING = 0b11

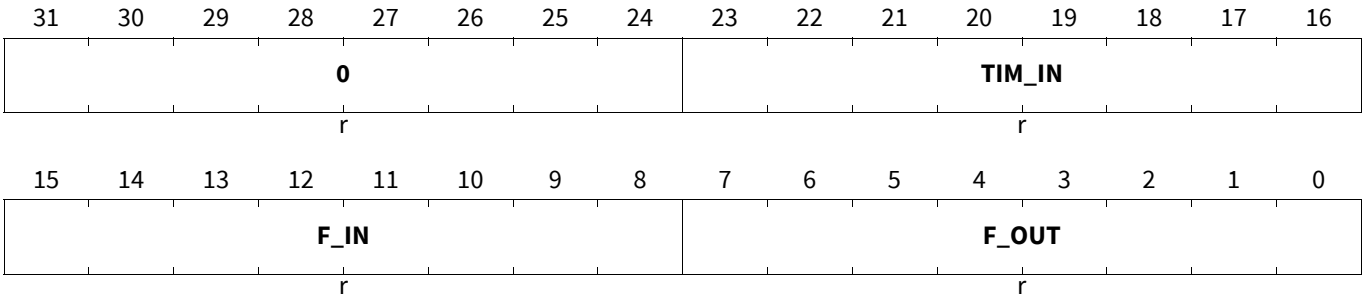
TDU_RESYNC	Behavior
0000_B	load counter TO_CNT with TOV2 on each active edge selected by TOCTRL or tdu_timeout_evt or on tdu_start_000_event [1]; reset counter TO_CNT1 on each active edge selected by TOCTRL or tdu_timeout_evt or on tdu_start_000_event [1]
---1 $_B$	load counter TO_CNT with TOV2 on each active edge selected by TOCTRL; reset counter TO_CNT1 on each active edge selected by TOCTRL
0-1- $_B$	load counter TO_CNT with TOV2 on tdu_word_evt
1-1-$_B$	reset counter TO_CNT on tdu_word_evt
-1--$_B$	reset counter TO_CNT1 on tdu_frame_evt
1000_B	load counter TO_CNT with TOV2; reset counter TO_CNT1 on event selected by EXT_CAP_SRC

28.13.8.19Register TIM[i]_INP_VAL

TIMi Input Value Observation Register

TIMi_INP_VAL (i=0-7)

TIMi Input Value Observation Register **(001074_H+i*800_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
F_OUT	7:0	r	**Signals after TIM FLT unit**
F_IN	15:8	r	**Signals after INPSRC selection, before TIM FLT unit**

Field	Bits	Type	Description
TIM_IN	23:16	r	**Signals after TIM input signal synchronization**
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.14 Timer Output Module (TOM)

28.14.1 Overview

The Timer Output Module (TOM) offers up to 16 independent channels (index x) to generate simple PWM signals at each output pin *TOM[i]_CH[x]_OUT*.

Additionally, at TOM output *TOM[i]_CH15_OUT* a pulse count modulated signal can be generated.

The architecture of the TOM sub-module is depicted in **Figure 47**.

Indices and their range as used inside this chapter are:

y=0,1

z=0…7

The following design variables are used inside this chapter. Please refer to AURIX device specific appendix for correct value.

cCTO: TOM channel count; number of channels per instance - 1.

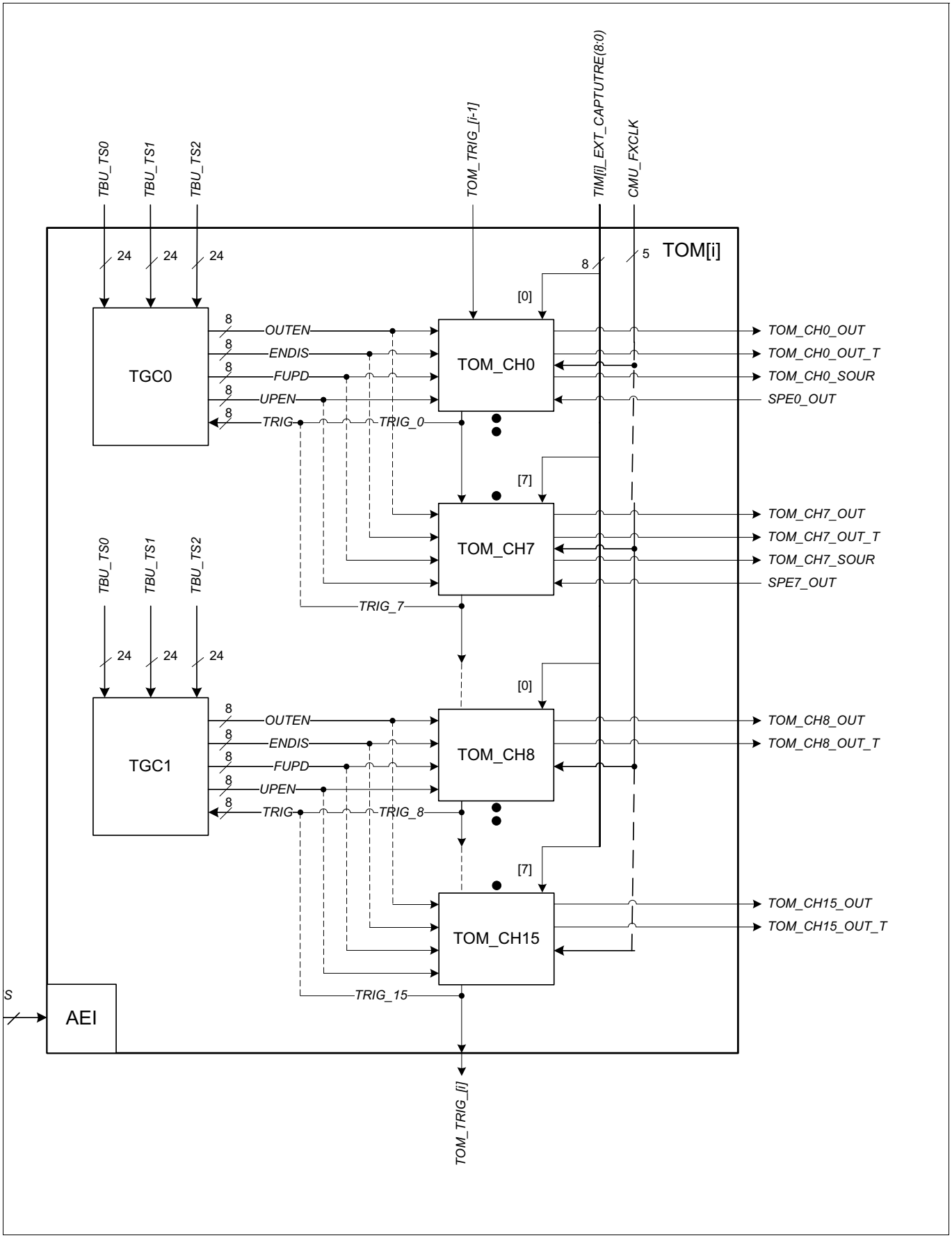


Figure 47 TOM block diagram

The two sub-modules TGC0 and TGC1 are global channel control units that control the enabling/disabling of the channels and their outputs as well as the update of their period and duty cycle register.

The module TOM receives two (three) timestamp values *TBU_TS0*, *TBU_TS1* (and *TBU_TS2*) in order to realize synchronized output behavior on behalf of a common time base.

The 5 dedicated clock line inputs *CMU_FXCLK* are providing divided clocks that can be selected to clock the output pins.

The trigger signal *TOM_TRIG_[i-1]* of TOM instance i comes from the preceding instance i-1, the trigger *TOM_TRIG_[i]* is routed to succeeding instance i+1.Note, TOM0 is connected to its own output *TOM_TRIG_0*, i.e. the last channel of TOM instance 0 can trigger the first channel of TOM instance 0 (this path is registered, which means delayed by one SYS_CLK period).

28.14.2 TOM Global Channel Control (TGC0, TGC1)

28.14.2.1 Overview

There exist two global channel control units (TGC0 and TGC1) to drive a number of individual TOM channels synchronously by external or internal events.

Each TGC[y] can drive up to eight TOM channels where TGC0 controls TOM channels 0 to 7 and TGC1 controls TOM channels 8 to 15.

The TOM sub-module supports four different kinds of signaling mechanisms:

- Global enable/disable mechanism for each TOM channel with control register **TOM[i]_TGC[y]_ENDIS_CTRL** and status register **TOM[i]_TGC[y]_ENDIS_STAT**
- Global output enable mechanism for each TOM channel with control register **TOM[i]_TGC[y]_OUTEN_CTRL** and status register **TOM[i]_TGC[y]_OUTEN_STAT**
- Global force update mechanism for each TOM channel with control register **TOM[i]_TGC[y]_FUPD_CTRL**
- Update enable of the register **CM0**, **CM1** and **CLK_SRC** for each TOM channel with the control bit field **UPEN_CTRL[z]** of **TOM[i]_TGC[y]_GLB_CTRL**

28.14.2.2 TGC Sub-unit

Each of the first three individual mechanisms (enable/disable of the channel, output enable and force update) can be driven by three different trigger sources. The three trigger sources are:

- the host CPU (bit **HOST_TRIG** of register **TOM[i]_TGC[y]_GLB_CTRL**)
- the TBU time stamp (signal *TBU_TS0, TBU_TS1, TBU_TS2* if available)
- the internal trigger signal *TRIG* (bunch of trigger signals *TRIG_[x]* which can be either the trigger *TRIG_CCU0* of channel x, the trigger of preceding channel x-1 (i.e. signal *TRIG_[x-1]*) or the external trigger *TIM_EXT_CAPTURE(t)* of assigned TIM channel t.

The first way is to trigger the control mechanism by a direct register write access via host CPU (bit **HOST_TRIG** of register **TOM[i]_TGC[y]_GLB_CTRL**).

The second way is provided by a compare match trigger on behalf of a specified time base coming from the module TBU (selected by bits **TBU_SEL**) and the time stamp compare value defined in the bit field **ACT_TB** of register **TOM[i]_TGC[y]_ACT_TB**.Note, a signed compare of **ACT_TB** and selected *TBU_TS*[*x*] with x=0,1,2 is performed.

The third possibility is the input *TRIG* (bunch of trigger signals *TRIG_[x]*) coming from the TOM channels 0 to 7 / 8 to 15.

The corresponding trigger signal *TRIG_[x]* coming from channel [x] can be masked by the register **TOM[i]_TGC[y]_INT_TRIG**.

To enable or disable each individual TOM channel, the register **TOM[i]_TGC[y]_ENDIS_CTRL** and/or **TOM[i]_TGC[y]_ENDIS_STAT** have to be used.

The register **TOM[i]_TGC[y]_ENDIS_STAT** controls directly the signal *ENDIS*. A write access to this register is possible.

The register **TOM[i]_TGC[y]_ENDIS_CTRL** is a shadow register that overwrites the value of register **TOM[i]_TGC[y]_ENDIS_STAT** if one of the three trigger conditions matches.

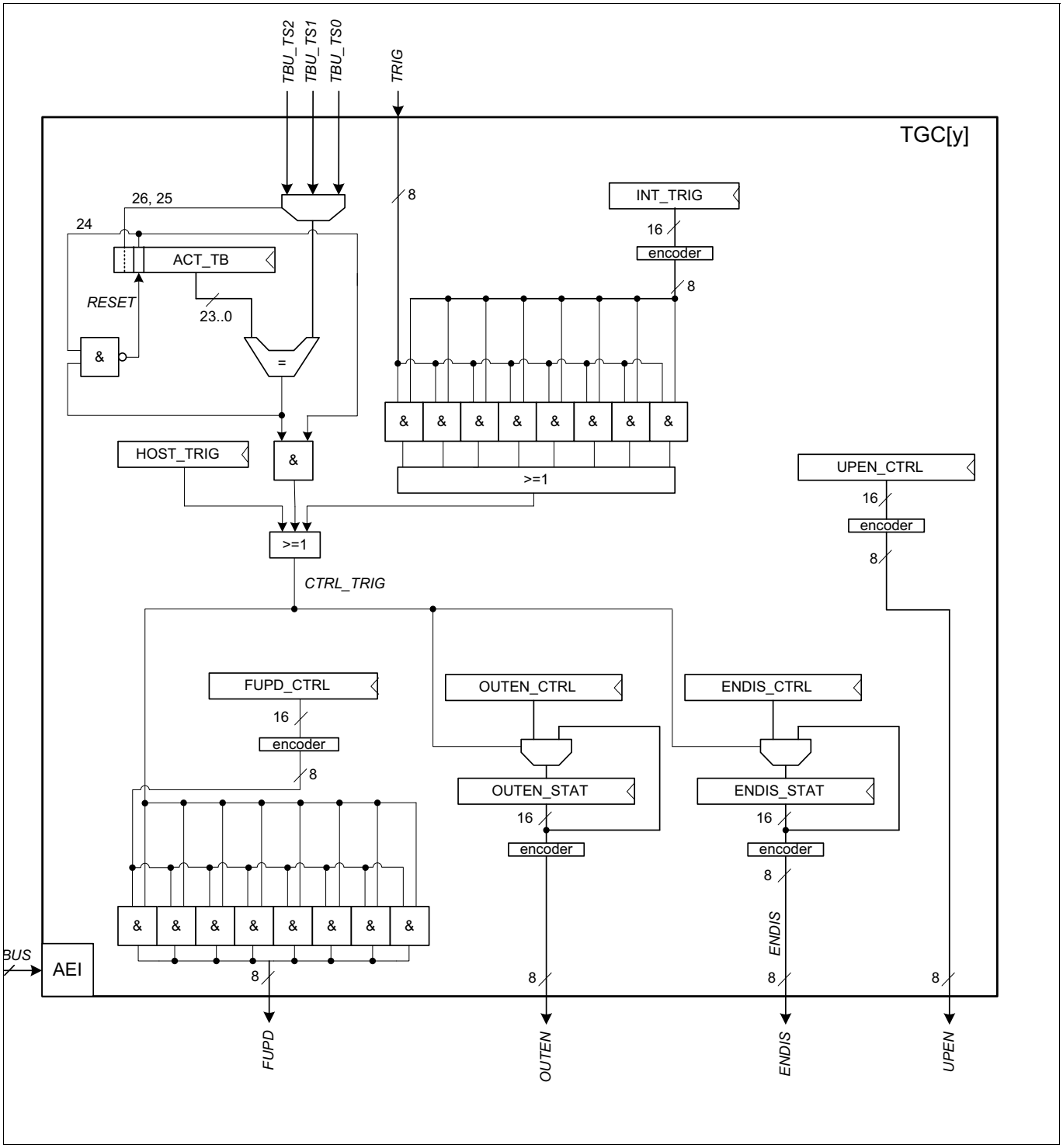


Figure 48 TOM Global channel control mechanism

The output of the individual TOM channels can be controlled using the register **TOM[i]_TGC[y]_OUTEN_CTRL** and **TOM[i]_TGC[y]_OUTEN_STAT**.

The register **TOM[i]_TGC[y]_OUTEN_STAT** controls directly the signal *OUTEN*. A write access to this register is possible.

The register **TOM[i]_TGC[y]_OUTEN_CTRL** is a shadow register that overwrites the value of register **TOM[i]_TGC[y]_OUTEN_STAT** if one of the three trigger conditions matches.

If a TOM channel is disabled by the register **TOM[i]_TGC[y]_OUTEN_STAT**, the actual value of the channel output at *TOM_CH[x]_OUT* is defined by the signal level bit (**SL**) defined in the channel control register **TOM[i]_CH[x]_CTRL**.If the output is enabled, the output at *TOM_CH[x]_OUT* depends on value of flip-flop **SOUR**.

The register **TOM[i]_TGC[y]_FUPD_CTRL** defines which of the TOM channels receive a *FORCE_UPDATE* event if the trigger signal *CTRL_TRIG* is raised.

Note: The force update request is stored and executed synchronized to the selected CMU_FXCLK.

The register bits **UPEN_CTRL[z]** defines for which TOM channel the update of the working register **CM0**, **CM1** and **CLK_SRC** by the corresponding shadow register **SR0**, **SR1** and **CLK_SRC_SR** is enabled. If update is enabled, the register **CM0**, **CM1** and **CLK_SRC** will be updated on reset of counter register **CN0** (see **Figure 49** and **Figure 50**). An exception is the configuration of SR0_TRIG=1 which enable the trigger generation defined by **SR0**. Then **CM0** is not updated with **SR0**.

28.14.3 TOM Channel

Each individual TOM channel comprises a Counter Compare Unit 0 (CCU0) which contains the counter register **CN0** and the period register **CM0**, a Counter Compare Unit 1 (CCU1) which contains the duty cycle register **CM1** and the Signal Output Generation Unit (SOU) which contains the output register **SOUR**. The architecture is depicted in **Figure 49** for channels 0 to 7 and in **Figure 50** for channels 8 to 15.

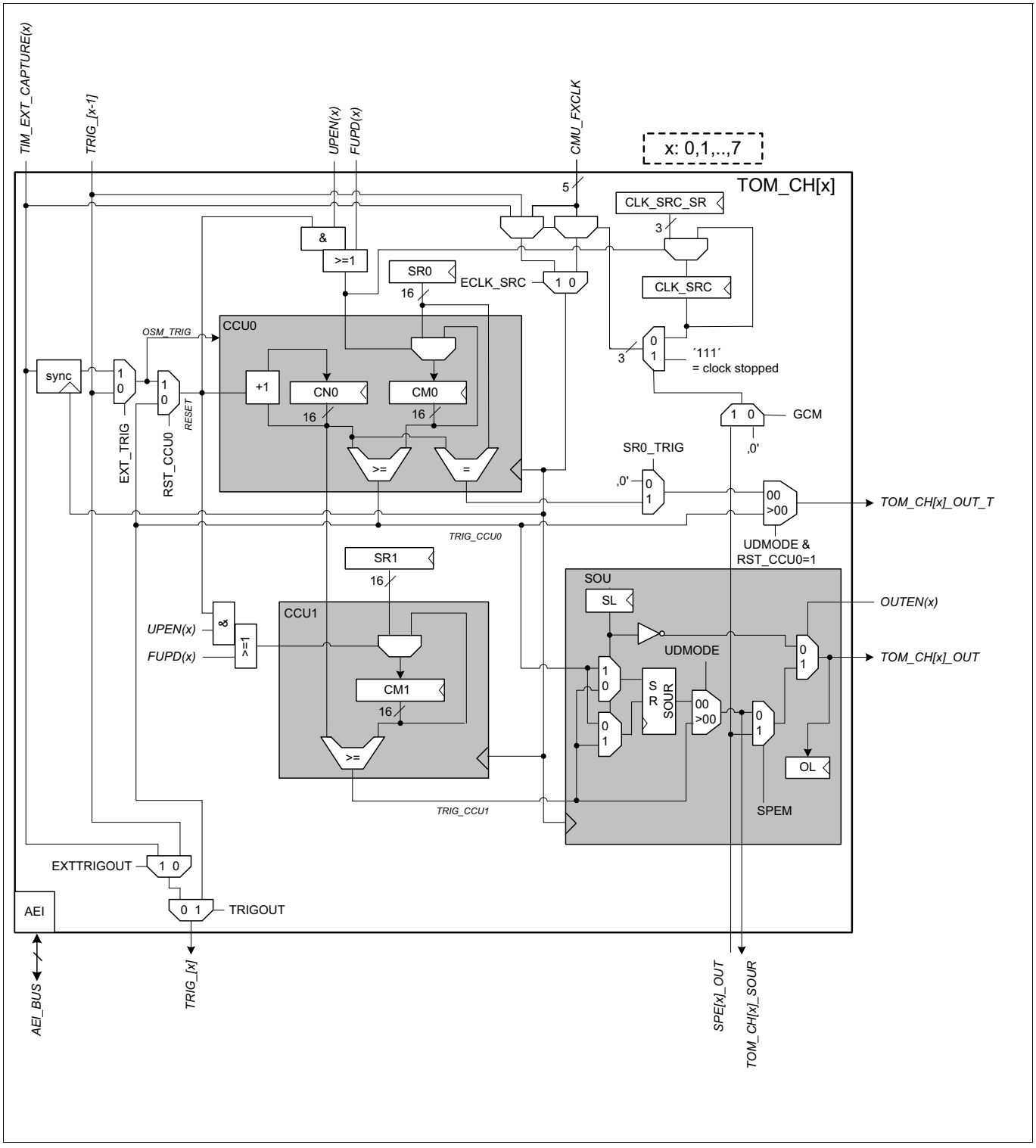


Figure 49 TOM Channel 0…7 architecture

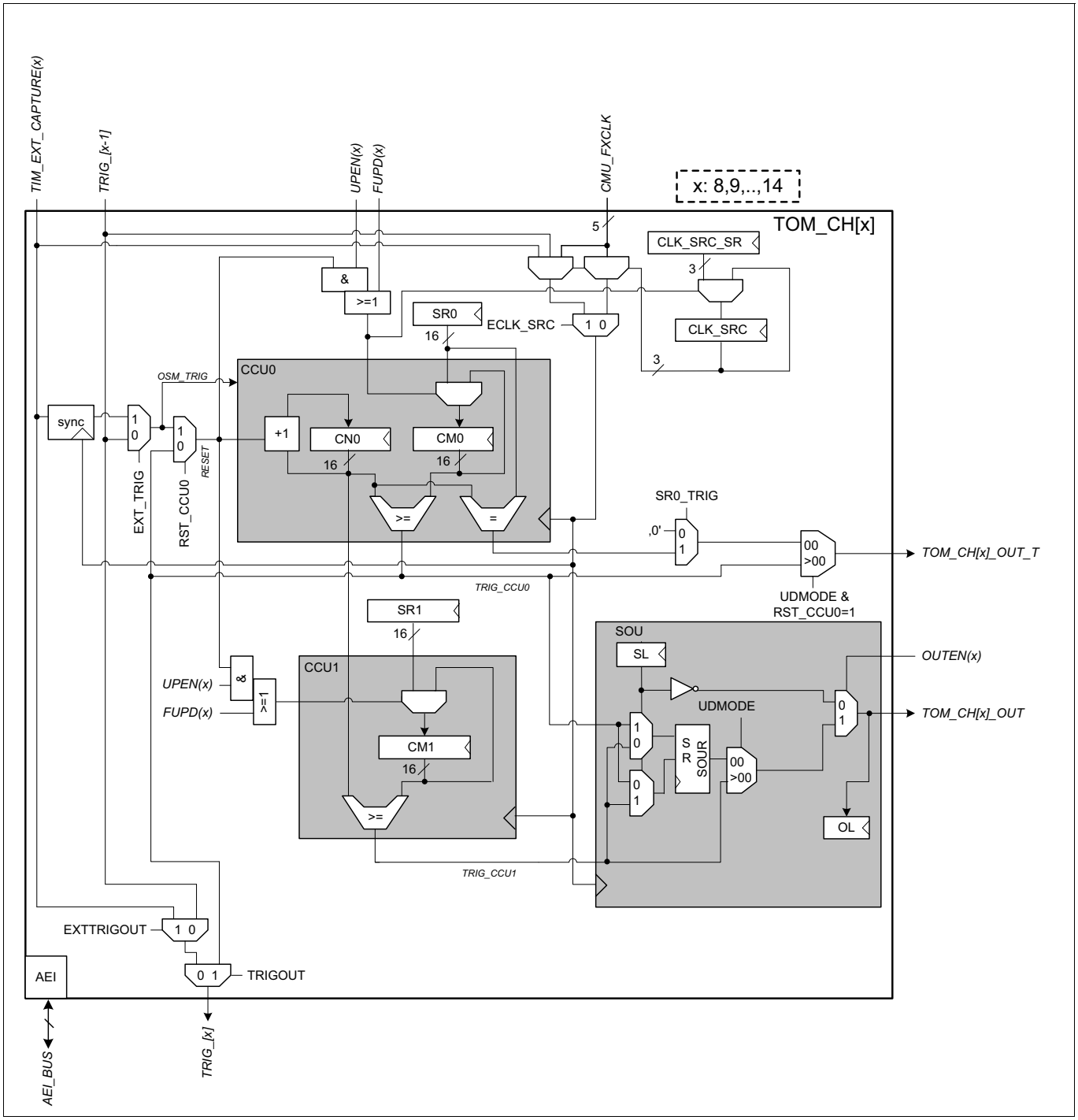


Figure 50 TOM Channel 8...14 architecture

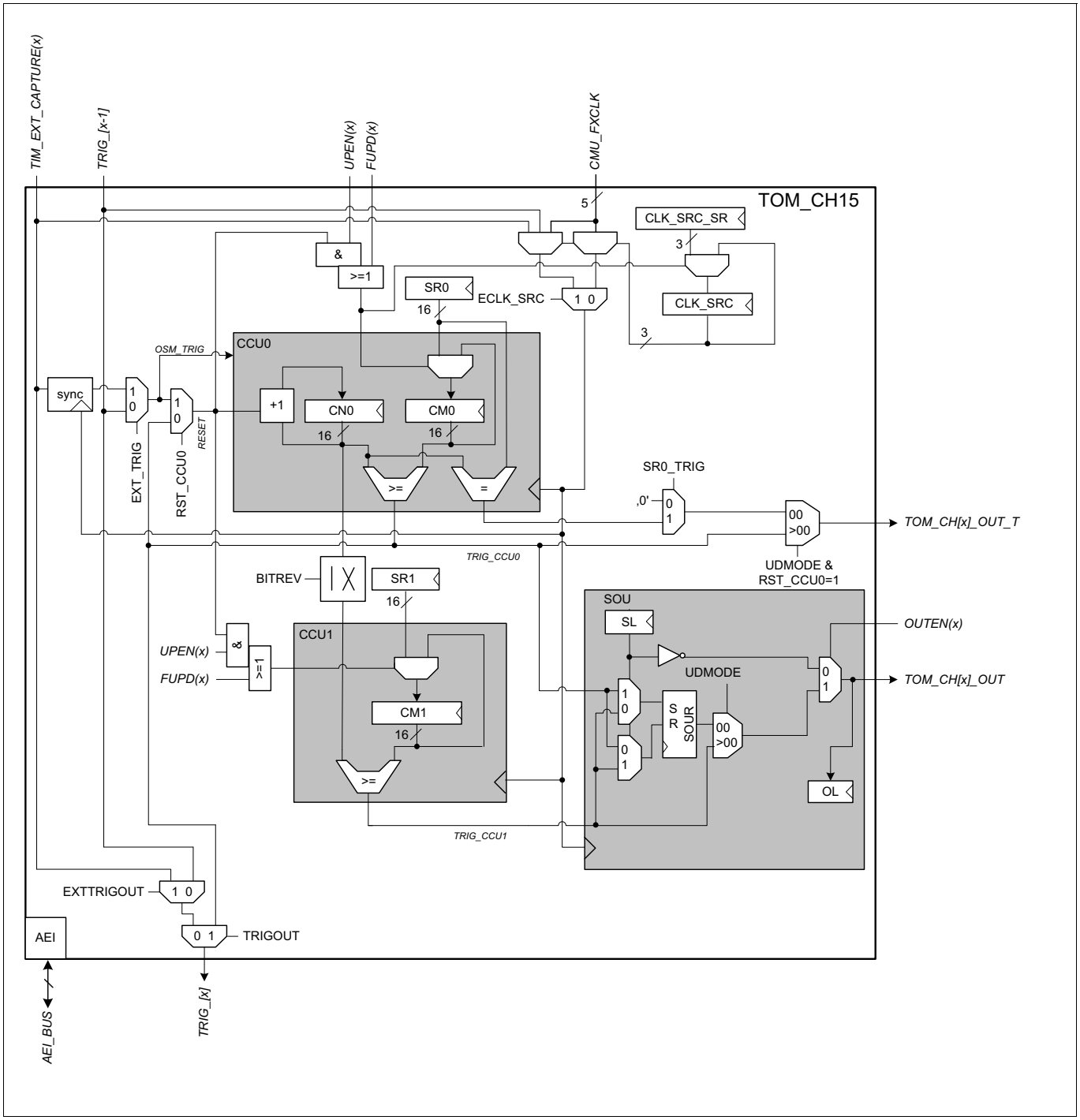


Figure 51 TOM Channel 15 architecture

The CCU0 contains a counter **CN0** which is clocked with one of the selected input frequencies (*CMU_FXCLK*) provided from outside of the sub-module.

Depending on configuration bits RST_CCU0 of register **TOM[i]_CH[x]_CTRL** the counter register **CN0** can be reset either when the counter value is equal to the compare value **CM0** (i.e. CN0 counts only 0 to CM0-1 and is then reset to 0) or when signaled by the TOM[i] trigger signal *TRIG_[x-1]* of the preceding channel [x-1] (which can also be the last channel of preceding instance TOM[i-1]) or the trigger signal *TIM_EXT_CAPTURE(x)* of the assigned TIM channel [x].

Note: As an exception, the input *TRIG_[0]* of instance TOM0 is triggered by its own last channel cCTO via signal *TRIG_[cCTO]*.

When the counter register **CN0** is greater or equal than the value **CM0** (in fact CM0-1) the sub-unit CCU0 triggers the SOU sub-unit and the succeeding TOM sub-module channel (signal *TRIG_CCU0*).

In the sub-unit CCU1 the counter register **CN0** is compared with the value of register **CM1**. If **CN0** is greater or equal than **CM1** the sub-unit CCU1 triggers the SOU sub-unit (signal *TRIG_CCU1*).

If counter register **CN0** of channel x is reset by its own CCU0 unit (i.e. the compare match of **CN0**>=**CM0**-1 configured by RST_CCU0=0), following statements are valid:

- **CN0** counts from 0 to **CM0**-1 and is then reset to 0.
- When CN0 is reset from CM0 to 0, an edge to SL is generated
- When CN0 is incrementing and reaches CN0 > CM1, an edge to !SL is generated.
- if **CM0**=0 or **CM0**=1, the counter **CN0** is constant 0.
- if **CM1**=0, the output is !SL = 0% duty cycle
- if **CM1** >= **CM0** and **CM0**>1, the output is SL = 100% duty cycle

If the counter register **CN0** of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by RST_CCU0=1), following statements are valid:

- **CN0** counts from 0 to MAX-1 and is then reset to 0 by trigger signal
- **CM0** defines the edge to SL value, **CM1** defines the edge to !SL value.
- if **CM0**=**CM1**, the output switches to SL if **CN0**=**CM0**=**CM1** (**CM0** has higher priority)
- if **CM0**=0 and **CM1**=MAX, the output is SL = 100% duty cycle
- if **CM0** > MAX, the output is !SL = 0% duty cycle, independent of **CM1**.

The hardware ensures that for both 0% and 100% duty cycle no glitch occurs at the output of the TOM channel.

The SOU sub-unit is responsible for output signal generation. On a trigger *TRIG_CCU0* from sub-unit CCU0 or *TRIG_CCU1* from sub-unit CCU1 an SR flip-flop of sub-unit SOU is either set or reset. If it is set or reset depends on the configuration bit **SL** of the control register **TOM[i]_CH[x]_CTRL**. The initial signal output level for the channel is the reverse value of the bit **SL**.

Figure 54 clarifies the PWM output behavior with respect to the **SL** bit definition. The output level on the TOM channel output pin *TOM[i]_CH[x]_OUT* is captured in bit **OL** of register **TOM[i]_CH[x]_STAT**.

28.14.3.1 Duty cycle, Period and Clock Frequency Update Mechanisms

The two action register **CM0** and **CM1** can be reloaded with the content of the shadow register **SR0** and **SR1**. The register **CLK_SRC** that determines the clock frequency of the counter register **CN0** can be reloaded with its shadow register **CLK_SRC_SR** (bit field in register **TOM[i]_CH[x]_CTRL**).

The update of the register **CM0**, **CM1** and **CLK_SRC** with the content of its shadow register is done when the reset of the counter register **CN0** is requested (via signal *RESET*). This reset of **CN0** is done if the comparison of **CN0** greater or equal than **CM0** is true or when the reset is triggered by another TOM channel [x-1] via the signal *TRIG_[x-1]* or when signaled via the signal *TIM_EXT_CAPTURE(x)* of the assigned TIM channel [x].

With the update of the register **CLK_SRC** at the end of a period a new counter **CN0** clock frequency can easily be adjusted.

In case of RST_CCU0=1 and update enabled by UPEN_CTRL[z] the register CM0, CM1 and CLK_SRC will be updated when CN0 is reset.

An update of duty cycle, period and counter **CN0** clock frequency becoming effective synchronously with start of a new period can easily be reached by performing following steps:

1. disable the update of the action register with the content of the corresponding shadow register by setting the channel specific configuration bit **UPEN_CTRL[z]** of register **TOM[i]_TGC[y]_GLB_CTRL** to '0'.
2. write new desired values to **SR0**, **SR1**, **CLK_SRC_SR**

3. enable update of the action register by setting the channel specific configuration bit **UPEN_CTRL[z]** of register **TOM[i]_TGC[y]_GLB_CTRL** to '1'.

28.14.3.1.1 Synchronous Update Of Duty Cycle Only

A synchronous update of only the duty cycle can be done by simply writing the desired new value to register **SR1** without preceding disable of the update mechanism (as described in the chapter above). The new duty cycle is then applied in the period following the period where the update of register **SR1** was done.

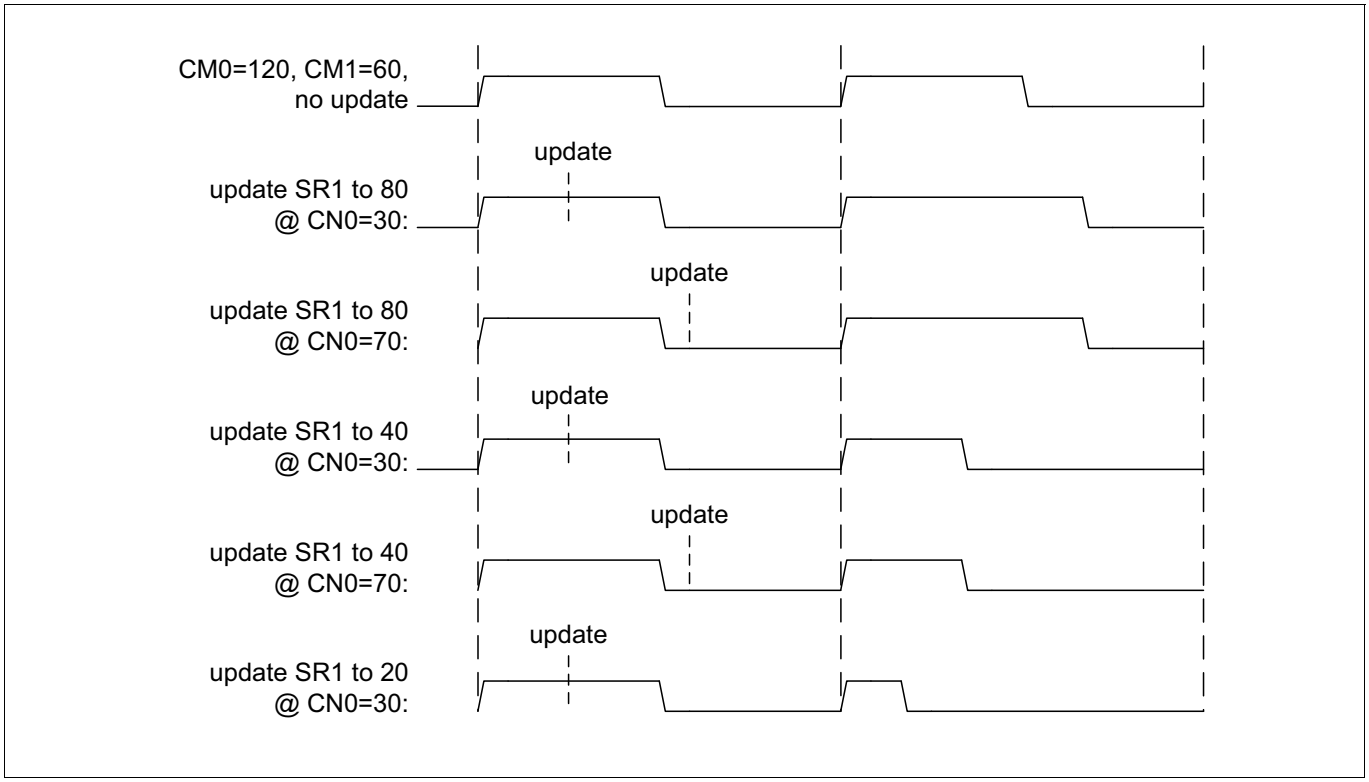


Figure 52 Synchronous update of duty cycle

28.14.3.1.2 Asynchronous Update Of Duty Cycle Only

If the update of the duty cycle should be performed independent of the start of a new period (asynchronous), the desired new value can be written directly to register **CM1**. In this case it is recommended to additionally either disable the synchronous update mechanism as a whole (i.e. clearing bits **UPEN_CTRL[z]** of corresponding channel [x] in register **TOM[i]_TGX[y]_GLB_CTRL**) or updating **SR1** with the same value as **CM1** before writing to **CM1**.

Depending on the point of time of the update of **CM1** in relation to the actual value of **CN0** and **CM1**, the new duty cycle is applied in the current period or the following period (see **Figure 53**). In any case the creation of glitches are avoided. The new duty cycle may jitter from update to update by a maximum of one period (given by **CM0**). However, the period remains unchanged.

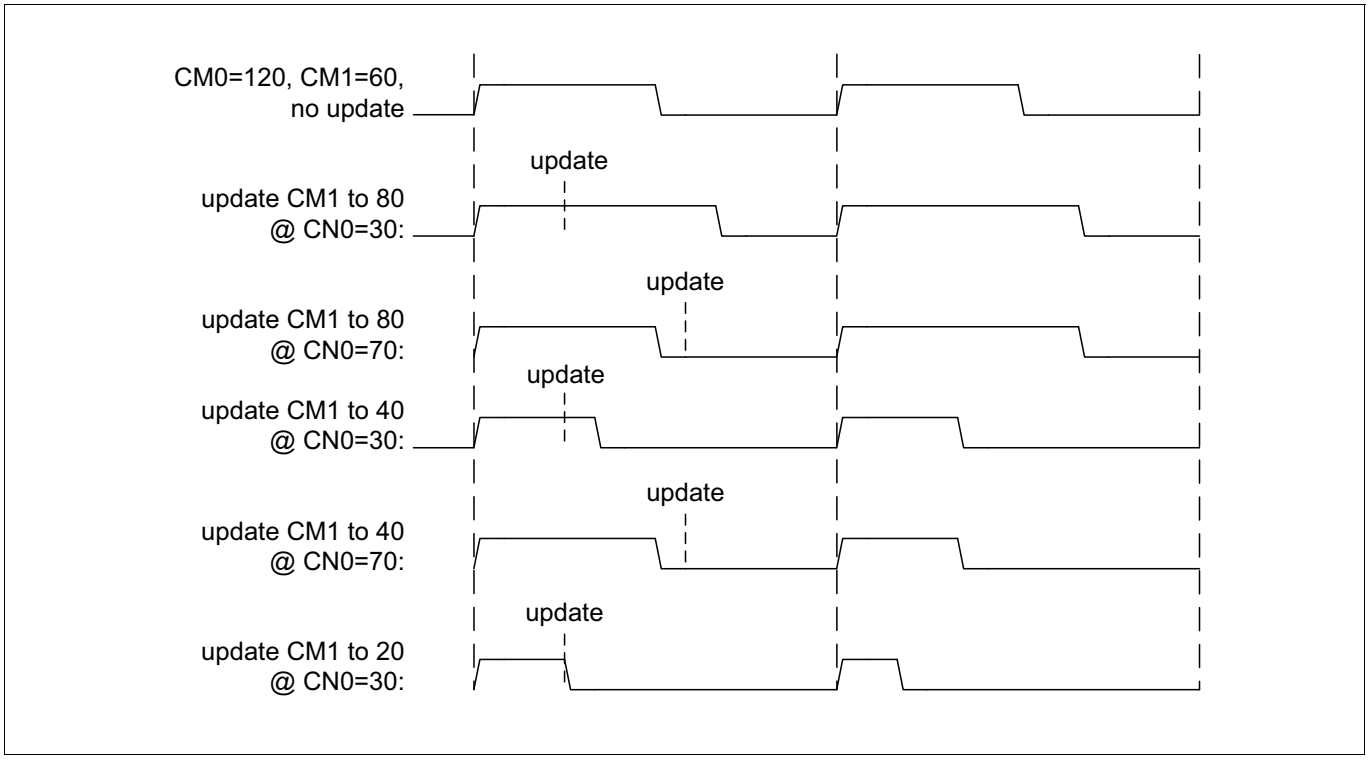


Figure 53 Asynchronous update of duty cycle

28.14.3.2 Continuous Counting Up Mode

In continuous mode the TOM channel starts incrementing the counter register **CN0** once it is enabled by setting the corresponding bits in register **TOM[i]_TGC[y]_ENDIS_STAT** (refer to **Section 28.14.2.2** for details of enabling a TOM channel).

The signal level of the generated output signal can be configured with the configuration bit **SL** of the channel configuration register **TOM[i]_CH[x]_CTRL**.

If the counter **CN0** is reset from **CM0** back to zero, the first edge of a period is generated at *TOM[i]_CH[x]_OUT*.

The second edge of the period is generated if **CN0** has reached **CM1**.

Every time the counter **CN0** has reached the value of **CM0** it is reset back to zero and proceeds with incrementing.

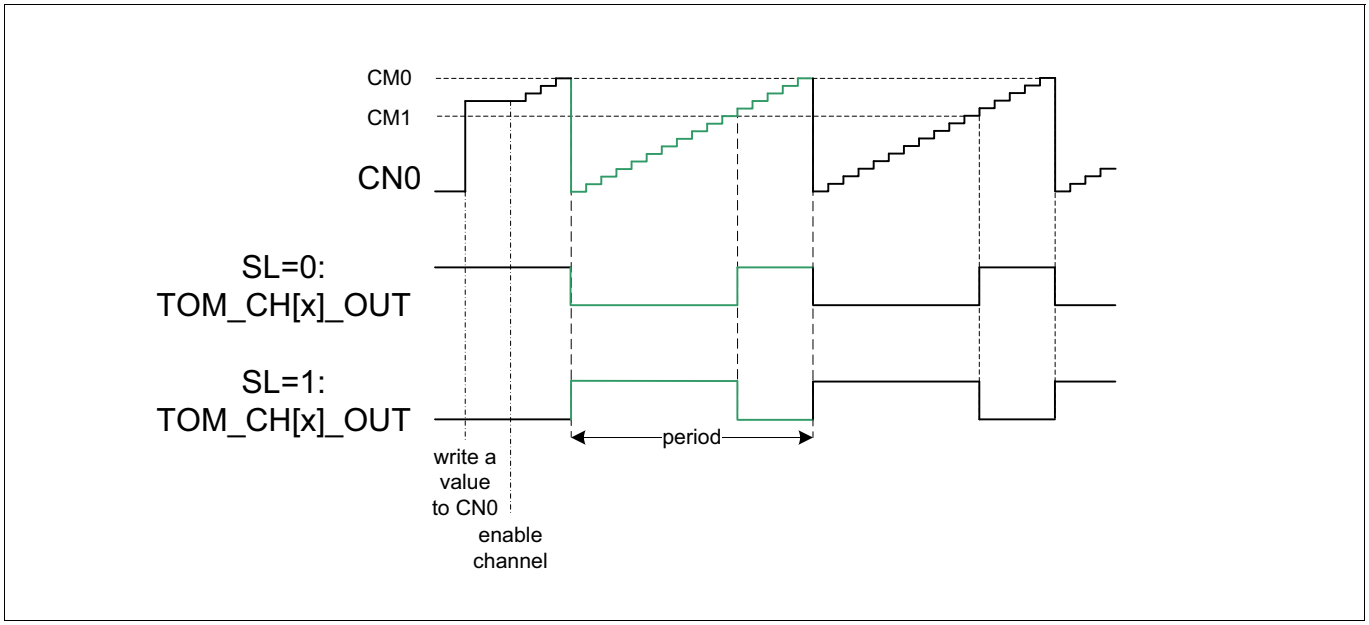


Figure 54 PWM Output with respect to configuration bit SL in continuous mode

28.14.3.3 Continuous Counting Up-Down Mode

In continuous mode, if **CN0** counts up and down (UDMODE != 0b00), depending on configuration bits RST_CCU0 of register **TOM[i]_CH[x]_CTRL** the counter register **CN0** changes direction either when the counter value is equal to the compare value **CM0,** has counted down to 0 or when triggered by the TOM[i] trigger signal *TRIG_[x-1]* of the preceding channel [x-1] (which can also be the last channel of preceding instance TOM[i-1]) or the trigger signal *TIM_EXT_CAPTURE(x)* of the assigned TIM channel [x].

In this case, if UPEN_CTRL[x]=1, also the working register **CM0**, **CM1** and **CLK_SRC** are updated depending on UDMODE.

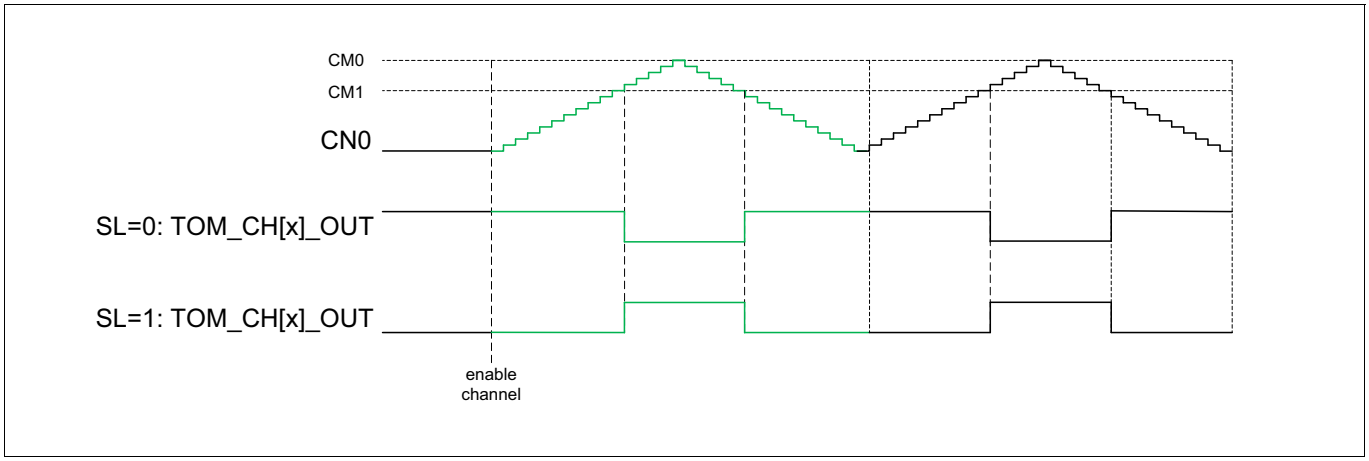


Figure 55 PWM Output behavior with respect to the SL bit in the ATOM[i]_CH[x]_CTRL register if UDMODE != 0b00

The clock of the counter register **CN0** can be one of the CMU clocks CMU_FXCLKx.The clock for **CN0** is defined by CLK_SRC_SR value in register **TOM[i]_CH[x]_CTRL.**The duration of a period in multiples of selected **CN0** counter clock ticks is defined by the **CM0** configuration value (i.e. **CM0** defines half of period in up-down mode).**CM1** defines the duty cycle value in clock ticks of selected **CN0** counter clock **CM0** defines half of duty cycle in up-down mode).

If counter register **CN0** of channel x is reset by its own CCU0 unit (i.e. the compare match of **CN0**>=**CM0** configured by RST_CCU0=0), following statements are valid:

- **CN0** counts continuously first up from 0 to **CM0**-1 and then down to 0.
- if **CN0** >= **CM1**, the output is set to SL
- if **CM1**=0, the output is SL (i.e. 100% duty cycle)
- if **CM1**>= **CM0**, the output is !SL (i.e. 0% duty cycle)
- On output *TOM[i]_CHx]_OUT* a PWM signal is generated. The period is defined by **CM0**, the duty cycle is defined by **CM1**.

This behavior is depicted in **Figure 55**.

If the counter register **CN0** of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by RST_CCU0=1), following statements are valid:

- **CN0** counts continuously first up. On a trigger signal the counter switches to count down mode. If <**CN0** has reached 0, it switches to count up mode.
- if **CN0** >= **CM1**, the output is set to SL
- if **CM1**=0, the output is SL (i.e. 100% duty cycle)
- if **CM1** >= **CM0**, the output is !SL (i.e. 0% duty cycle)
- On output **TOM[i]_CHx]_OUT** a PWM signal is generated. The period is defined by the CCU0 trigger of triggering channel, the duty cycle is defined by **CM1**.
- On output **TOM[i]_CHx]_OUT** a PWM signal is generated. The period is defined by the CCU0 trigger of triggering channel, the duty cycle is defined by **CM0**.

This behavior is depicted in figure **Figure 56**.

Note that in case of up-down counter mode and RST_CCU0=1 it is recommended that:

- the triggering channel and the triggered channel are both running in up-down mode,
- the time between two trigger signals is equal to the time needed for CN0 of triggered to count back to 0 and again up to the same upper value.

The second recommendation can be reached by synchronizing the start of triggering channel and triggered channel,i.e. let both channels start with CN0 value 0. Note that if there is a synchronization register in the trigger chain (indicated by value TOM_TRIG_CHAIN in register CCM[i]_HW_CONF), the additional delay of the trigger by one clock period has to be taken into account by starting at triggering channel with a CN0 value 1 (+1 compared to CN0 of triggered channel).

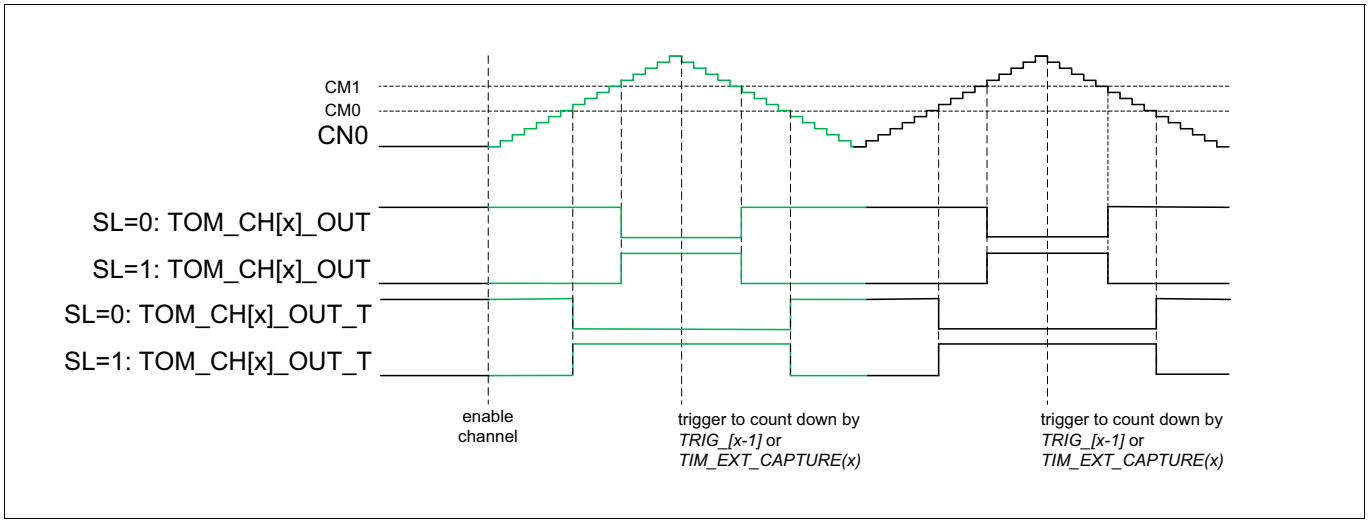


Figure 56 PWM Output behavior in case of RST_CCU0=1 and UDMODE != 0b00

28.14.3.4 One-shot Counting Up Mode

In one-shot mode, the TOM channel generates one pulse with a signal level specified by the configuration bit **SL** in the channel [x] configuration register **TOM[i]_CH[x]_CTRL**.

First the channel has to be enabled by setting the corresponding **TOM[i]_TGC[y]_ENDIS_STAT** value and the one-shot mode has to be enabled by setting bit **OSM** in register **TOM[i]_CH[x]_CTRL**.

In one-shot mode the counter **CN0** will not be incremented once the channel is enabled.

A write access to the register **CN0** triggers the start of pulse generation (i.e. the increment of the counter register **CN0**).

If SPE mode of TOM[i] channel 2 is enabled (set bit **SPEM** of register **TOM[i]_CH2_CTRL**), also the trigger signal *SPE[i]_NIPD* can trigger the reset of register **CN0** to zero and a start of the pulse generation.

The new value of **CN0** determines the start delay of the first edge. The delay time of the first edge is given by (**CM0-CN0**) multiplied with period defined by current value of **CLK_SRC**.

If the counter **CN0** is reset from **CM0** back to zero, the first edge at *TOM[i]_CH[x]_OUT* is generated.

To avoid an update of **CMx** register with content of **SRx** register at this point in time, the automatic update should be disabled by setting UPEN_CTRL[x] = 00 (in register **TOM[i]_TGC[y]_GLB_CTRL**)

The second edge is generated if **CN0** is greater or equal than **CM1** (i.e. **CN0** was incremented until it has reached **CM1** or **CN0** is greater than **CM1** after an update of **CM1**).

If the counter **CN0** has reached the value of **CM0** a second time, the counter stops.

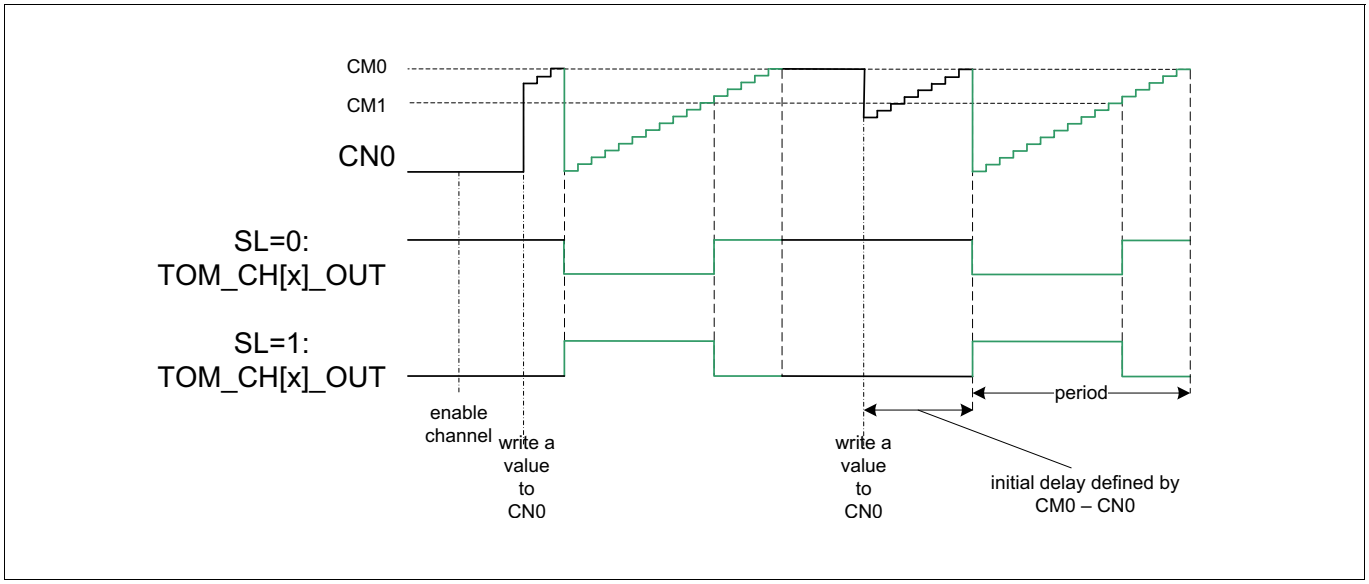


Figure 57 PWM Output with respect to configuration bit SL in one-shot mode: trigger by writing to CN0

Further output of single periods can be started by a write access to register CN0.

If CN0 is already incrementing (i.e. started by writing to CN0 a value CN0start < CM0), the affect of a second write access to CN0 depends on the phase of CN0:

phase 1: update of CN0 before CN0 reaches first time CM0

phase 2: update of CN0 after CN0 has reached first time CM0 but is less than CM1

phase 3: update of CN0 after CN0 has reached first time CM0 and CN0 is greater than or equal CM1

In phase 1: writing to counter CN0 a value CN0new < CM0 leads to a shift of first edge (generated if CN0 reaches CM0 first time) by the time CM0-CN0new.

In phase 2: writing to incrementing counter CN0 a value CN0new < CM1 while CN0old is below CM1 leads to a lengthening of the pulse. The counter CN0 stops if it reaches CM0.

In phase 3: Writing to incrementing counter CN0 a value CN0new while CN0old is already greater than or equal CM1 leads to an immediate restart of a single pulse generation inclusive the initial delay defined by CM0 - CN0new.

If a channel is configured to one-shot mode and configuration bit OSM_TRIG is set to 1, the trigger signal *OSM_TRIG* (i.e. *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*) triggers start of one pulse generation.

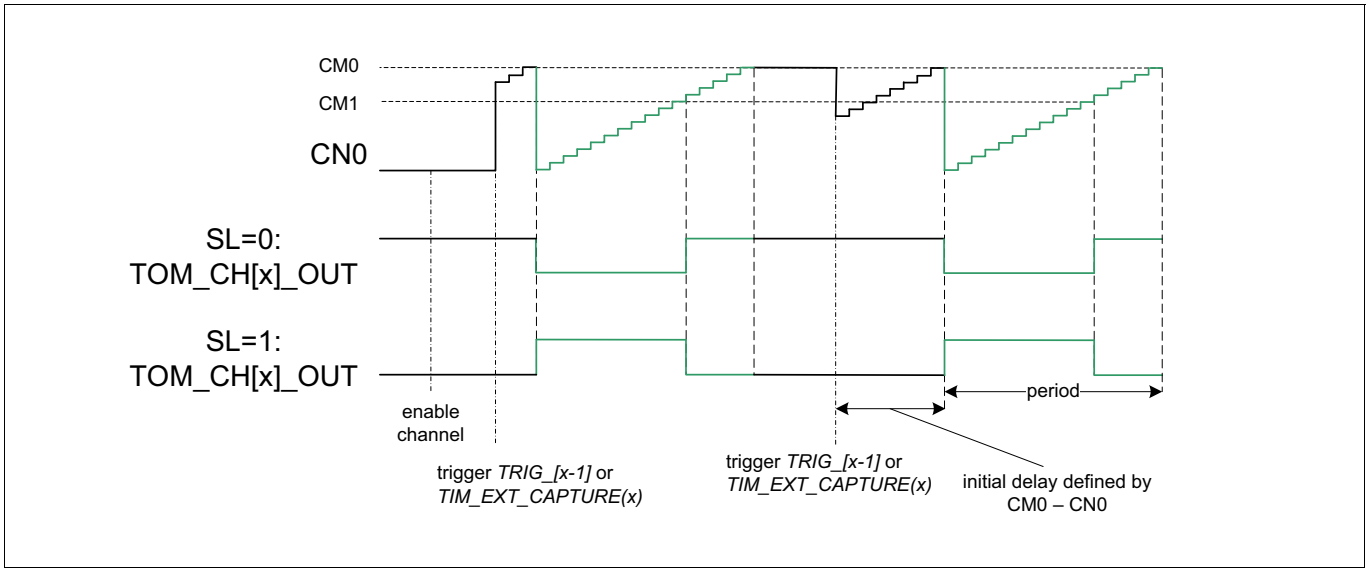


Figure 58 PWM Output with respect to configuration bit SL in one-shot mode: trigger by TRIG_[x-1] or TIM_EXT_CAPTURE(x)

28.14.3.5 One-shot Counting Up-Down Mode

The TOM channel can operate in one-shot counting up-down mode when the bit OSM = 1 and the UDMODE != 0b00. One-shot mode means that a single pulse with the pulse level defined in bit SL is generated on the output line.

First the channel has to be enabled by setting the corresponding **ENDIS_STAT** value.

In One-shot mode the counter **CN0** will not be incremented once the channel is enabled.

A write access to the register **CN0** triggers the start of pulse generation (i.e. the increment of the counter register **CN0**).

To avoid an update of **CMx** register with content of **SRx** register at this point in time, the automatic update should be disabled by setting UPEN_CTRL[x] = 0b00 (in register **TOM[i]_CH[x]_CTRL**)

If the counter **CN0** is greater or equal than **CM1**, the output *TOM[i]_CH[x]_OUT* is set to SL value.

If the counter **CN0** is less than **CM1**, the output *TOM[i]_CH[x]_OUT* is set to !SL value.

If the counter **CN0** has reached the value 0 (by counting down), it stops.

The new value of **CN0** determines the start delay of the first edge. The delay time of the first edge is given by (**CM1-CN0**) multiplied with period defined by current value of **CLK_SRC**.

Figure 59 depicts the pulse generation in one-shot mode.

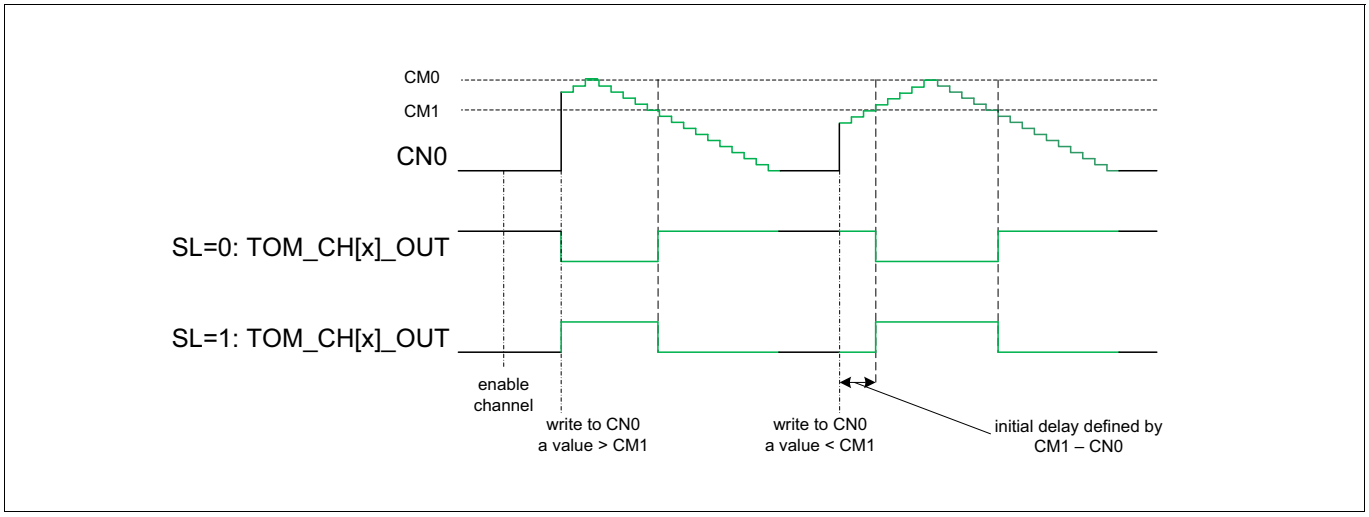


Figure 59 PWM Output with respect to configuration bit SL in one-shot counting up-down mode and UDMODE != 0b00: trigger by writing to CN0

Further output of single pulses can be started by writing to register **CN0**.

If a channel is configured to one-shot counting up-down mode and configuration bit OSM_TRIG is set to 1, the trigger signal *OSM_TRIG* (i.e. *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*) triggers start of one pulse generation.

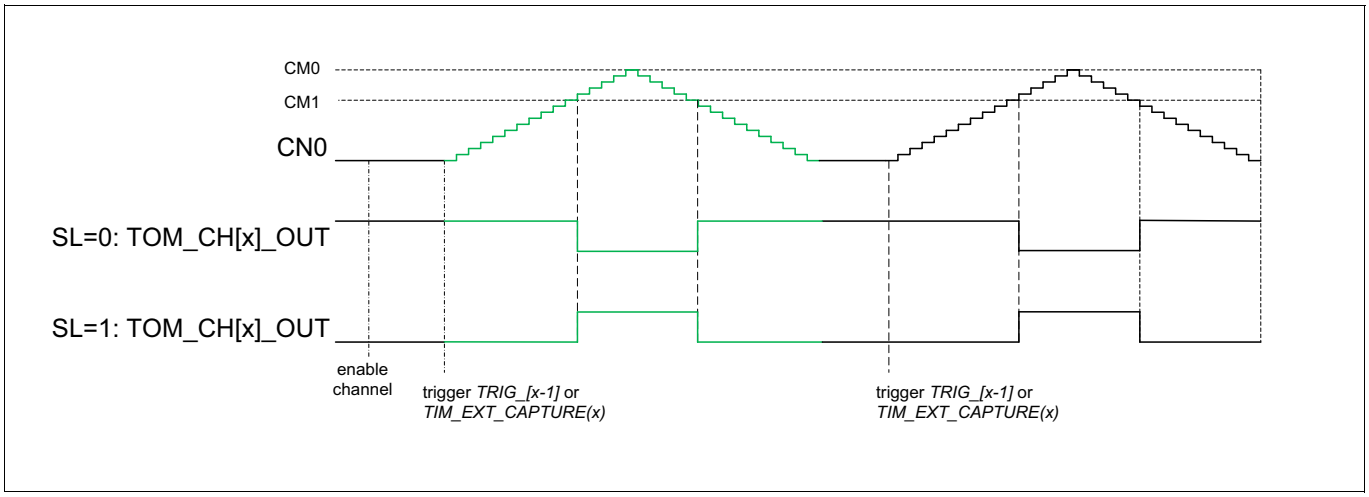


Figure 60 PWM Output with respect to configuration bit SL in one-shot counting up-down mode and UDMODE != 0b00: trigger by TRIG_[x-1] or TIM_EXT_CAPTURE(x)

28.14.3.6 Pulse Count Modulation Mode

At the output *TOM[i]_CH15_OUT* a pulse count modulated signal can be generated instead of the simple PWM output signal.

Figure **Figure 51** outlines the circuit for Pulse Count Modulation.

The PCM mode is enabled by setting bit **BITREV** to 1.

With the configuration bit **BITREV**=1 a bit-reversing of the counter output **CN0** is configured. In this case the bits LSB and MSB are swapped, the bits LSB+1 and MSB-1 are swapped, the bits LSB+2 and MSB-2 are swapped and so on.

The effect of bit-reversing of the **CN0** register value is shown in the following **Figure 61**.

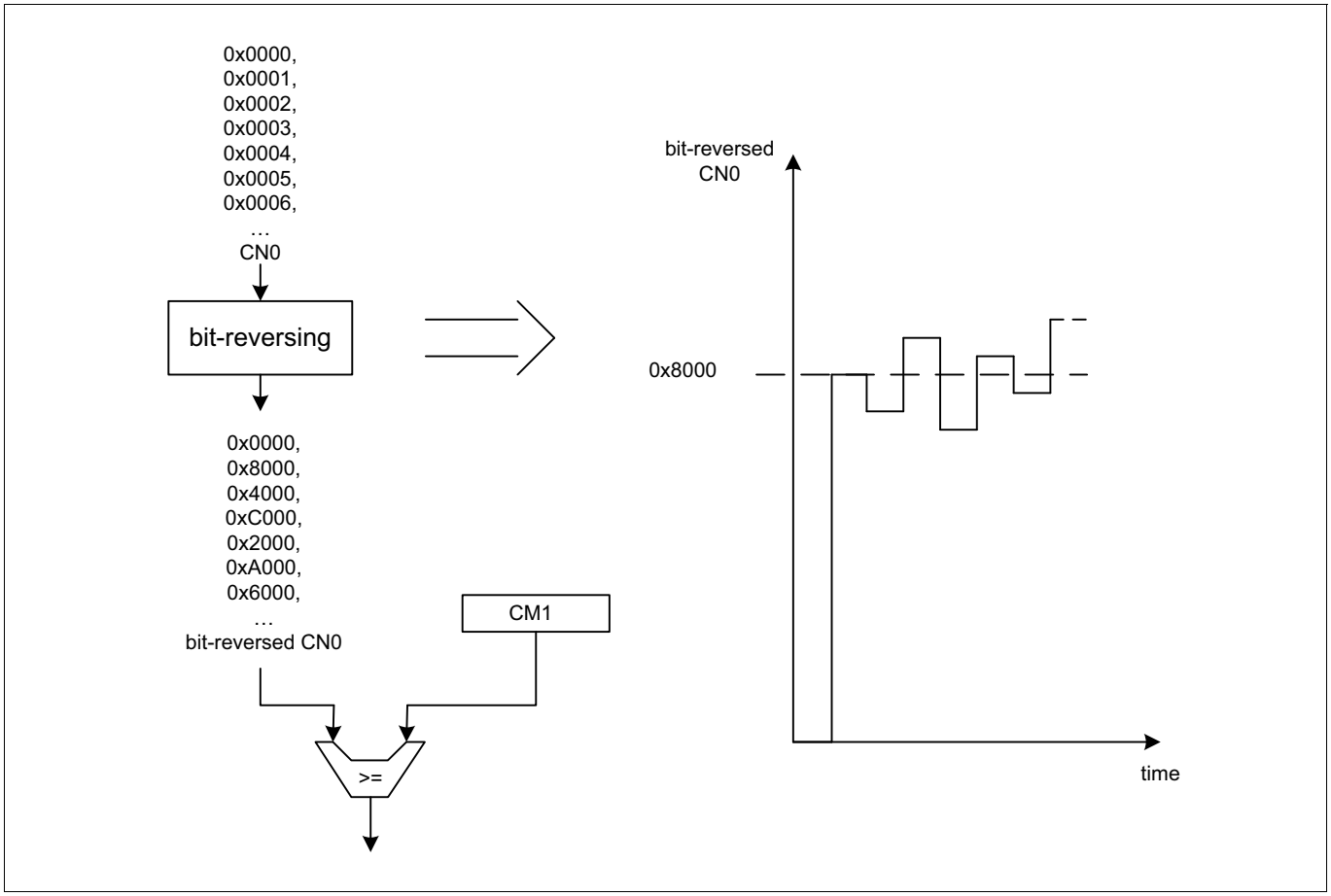


Figure 61 Bit reversing of counter CN0 output

In the PCM mode the counter register **CN0** is incremented by every clock tick depending on configured CMU clock (*CMU_FXCLK*).

The output of counter register **CN0** is first bit-reversed and then compared with the configured register value **CM1**.

If the bit-reversed value of register **CN0** is greater than **CM1**, the SR flip-flop of sub-module SOU is set (depending on configuration register **SL**) otherwise the SR flip-flop is reset. This generates at the output *TOM[i]_CH15_OUT* a pulse count modulated signal.

In PCM mode the **CM0** register - in which the period is defined - normally has to be set to its maximum value 0xFFFF.

To reduce time period of updating duty cycle value in **CM1** register, it is additionally possible to setup period value in **CM0** register to smaller values than maximum value as described before.

Possible values for **CM0** register are each even numbered values to the power of 2 e.g. 0x8000, 0x4000, 0x2000 ….

In this case the duty cycle has to be configured in the following manner.

Depending on how much the period in CM0 register is decreased - means shifted right starting from 0x10000 - the duty cycle in CM1 register has to be shifted left (= rotated: shift MSB back into LSB) with same value, e.g.

period CM0 = 0x0100 -> shifted 8 bits right from 0x10000

--> so duty cycle has to be shifted left 8 bit:

e.g. 50% duty cycle = 0x00080 -> shift 8 bits left -> CM1 = 0x8000

More examples:

period CM0	-->	duty cycle	-->	no shift	-->	CM1
0xFFFF	-->	0x8000	-->	no shift	-->	0x8000
0x8000	-->	0x4000	-->	shift 1 bit left	-->	0x8000
0x4000	-->	0x1000	-->	shift 2 bits left	-->	0x4000
0x2000	-->	0x0FFF	-->	shift 3 bits left	-->	0x7FF8
0x1000	-->	0x0333	-->	shift 4 bits left	-->	0x3330
0x0800	-->	0x0055	-->	shift 5 bits left	-->	0x0AA0
…						
0x0020	-->	0x0008	-->	shift 19 bits left	-->	0x4000
0x0010	-->	0x0005	-->	shift 20 bits left	-->	0x5000
…						

Note: In this mode the interrupt CCU1TC (see register **TOM[i]_CH[x]_IRQ_NOTIFY**) is set every time if bit reverse value of **CN0** is greater or equal than **CM1** which may be multiple times during one period. Therefore, from application point of view it is not useful to enable this interrupt.

28.14.3.7 Trigger Generation

For applications with constant PWM period defined by CM0, it is not necessary to update regularly the **CM0** register with **SR0** register. For these applications the **SR0** register can be used to define an additional output signal and interrupt trigger event.

If bit SR0_TRIG in register **TOM[i]_CH[x]_CTRL** is set, the register **SR0** is no longer used as a shadow register for register **CM0**. Instead, **SR0** is compared against **CN0** and if both are equal, a pulse of signal level '1' is generated at the output *TOM[i]_CH[x]_OUT_T*.The bit SR0_TRIG should only be set if bit RST_CCU0 of this channel is 0.

If bit SR0_TRIG is set the interrupt notify flag CCU1TC is no longer set on a compare match of **CM1** and **CN0**. Instead, the CCU1TC interrupt notify flag is set in case of a compare equal match of **SR0** and **CN0**.

With configuration bit TRIG_PULSE one can select if the output *TOM[i]_CH[x]_OUT_T* is high as long as CN0=SR0 (TRIG_PULSE=0) or if there will be only one pulse of length one SYS_CLK period when CN0 becomes SR0 (TRIG_PULSE=1).

The TOM output signal routing to DTM or GTM top level is described in subchapter “DTM connections on GTM-IP top level”

28.14.4 TOM BLDC Support

The TOM sub-module offers in combination with the SPE sub-module a BLDC support. To drive a BLDC engine TOM channels 0 to 7 can be used.

The BLDC support can be configured by setting the **SPEM** bit inside the **TOM[i]_CH[z]_CTRL** register. When this bit is set the TOM channel output is controlled through the SPE_OUT(z) signal coming from the SPE sub-module (see **Figure 49**). Please refer to chapter “Sensor Pattern Evaluation” for a detailed description of the SPE sub-module.

The TOM[i]_CH2,6,7,8 or 9 can be used together with the SPE module to trigger a delayed update of the **SPE_OUT_CTRL** register (i.e. commutation delay) after new input pattern detected by SPE (signaled by *SPE[i]_NIPD*). This feature is configured on TOM[i]_CH2,6,7,8 or 9 by setting SPE_TRIG=1 and OSM=1. With this configuration the TOM channel i generates one single PWM pulse on trigger by signal *SPE_NIPD*.

For details please refer to chapter of SPE sub-module description.

28.14.5 TOM Gated Counter Mode

Each TOM - SPE module combination provides also the feature of a gated counter mode. This is reached by using the *FSOI* input of a TIM module to gate the clock of a CCU0 sub-module.

To configure this mode, registers of module SPE should be set as following:

- the SPE should be enabled (bit **SPE_EN** = 1),
- all three TIM inputs should be disabled **SIE0** = **SIE1** = **SIE2** = 0),
- **SPE[i]_OUT_CTRL** should be set to 00005555h (set **SPE_OUT()** to '0'),
- mode FSOM should be enabled (**FSOM**=1),
- set in bit field **FSOL** bit c if channel c of module TOM is chosen for gated counter mode
- Additionally in module TOM
  - mode should be disabled (**SPEM**=0) and
  - the gated counter mode should be enabled (**GCM**=1)

As a result of this configuration, the counter **CN0** in sub-module CCU0 of TOM channel c counts as long as input *FSOI* is '0'.

28.14.6 TOM Interrupt signals

Table 48 TOM Interrupt signals

Signal	Description
TOM_CCU0TCx_IRQ	CCU0 Trigger condition interrupt for channel x
TOM_CCU1TCx_IRQ	CCU1 Trigger condition interrupt for channel x

28.14.7 TOM Configuration Register Overview

Table 49 TOM Configuration Register Overview

Register name	Description	see Page
TOM[i]_TGC[y]_GLB_CTRL	TOMi TGC y global control register	221
TOM[i]_TGC[y]_ENDIS_CTRL	TOMi TGC y enable/disable control register	222
TOM[i]_TGC[y]_ENDIS_STAT	TOMi TGC y enable/disable status register	223
TOM[i]_TGC[y]_ACT_TB	TOMi TGC y action time base register	224
TOM[i]_TGC[y]_OUTEN_CTRL	TOMi TGC y output enable control register	225
TOM[i]_TGC[y]_OUTEN_STAT	TOMi TGC y output enable status register	226
TOM[i]_TGC[y]_FUPD_CTRL	TOMi TGC y force update control register	226
TOM[i]_TGC[y]_INT_TRIG	TOMi TGC y internal trigger control register	227
TOM[i]_CH[x]_CTRL	TOMi channel x control register	228
TOM[i]_CH[x]_CN0	TOMi channel x CCU0 counter register	232
TOM[i]_CH[x]_CM0	TOMi channel x CCU0 compare register	233
TOM[i]_CH[x]_SR0	TOMi channel x CCU0 compare shadow register	233
TOM[i]_CH[x]_CM1	TOMi channel x CCU1 compare register	234
TOM[i]_CH[x]_SR1	TOMi channel x CCU1 compare shadow register	234
TOM[i]_CH[x]_STAT	TOMi channel x status register	235
TOM[i]_CH[x]_IRQ_NOTIFY	TOMi channel x interrupt notification register	235
TOM[i]_CH[x]_IRQ_EN	TOMi channel x interrupt enable register	236
TOM[i]_CH[x]_IRQ_FORCINT	TOMi channel x force interrupt register	237
TOM[i]_CH[x]_IRQ_MODE	TOMi channel x interrupt mode register	237

28.14.8 TOM Configuration Register Description

28.14.8.1 Register TOM[i]_TGC[y]_GLB_CTRL

TOMi TGC0 Global Control Register

TOMi_TGC0_GLB_CTRL (i=0-5)
TOMi TGC0 Global Control Register **(008030_H+i*800_H)** **Application Reset Value: 0000 0000_H**
TOMi_TGC1_GLB_CTRL (i=0-5)
TOMi TGC1 Global Control Register **(008230_H+i*800_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPEN_CTRL7		UPEN_CTRL6		UPEN_CTRL5		UPEN_CTRL4		UPEN_CTRL3		UPEN_CTRL2		UPEN_CTRL1		UPEN_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST_CH7	RST_CH6	RST_CH5	RST_CH4	RST_CH3	RST_CH2	RST_CH1	RST_CH0	0							HOST_TRIG
w	w	w	w	w	w	w	w	r							w

Field	Bits	Type	Description
HOST_TRIG	0	w	**Trigger request signal (see TGC0, TGC1) to update the register ENDIS_STAT and OUTEN_STAT** This flag is reset automatically after triggering the update. 0_B No trigger request 1_B Set trigger request
RST_CHx (x=0-7)	x+8	w	**Software reset of channel x** This bit is cleared automatically after write by CPU. The channel register are set to their reset values and channel x operation is stopped immediately. The SR flip-flop SOUR is set to '1'. 0_B No action 1_B Reset channel x
UPEN_CTRLx (x=0-7)	2*x+17:2*x +16	rw	**TOM channel x enable update of register CM0, CM1 and CLK_SRC from SR0, SR1 and CLK_SRC_SR** Write / Read : 00_B Don’t care, bits 1:0 will not be changed / update disabled 01_B Disable update / -- 10_B Enable update / -- 11_B Don’t care, bits 1:0 will not be changed / update enabled
0	7:1	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.2 Register TOM[i]_TGC[y]_ENDIS_CTRL

TOMi TGC0 Enable/Disable Control Register

TOMi_TGC0_ENDIS_CTRL (i=0-5)
TOMi TGC0 Enable/Disable Control Register(008070_H+i*800_H) **Application Reset Value: 0000 0000_H**
TOMi_TGC1_ENDIS_CTRL (i=0-5)
TOMi TGC1 Enable/Disable Control Register(008270_H+i*800_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIS_CTRL7		ENDIS_CTRL6		ENDIS_CTRL5		ENDIS_CTRL4		ENDIS_CTRL3		ENDIS_CTRL2		ENDIS_CTRL1		ENDIS_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
ENDIS_CTRLx (x=0-7)	2*x+1:2*x	rw	**TOM channel x enable/disable update value** If FREEZE = 0: If a TOM channel is disabled, the counter CN0 is stopped and the output register of SOU unit is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value. If FREEZE = 1: If a TOM channel is disabled, the counter CN0 is stopped. On an enable event, the counter CN0 starts counting from its current value. Write of following double bit values is possible: If the output is disabled (OUTEN[0]=0), the TOM channel 0 output TOM[i]_CH0_OUT is the inverted value of bit SL. 00_B Don't care, bits 1:0 of register ENDIS_STAT will not be changed on an update trigger 01_B Disable channel on an update trigger 10_B Enable channel on an update trigger 11_B Don't change bits 1:0 of this register
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.3 Register TOM[i]_TGC[y]_ENDIS_STAT

TOMi TGC0 Enable/Disable Status Register

TOMi_TGC0_ENDIS_STAT (i=0-5)
TOMi TGC0 Enable/Disable Status Register (008074_H+i*800_H) **Application Reset Value: 0000 0000_H**
TOMi_TGC1_ENDIS_STAT (i=0-5)
TOMi TGC1 Enable/Disable Status Register (008274_H+i*800_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

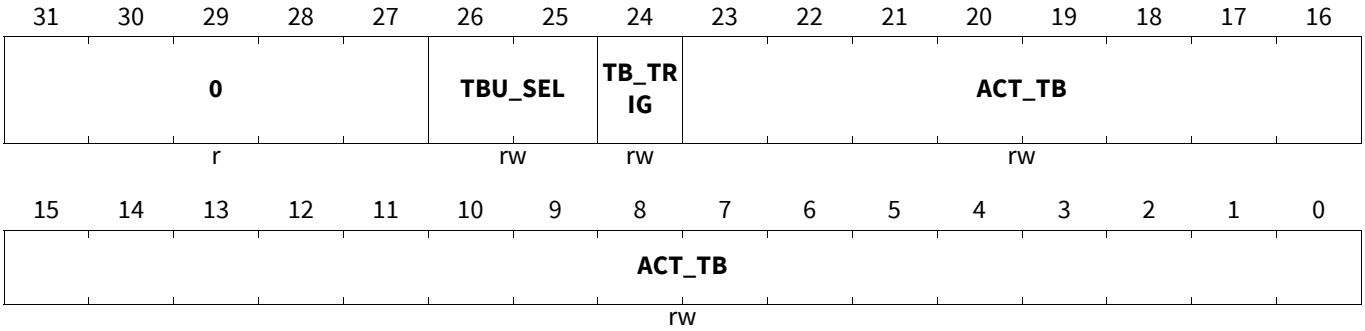
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIS_STAT7		ENDIS_STAT6		ENDIS_STAT5		ENDIS_STAT4		ENDIS_STAT3		ENDIS_STAT2		ENDIS_STAT1		ENDIS_STAT0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
ENDIS_STATx (x=0-7)	2*x+1:2*x	rw	**TOM channel x enable/disable update value** If FREEZE = 0: If a TOM channel is disabled, the counter CN0 is stopped and the output register of SOU unit is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value. If FREEZE = 1: If a TOM channel is disabled, the counter CN0 is stopped. On an enable event, the counter CN0 starts counting from its current value. Write of following double bit values is possible: 00_B Don't care, bits 1:0 will not be changed / channel disabled 01_B Disable channel / -- 10_B Enable channel / -- 11_B Don't care, bits 1:0 will not be changed / channel enabled
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.4 Register TOM[i]_TGC[y]_ACT_TB

TOMi TGC0 Action Time Base Register

TOMi_TGC0_ACT_TB (i=0-5)
TOMi TGC0 Action Time Base Register **(008034_H+i*800_H)** **Application Reset Value: 0000 0000_H**
TOMi_TGC1_ACT_TB (i=0-5)
TOMi TGC1 Action Time Base Register **(008234_H+i*800_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ACT_TB	23:0	rw	**Time base value** Specifies the signed compare value with selected signal TBU_TS[x], x=0..2 If selected TBU_TS[x] value is in the interval [**ACT_TB** - 007FFFFFh, **ACT_TB**], the event is in the past, and the trigger is generated immediately. Otherwise, the event is in the future, and the trigger is generated if selected TBU_TS[x] is equal to **ACT_TB**.
TB_TRIG	24	rw	**Set trigger request** This flag is reset automatically if the selected time base unit (*TBU_TS0* or *TBU_TS1* or *TBU_TS2,* if present) has reached the value ACT_TB and the update of the register was triggered. 0_B No trigger request 1_B Set trigger request
TBU_SEL	26:25	rw	**Selection of time base used for comparison** The bit combination 0b10 is only applicable if the TBU of the device contains three time base channels. Otherwise, this bit combination is also reserved. Please refer to GTM Architecture block diagram on page 3 to determine the number of channels for TBU of this device. 00_B TBU_TS0 selected 01_B TBU_TS1 selected 10_B TBU_TS2 selected 11_B Same as 0b00
0	31:27	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.5 Register TOM[i]_TGC[y]_OUTEN_CTRL

TOMi TGC0 Output Enable Control Register

TOMi_TGC0_OUTEN_CTRL (i=0-5)
TOMi TGC0 Output Enable Control Register(008078_H+i*800_H) **Application Reset Value: 0000 0000_H**
TOMi_TGC1_OUTEN_CTRL (i=0-5)
TOMi TGC1 Output Enable Control Register(008278_H+i*800_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTEN_CTRL 7		OUTEN_CTRL 6		OUTEN_CTRL 5		OUTEN_CTRL 4		OUTEN_CTRL 3		OUTEN_CTRL 2		OUTEN_CTRL 1		OUTEN_CTRL 0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
OUTEN_CTRL x (x=0-7)	2*x+1:2*x	rw	**Output TOM[i]_CHx_OUT enable/disable update value** Write of following double bit values is possible: If the channel is disabled (ENDIS[0]=0) or the output is disabled (OUTEN[0]=0), the TOM channel 0 output *TOM[i]_CH0_OUT* is the inverted value of bit SL. 00_B Don't care, bits 1:0 of register OUTEN_STAT will not be changed on an update trigger 01_B Disable channel output on an update trigger 10_B Enable channel output on an update trigger 11_B Don't change bits 1:0 of this register
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.6 Register TOM[i]_TGC[y]_OUTEN_STAT

TOMi TGC0 Output Enable Status Register

TOMi_TGC0_OUTEN_STAT (i=0-5)
TOMi TGC0 Output Enable Status Register (00807C$_H$+i*800$_H$) **Application Reset Value: 0000 0000$_H$**
TOMi_TGC1_OUTEN_STAT (i=0-5)
TOMi TGC1 Output Enable Status Register (00827C$_H$+i*800$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTEN_STAT7		OUTEN_STAT6		OUTEN_STAT5		OUTEN_STAT4		OUTEN_STAT3		OUTEN_STAT2		OUTEN_STAT1		OUTEN_STAT0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
OUTEN_STATx (x=0-7)	2*x+1:2*x	rw	**Control/status of output TOM[i]_CHx_OUT** Write of following double bit values is possible: 00_B Don't care, bits 1:0 will not be changed / output disabled 01_B Disable output / -- 10_B Enable output / -- 11_B Don't care, bits 1:0 will not be changed / output enabled
0	31:16	r	**Reserved** Read as zero, shall be written as zero

28.14.8.7 Register TOM[i]_TGC[y]_FUPD_CTRL

TOMi TGC0 Force Update Control Register

TOMi_TGC0_FUPD_CTRL (i=0-5)
TOMi TGC0 Force Update Control Register (008038$_H$+i*800$_H$) **Application Reset Value: 0000 0000$_H$**
TOMi_TGC1_FUPD_CTRL (i=0-5)
TOMi TGC1 Force Update Control Register (008238$_H$+i*800$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSTCN0_CH7		RSTCN0_CH6		RSTCN0_CH5		RSTCN0_CH4		RSTCN0_CH3		RSTCN0_CH2		RSTCN0_CH1		RSTCN0_CH0	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FUPD_CTRL7		FUPD_CTRL6		FUPD_CTRL5		FUPD_CTRL4		FUPD_CTRL3		FUPD_CTRL2		FUPD_CTRL1		FUPD_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
FUPD_CTRLx (x=0-7)	2*x+1:2*x	rw	**Force update of TOM channel x operation registers** If enabled, force update of register CM0, CM1 and CLK_SRC triggered by HOST_TRIG, ACT_TB compare match, or internal trigger. Write / Read: The force update request is stored and executed synchronized to the selected FXCLK. 00_B Don't care, bits 1:0 will not be changed / force update disabled 01_B Disable force update / -- 10_B Enable force update / -- 11_B Don't care, bits 1:0 will not be changed / force update enabled
RSTCN0_CHx (x=0-7)	2*x+17:2*x +16	rw	**Reset CN0 of channel x on force update event** If enabled, reset CN0 triggered by HOST_TRIG, ACT_TB compare match, or internal trigger. Write / Read: 00_B Don't care, bits 1:0 will not be changed / CN0 is not reset on forced update 01_B Do not reset CN0 on forced update / -- 10_B Reset CN0 on forced update / -- 11_B Don't care, bits 1:0 will not be changed / CN0 is reset on forced update

28.14.8.8 Register TOM[i]_TGC[y]_INT_TRIG

TOMi TGC0 Internal Trigger Control Register

TOMi_TGC0_INT_TRIG (i=0-5)
TOMi TGC0 Internal Trigger Control Register($00803C_H$+i*800_H) **Application Reset Value: 0000 0000_H**
TOMi_TGC1_INT_TRIG (i=0-5)
TOMi TGC1 Internal Trigger Control Register($00823C_H$+i*800_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_TRIG7		INT_TRIG6		INT_TRIG5		INT_TRIG4		INT_TRIG3		INT_TRIG2		INT_TRIG1		INT_TRIG0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
INT_TRIGx (x=0-7)	2*x+1:2*x	rw	**Select input signal TRIG_x as a trigger source** Write / Read: 00_B Don't care, bits 1:0 will not be changed / internal trigger from channel 0 (TRIG_0) not used 01_B Do not use internal trigger from channel 0 (TRIG_0) / -- 10_B Use internal trigger from channel 0 (TRIG_0) / -- 11_B Don't care, bits 1:0 will not be changed / internal trigger from channel 0 (TRIG_0) used
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.9 Register TOM[i]_CH[x]_CTRL

TOMi Channel x Control Register

TOMi_CHx_CTRL (i=0-5;x=0-15)

TOMi Channel x Control Register **(008000_H+i*800_H+x*40_H)** **Application Reset Value: 0000 0800_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREEZE	**0**	**GCM**	**SPEM**	**BITREV**	**OSM**	**SPE_TRIG**	**TRIGOUT**	**EXTTRIGOUT**	**EXT_TRIG**	**OSM_TRIG**	**RST_CCU0**	**UDMODE**		**TRIG_PULSE**	**0**
rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECLK_SRC	**CLK_SRC_SR**			**SL**	**0**			**SR0_TRIG**	**0**						
rw	rw			rw	r			rw	r						

Field	Bits	Type	Description
SR0_TRIG	7	rw	**SR0 is used to generate a trigger on output TOM[i]_CH[x]_OUT_T if equal to CN0** *Note: This bit should only be set if RST_CCU0 of this channel is 0.* 0_B SR0 is used as a shadow register for register CM0. 1_B SR0 is not used as a shadow register for register CM0. SR0 is compared with CN0 and if both are equal, a trigger pulse is generated at output TOM[i]_CH[x]_OUT_T.
SL	11	rw	**Signal level for duty cycle** If the output is disabled, the output TOM_OUT[x] is set to the inverse value of SL. Reset value depends on the hardware configuration chosen by silicon vendor. 0_B Low signal level 1_B High signal level

Field	Bits	Type	Description
CLK_SRC_SR	14:12	rw	**Clock source select for channel** The register CLK_SRC is updated with the value of CLK_SRC_SR together with the update of register CM0 and CM1. The input of the FX clock divider depends on the value of FXCLK_SEL (see CMU). *Note: This register is a shadow register for the register CLK_SRC. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done either by an end of a period or a forced update.* *Note: If clock of channel is stopped (i.e. ECLK_SRC=0 and CLK_SRC=101_B/110_B/111_B), the channel can only be restarted by resetting CLK_SRC_SR to a value of 000_B to 100_B and forcing an update via the force update mechanism.* If ECLK_SRC=0 / ECLK_SRC=1: 000_B CMU_FXCLK (0) selected / CMU_FXCLK (0) selected 001_B CMU_FXCLK (1) selected / CMU_FXCLK (1) selected 010_B CMU_FXCLK (2) selected / CMU_FXCLK (2) selected 011_B CMU_FXCLK (3) selected / CMU_FXCLK (3) selected 100_B CMU_FXCLK (4) selected / CMU_FXCLK (4) selected 101_B Clock of channel stopped / TRIG[x-1] selected 110_B Clock of channel stopped / TIM_EXT_CAPTURE[x] selected 111_B Clock of channel stopped / reserved
ECLK_SRC	15	rw	**Extend CLK_SRC** 0_B CLK_SRC_SR set 1 selected (see bit CLK_SRC_SR) 1_B CLK_SRC_SR set 2 selected (see bit CLK_SRC_SR)
TRIG_PULSE	17	rw	**Trigger output pulse length of one SYS_CLK period** 0_B Output on TOM[i]_OUT[x]_T is 1 as long as CN0=SR0 (if SR=_TRIG=1) 1_B Output on TOM[i]_OUT[x]_T is 1 for only one SYS_CLK period if CN0=SR0 (if SR=_TRIG=1)
UDMODE	19:18	rw	**Up-down counter mode** 00_B Up-down counter mode disabled: CN0 counts always up 01_B Up-down counter mode enabled: CN0 counts up and down, CM0, CM1 are updated if CN0 reaches 0 (i.e. changes from down to up) 10_B Up-down counter mode enabled: CN0 counts up and down, CM0, CM1 are updated if CN0 reaches CM0 (i.e. changes from up to down) 11_B Up-down counter mode enabled: CN0 counts up and down, CM0, CM1 are updated if CN0 reaches 0 or CM0 (i.e. changes direction)

Field	Bits	Type	Description
RST_CCU0	20	rw	**Reset source of CCU0** *Note: On TOM channel 2, SPEM=1 has special meaning. If SPEM = 1, the signal SPE_NIPD triggers the reset of CN0 independent of RST_CN0.* *Note: This bit should only be set if bit OSM=0 (i.e. in continuous mode).* 0_B Reset counter register CN0 to 0 on matching comparison CM0 1_B Reset counter register CN0 to 0 on trigger TRIG_[x-1] or TIM_EXT_CAPTURE(x).
OSM_TRIG	21	rw	**Enable trigger of one-shot pulse by trigger signal OSM_TRIG** *Note: This bit should only be set if bit OSM=1 and bit RST_CCU0=0.* 0_B Signal OSM_TRIG cannot trigger start of single pulse generation 1_B Signal OSM_TRIG can trigger start of single pulse generation (only if bit OSM = 1)
EXT_TRIG	22	rw	**Select TIM_EXT_CAPTURE(x) as trigger signal** 0_B Signal TRIG_[x-1] is selected as trigger to reset CN0 or to start single pulse generation 1_B Signal TIM_EXT_CAPTURE(x) is selected
EXTTRIGOUT	23	rw	**TIM_EXT_CAPTURE(x) as potential output signal TRIG_[x]** 0_B Signal TRIG_[x-1] is selected as output on TRIG_[x] (if TRIGOUT=0) 1_B Signal TIM_EXT_CAPTURE(x) is selected as output on TRIG_[x] (if TRIGOUT=0)
TRIGOUT	24	rw	**Trigger output selection (output signal TRIG_[x]) of module TOM_CH[x]** 0_B TRIG_[x] is TRIG_[x-1] or TIM _EXT_CAPTURE(x) 1_B TRIG_[x] is TRIG_CCU0

Field	Bits	Type	Description
SPE_TRIG	25	rw	**SPE trigger to reset CN0** For TOM channel 2, 6 and 7, this bit defines, in combination with bit SPEM, the source of output pin TOM[i]_CH[x]_OUT, and if CN0 can be reset by TOM input signal SPE[i]_NIPD. *Note: For TOM channel 8 and 9, this bit defines only if CN0 reset is defined by input signal SPE[i]_NIPD or by configuration of RST_CCU0. The output TOM[i]_CH[x]_OUT is not affected. The configuration bit SPEM is not available for these channels, and thus assumed to be 0.* *Note: If a configuration of SPEM | SPE_TRIG = 0 | 1 or 1 | 0 is chosen (i.e. CN0 is reset by signal SPE[i]_NIPD), the one-shot mode in corresponding TOM channel should also be enabled by setting bit OSM=1 to generate one PWM pulse in case of trigger SPE[i]_NIPD.* *Note: In SPE module, one of the trigger signals TOM[i]_CH2_TRIG_CCU1, TOM[i]_CH6_TRIG_CCU1, TOM[i]_CH7_TRIG_CCU1, TOM[i]_CH8_TRIG_CCU1, or TOM[i]_CH9_TRIG_CCU1 can be used to trigger the update of register SPE[i]_OUT_CTRL.* If SPEM=0 / SPEM=1: 0_B TOM[i]_CH[x]_OUT defined by TOM[i] channel x SOUR register, CN0 reset is defined by configuration of bit RST_CCU0 / TOM[i]_CH[x]_OUT is defined by SPE[i]_OUT[x], CN0 is reset by signal SPE[i]_NIPD 1_B TOM[i]_CH[x]_OUT defined by TOM[i] channel x SOUR register, CN0 is reset by signal SPE[i]_NIPD / TOM[i]_CH[x]_OUT is defined by SPE[i]_OUT[x], CN0 reset is defined by configuration of bit RST_CCU0
OSM	26	rw	**One-shot mode** In this mode, the counter CN0 counts for only one period. The length of period is defined by CM0. A write access to the register CN0 triggers the start of counting. 0_B One-shot mode disabled 1_B One-shot mode enabled
BITREV	27	rw	**Bit-reversing of output of counter register CN0** *Note: This bit enables the PCM mode of channel 15.*

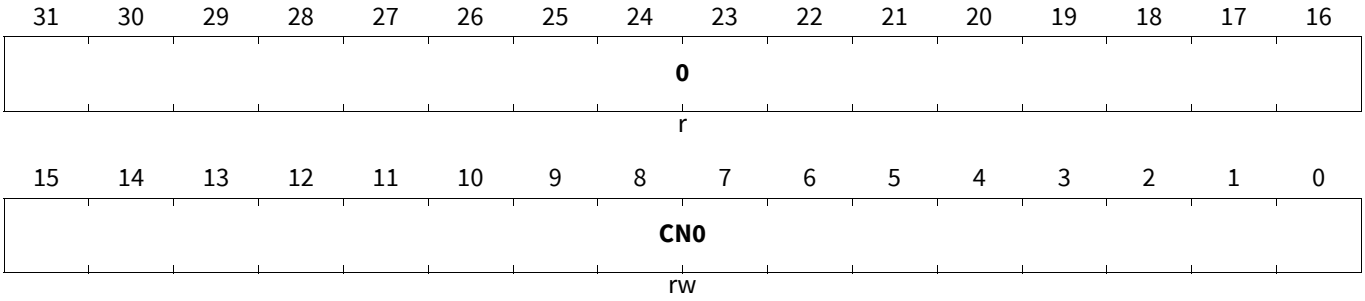
Field	Bits	Type	Description
SPEM	28	rw	**SPE output mode enable for channel** *Note: The SPE output mode is only implemented for TOM instances connected to an SPE module, and only for TOM channels 0 to 7.* *Note: For TOM channel 2, 6 and 7, this bit defines, in combination with bit SPE_TRIG, the source of output pin TOM[i]_CH[x]_OUT, and if CN0 can be reset by TOM input signal SPE[i]_NIPD.* 0_B SPE output mode disabled: TOM[i]_CH[x]_OUT defined by TOM[i] channel x SOUR register 1_B SPE output mode enabled: TOM[i]_CH[x]_OUT is defined by SPE[i]_OUT[x]
GCM	29	rw	**Gated Counter Mode enable** The Gated Counter mode is only available for TOM instances connected to an SPE module, and only for channels 0 to 7. 0_B Gated Counter mode disabled 1_B Gated Counter mode enabled
FREEZE	31	rw	**FREEZE** 0_B A channel disable/enable may change internal register and output register 1_B A channel enable/disable does not change an internal or output register, but stops counter CN0
0	6:0, 10:8, 16, 30	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.10Register TOM[i]_CH[x]_CN0

TOMi Channel x CCU0 Counter Register

TOMi_CHx_CN0 (i=0-5;x=0-15)

TOMi Channel x CCU0 Counter Register (008014_H+i*800_H+x*40_H) **Application Reset Value: $0000\ 0000_H$**



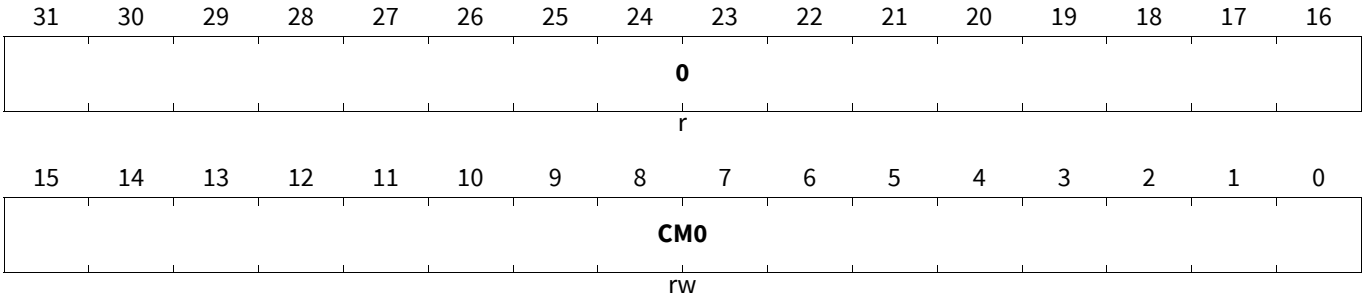
Field	Bits	Type	Description
CN0	15:0	rw	**TOM CCU0 counter register** This counter is stopped if the TOM channel is disabled and not reset on an enable event of TOM channel.

Field	Bits	Type	Description
0	31:16	r	**Reserved** Read as zero, shall be written as zero

28.14.8.11Register TOM[i]_CH[x]_CM0

TOMi Channel x CCU0 Compare Register

TOMi_CHx_CM0 (i=0-5;x=0-15)
TOMi Channel x CCU0 Compare Register($00800C_H$+i*800_H+x*40_H) **Application Reset Value: 0000 0000_H**

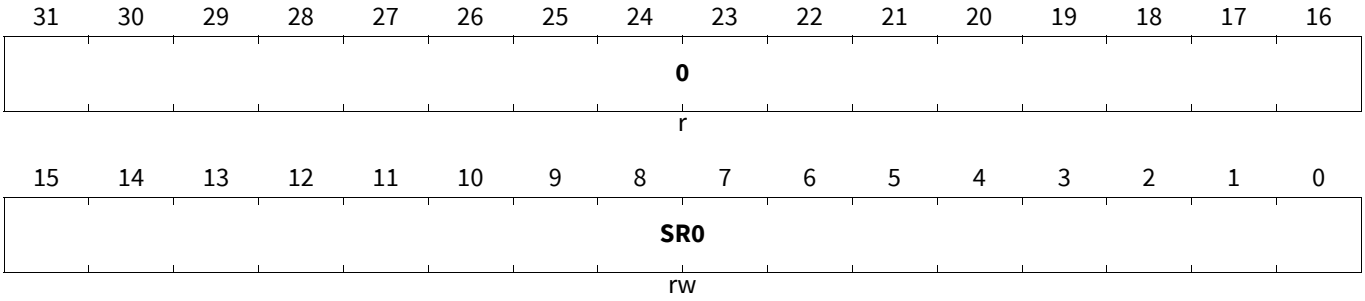


Field	Bits	Type	Description
CM0	15:0	rw	**TOM CCU0 compare register** Setting CM0 < CM1 configures a duty cycle of 100%.
0	31:16	r	**Reserved** Read as zero, shall be written as zero

28.14.8.12Register TOM[i]_CH[x]_SR0

TOMi Channel x CCU0 Compare Shadow Register

TOMi_CHx_SR0 (i=0-5;x=0-15)
TOMi Channel x CCU0 Compare Shadow Register(008004_H+i*800_H+x*40_H) **Application Reset Value: 0000 0000_H**



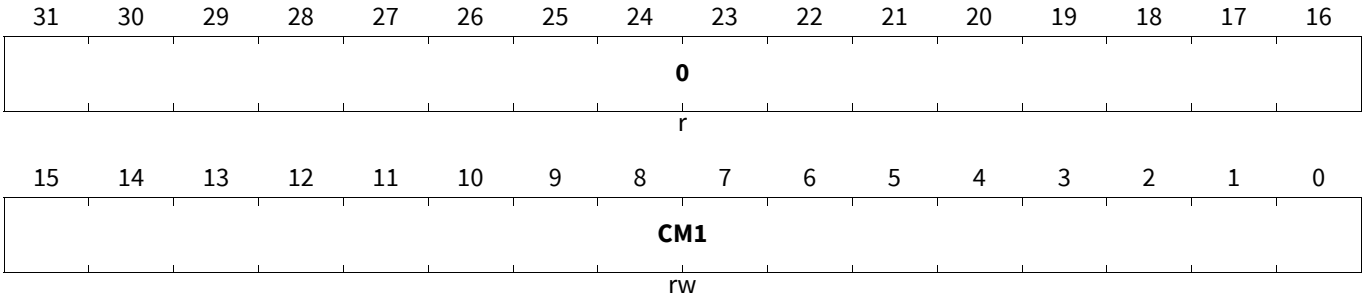
Field	Bits	Type	Description
SR0	15:0	rw	**TOM channel x shadow register SR0 for update of compare register CM0**

Field	Bits	Type	Description
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.13Register TOM[i]_CH[x]_CM1

TOMi Channel x CCU1 Compare Register

TOMi_CHx_CM1 (i=0-5;x=0-15)
TOMi Channel x CCU1 Compare Register(008010_H+i*800_H+x*40_H) **Application Reset Value: 0000 0000_H**

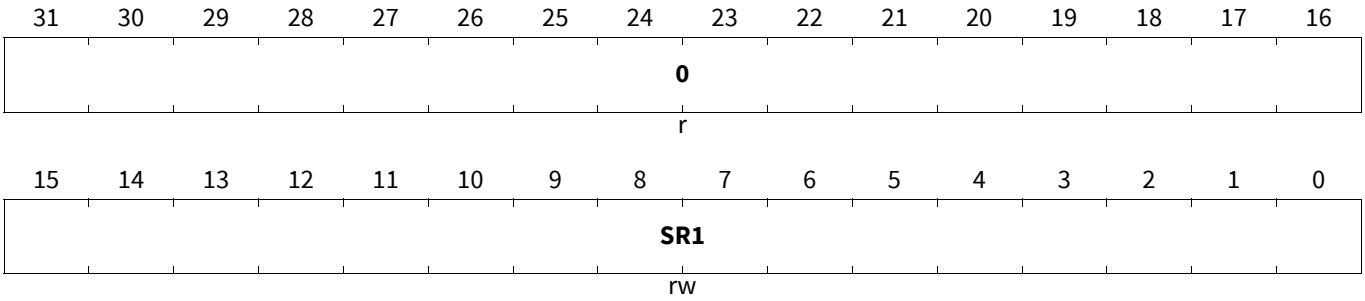


Field	Bits	Type	Description
CM1	15:0	rw	**TOM CCU1 compare register** Setting CM1 = 0 configures a duty cycle of 0%, independent of the configured value of CM0.
0	31:16	r	**Reserved** Read as zero, shall be written as zero

28.14.8.14Register TOM[i]_CH[x]_SR1

TOMi Channel x CCU1 Compare Shadow Register

TOMi_CHx_SR1 (i=0-5;x=0-15)
TOMi Channel x CCU1 Compare Shadow Register(008008_H+i*800_H+x*40_H) **Application Reset Value: 0000 0000_H**



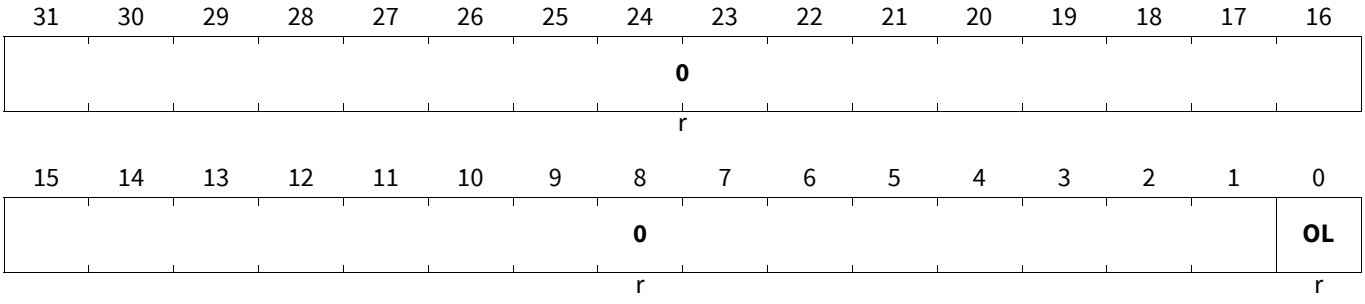
Field	Bits	Type	Description
SR1	15:0	rw	**TOM channel x shadow register SR1 for update of compare register CM1**

Field	Bits	Type	Description
0	31:16	r	**Reserved** Read as zero, shall be written as zero

28.14.8.15Register TOM[i]_CH[x]_STAT

TOMi Channel x Status Register

TOMi_CHx_STAT (i=0-5;x=0-15)
TOMi Channel x Status Register **(008018_H+i*800_H+x*40_H)** **Application Reset Value: 0000 0000_H**

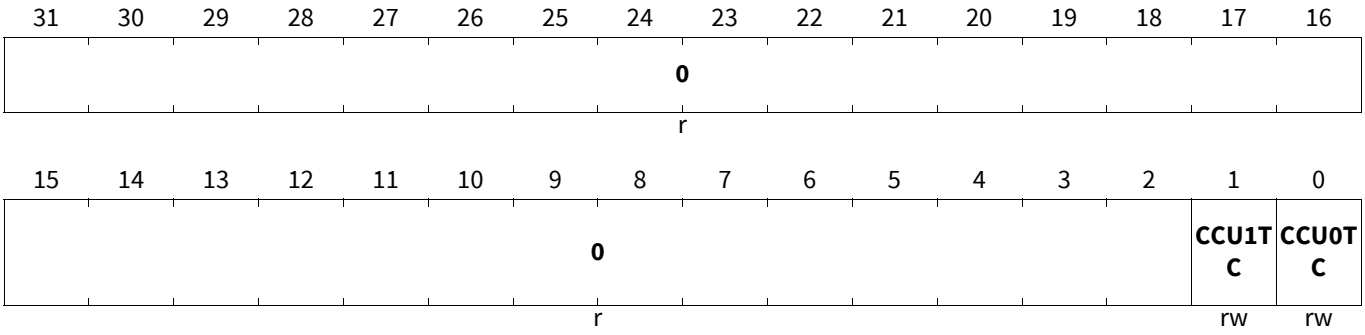


Field	Bits	Type	Description
OL	0	r	**Output level of output TOM_OUT(x)** Reset value is the inverted value of SL bit, which depends on the hardware configuration chosen by silicon vendor.
0	31:1	r	**Reserved** Read as zero, shall be written as zero

28.14.8.16Register TOM[i]_CH[x]_IRQ_NOTIFY

TOMi Channel x Interrupt Notification Register

TOMi_CHx_IRQ_NOTIFY (i=0-5;x=0-15)
TOMi Channel x Interrupt Notification Register($00801C_H$+i*800_H+x*40_H) **Application Reset Value: 0000 0000_H**



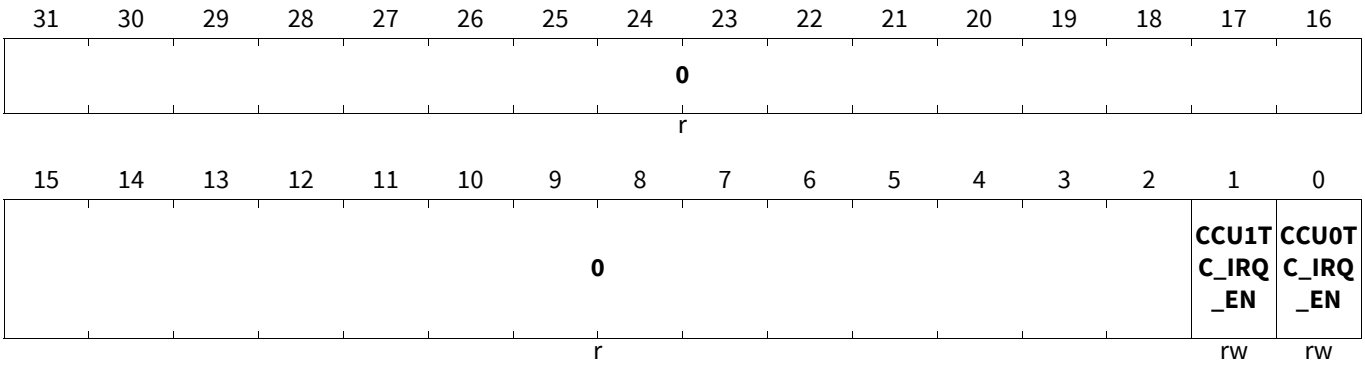
Field	Bits	Type	Description
CCU0TC	0	rw	**CCU0 Trigger condition interrupt for channel x** The notification of the interrupt is only triggered one time after reaching the condition CN0 >= CM0. To enable re-trigger of the notification, first the condition CN0 < CM1 has to be reached. *Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.* 0_B No interrupt occurred 1_B The condition CN0 >= CM0 was detected
CCU1TC	1	rw	**CCU1 Trigger condition interrupt for channel x** The notification of the interrupt is only triggered one time after reaching the condition CN0 >= CM1. To enable re-trigger of the notification, first the condition CN0 < CM1 has to be reached. *Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.* 0_B No interrupt occurred 1_B The condition CN0 >= CM1 was detected (if SR0_TRIG=0) / the condition SR0=CN0 was detected (if SR0_TRIG=1)
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.17Register TOM[i]_CH[x]_IRQ_EN

TOMi Channel x Interrupt Enable Register

TOMi_CHx_IRQ_EN (i=0-5;x=0-15)

TOMi Channel x Interrupt Enable Register(008020_H+i*800_H+x*40_H) **Application Reset Value: 0000 0000_H**



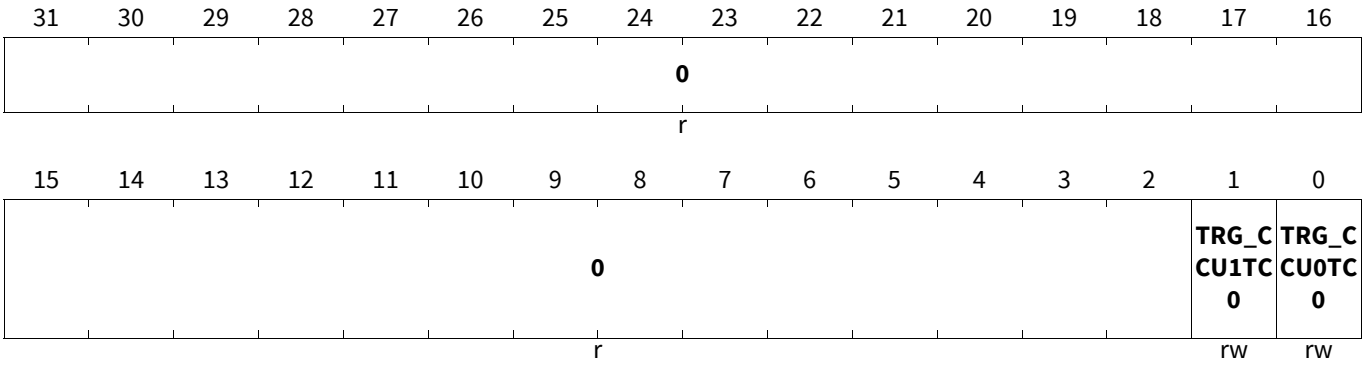
Field	Bits	Type	Description
CCU0TC_IRQ_EN	0	rw	**TOM_CCU0TC_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
CCU1TC_IRQ_EN	1	rw	**TOM_CCU1TC_IRQ interrupt enable** Coding see bit 0.
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.18Register TOM[i]_CH[x]_IRQ_FORCINT

TOMi Channel x Force Interrupt Register

TOMi_CHx_IRQ_FORCINT (i=0-5;x=0-15)

TOMi Channel x Force Interrupt Register(008024_H+i*800_H+x*40_H) **Application Reset Value: 0000 0000_H**



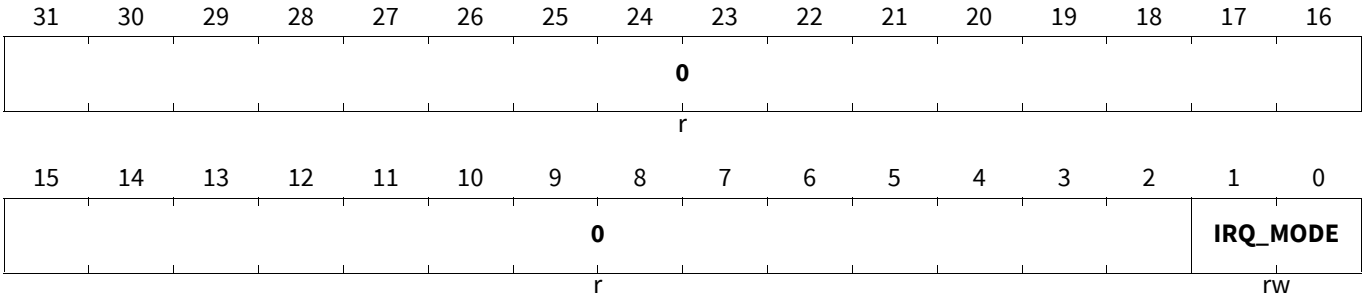
Field	Bits	Type	Description
TRG_CCU0TC0	0	rw	**Trigger TOM_CCU0TC0_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL 0_B No interrupt triggering 1_B Assert CCU0TC0_IRQ interrupt for one clock cycle
TRG_CCU1TC0	1	rw	**Trigger TOM_CCU1TC0_IRQ interrupt by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B No interrupt triggering 1_B Assert CCU1TC0_IRQ interrupt for one clock cycle
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.14.8.19Register TOM[i]_CH[x]_IRQ_MODE

TOMi Channel x Interrupt Mode Register

TOMi_CHx_IRQ_MODE (i=0-5;x=0-15)

TOMi Channel x Interrupt Mode Register(008028_H+i*800_H+x*40_H) **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** The interrupt modes are described in **Section 28.4.5**. 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero

28.15 ARU-connected Timer Output Module (ATOM)

28.15.1 Overview

The ARU-connected Timer Output Module (ATOM) is able to generate complex output signals without CPU interaction due to its connectivity to the ARU. Typically, output signal characteristics are provided over the ARU connection through sub-modules connected to ARU like e.g. the MCS, DPLL or PSM. Each ATOM sub-module contains eight output channels which can operate independently from each other in several configurable operation modes. A block diagram of the ATOM sub-module is depicted in **Figure 62**.

The following design variables are used inside this chapter. Please refer to device specific appendix for correct value.

cCATO: ATOM channel count; number of channels per instance - 1

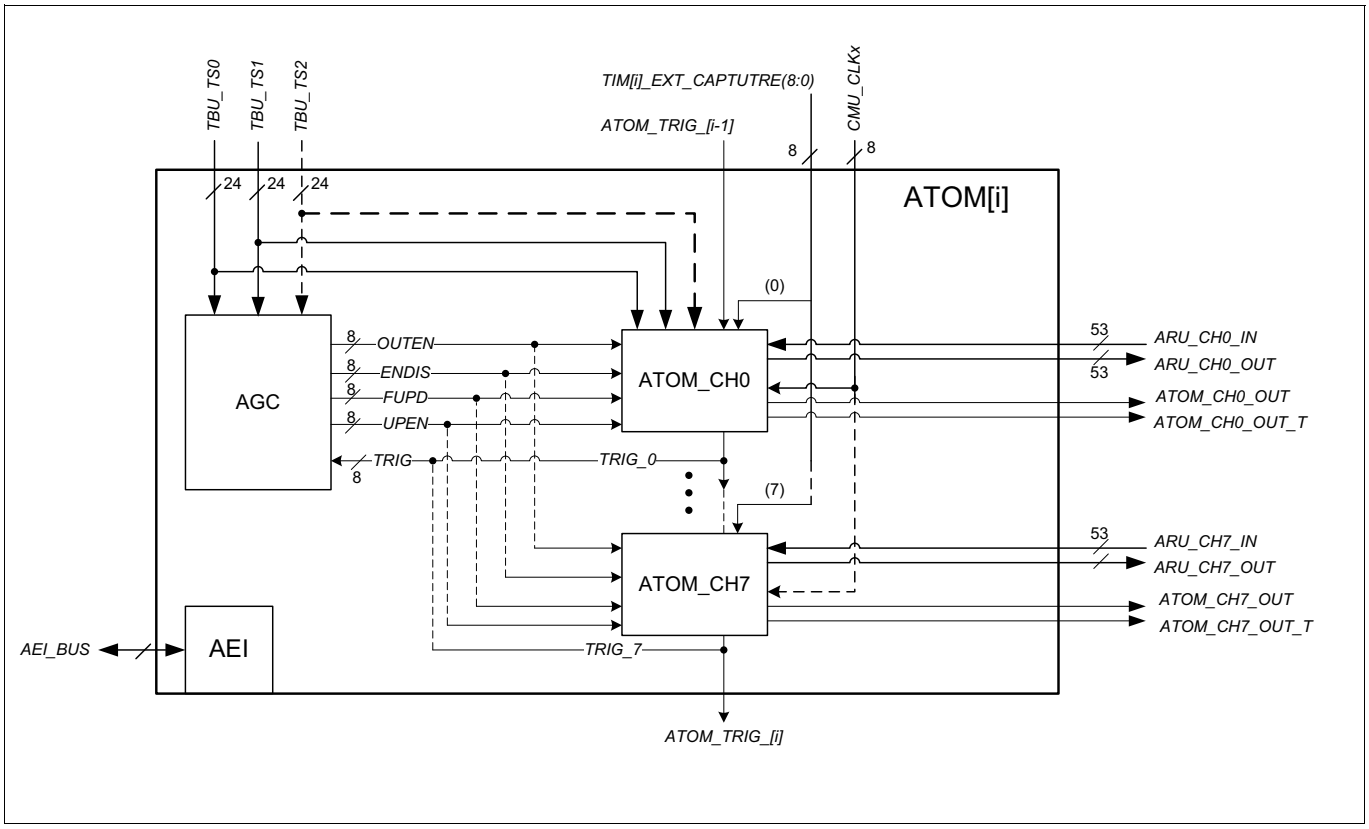


Figure 62 ATOM block diagram

The architecture of the ATOM sub-module is similar to the TOM sub-module, but there are some differences. First, the ATOM integrates only eight output channels. Hence, there exists one ATOM Global Control sub-unit (AGC) for the ATOM channels. The ATOM is connected to the ARU and can set up individual read requests from the ARU and write requests to the ARU. Furthermore, the ATOM channels are able to generate signals on behalf of time stamps and the ATOM channels are able to generate a serial output signal on behalf of an internal shift register.

Each ATOM channel provides five modes of operation:

- ATOM Signal Output Mode Immediate (SOMI)
- ATOM Signal Output Mode Compare (SOMC)
- ATOM Signal Output Mode PWM (SOMP)
- ATOM Signal Output Mode Serial (SOMS)

- ATOM Signal Output Mode Buffered Compare (SOMB)

These modes are described in more detail in **Section 28.15.3**.

The ATOM channels' operation registers (e.g. counter, compare registers) are 24 bit wide. Moreover, the input clocks for the ATOM channels come from the configurable *CMU_CLKx* signals of the CMU sub-module. This gives the freedom to select a programmable input clock for the ATOM channel counters. The ATOM channel is able to generate a serial bit stream, which is shifted out at the *ATOM[i]_CH[x]_OUT* output. When configured in this serial shift mode (SOMS) the selected CMU clock defines the shift frequency.

Each ATOM channel provides a so called *operation* and *shadow* register set. With this architecture it is possible to work with the operation register set, while the shadow register set can be reloaded with new parameters over CPU and/or ARU.

When update via ARU is selected, it is possible to configure for ATOM SOMP mode if both shadow registers are updated via ARU or only one of the shadow registers is updated.

On the other hand, the shadow registers can be used to provide data to the ARU when one or both of the compare units inside an ATOM channel match. This feature is only applicable in SOMC mode.

In TOM channels it is possible to reload the content of the operation registers with the content of the corresponding shadow registers and change the clock input signal for the counter register simultaneously. This simultaneous change of the input clock frequency together with reloading the operation registers is also implemented in the ATOM channels.

In addition to the feature that the CPU can select another *CMU_CLKx* during operation (i.e. updating the shadow register bit field CLK_SRC_SR of the **ATOM[i]_CH[x]_CTRL** register), the selection can also be changed via the ARU. Then, for the clock source update, the ACBI register bits of the **ATOM[i]_CH[x]_STAT** register are used as a shadow register for the new clock source.

In general, the behavior of the compare units CCU0 and CCU1 and the output signal behavior is controlled with the ACB bit field inside the **ATOM[i]_CH[x]_CTRL** register when the ARU connection is disabled and the behavior is controlled via ARU through the ACBI bit field of the **ATOM[i]_CH[x]_STAT** register, when the ARU is enabled.

Since the ATOM is connected to the ARU, the shadow registers of an ATOM channel can be reloaded via the ARU connection or via CPU over its AEI interface. When loaded via the ARU interface, the shadow registers act as a buffer between the ARU and the channel operation registers. Thus, a new parameter set for a PWM can be reloaded via ARU into the shadow registers, while the operation registers work on the actual parameter set.

The trigger signal *ATOM_TRIG_[i-1]* of ATOM instance i comes from the preceding instance i-1, the trigger *ATOM_TRIG_[i]* is routed to succeeding instance i+1. Note, ATOM0 is connected to its own output ATOM_TRIG_0, i.e. the last channel of ATOM instance 0 can trigger the first channel of ATOM instance 0 (this path is registered, which means delayed by one SYS_CLK period).

28.15.1.1 ATOM Global Control (AGC)

Synchronous start, stop and update of work register of up to 8 channels is possible with the AGC sub-unit. This sub-unit has the same functionality as the TGC sub-unit of the TOM sub-module.

28.15.1.1.1 Overview

There exists one global channel control unit (AGC) to drive a number of individual ATOM channels synchronously by external or internal events.

An AGC can drive up to eight ATOM channels.

The ATOM sub-module supports four different kinds of signaling mechanisms:

- Global enable/disable mechanism for each ATOM channel with control register **ATOM[i]_AGC_ENDIS_CTRL** and status register **ATOM[i]_AGC_ENDIS_STAT**

- Global output enable mechanism for each ATOM channel with control register **ATOM[i]_AGC_OUTEN_CTRL** and status register **ATOM[i]_AGC_OUTEN_STAT**
- Global force update mechanism for each ATOM channel with control register **ATOM[i]_AGC_FUPD_CTRL**
- Update enable of the register **CM0**, **CM1** and **CLK_SRC** for each ATOM channel with the control bit field **UPEN_CTRL[z]** of **ATOM[i]_AGC_GLB_CTRL**

28.15.1.1.2 AGC Sub-unit

Each of the first three individual mechanisms (enable/disable of the channel, output enable and force update) can be driven by three different trigger sources. The three trigger sources are:

- the host CPU (bit **HOST_TRIG** of register **ATOM[i]_AGC_GLB_CTRL**)
- the TBU time stamp (signal *TBU_TS0…2* if available)
- the internal trigger signal *TRIG* (bunch of trigger signals *TRIG_[x]* which can be either the trigger *TRIG_CCU0* of channel x, the trigger of preceding channel x-1 (i.e. signal *TRIG_[x-1]*) or the external trigger *TIM_EXT_CAPTURE(x)* of assigned TIM channel x.

The first way is to trigger the control mechanism by a direct register write access via host CPU (bit **HOST_TRIG** of register **AOM[i]_AGC_GLB_CTRL**).

The second way is provided by a compare match trigger on behalf of a specified time base coming from the module TBU (selected by bits **TBU_SEL**) and the time stamp compare value defined in the bit field **ACT_TB** of register **ATOM[i]_AGC_ACT_TB**. Note, a cyclic event compare of **ACT_TB** and selected *TBU_TS*[*x*] is performed.

The third possibility is the input *TRIG* (bunch of trigger signals *TRIG_[x]*) coming from the ATOM channels 0 to 7.

The corresponding trigger signal *TRIG_[x]* coming from channel [x] can be masked by the register **ATOM[i]_AGC_INT_TRIG**.

To enable or disable each individual ATOM channel, the registers **ATOM[i]_AGC_ENDIS_CTRL** and/or **ATOM[i]_AGC_ENDIS_STAT** have to be used.

The register **ATOM[i]_AGC_ENDIS_STAT** controls directly the signal *ENDIS*. A write access to this register is possible.

The register **ATOM[i]_AGC_ENDIS_CTRL** is a shadow register that overwrites the value of register **AOM[i]_AGC_ENDIS_STAT** if one of the three trigger conditions matches.

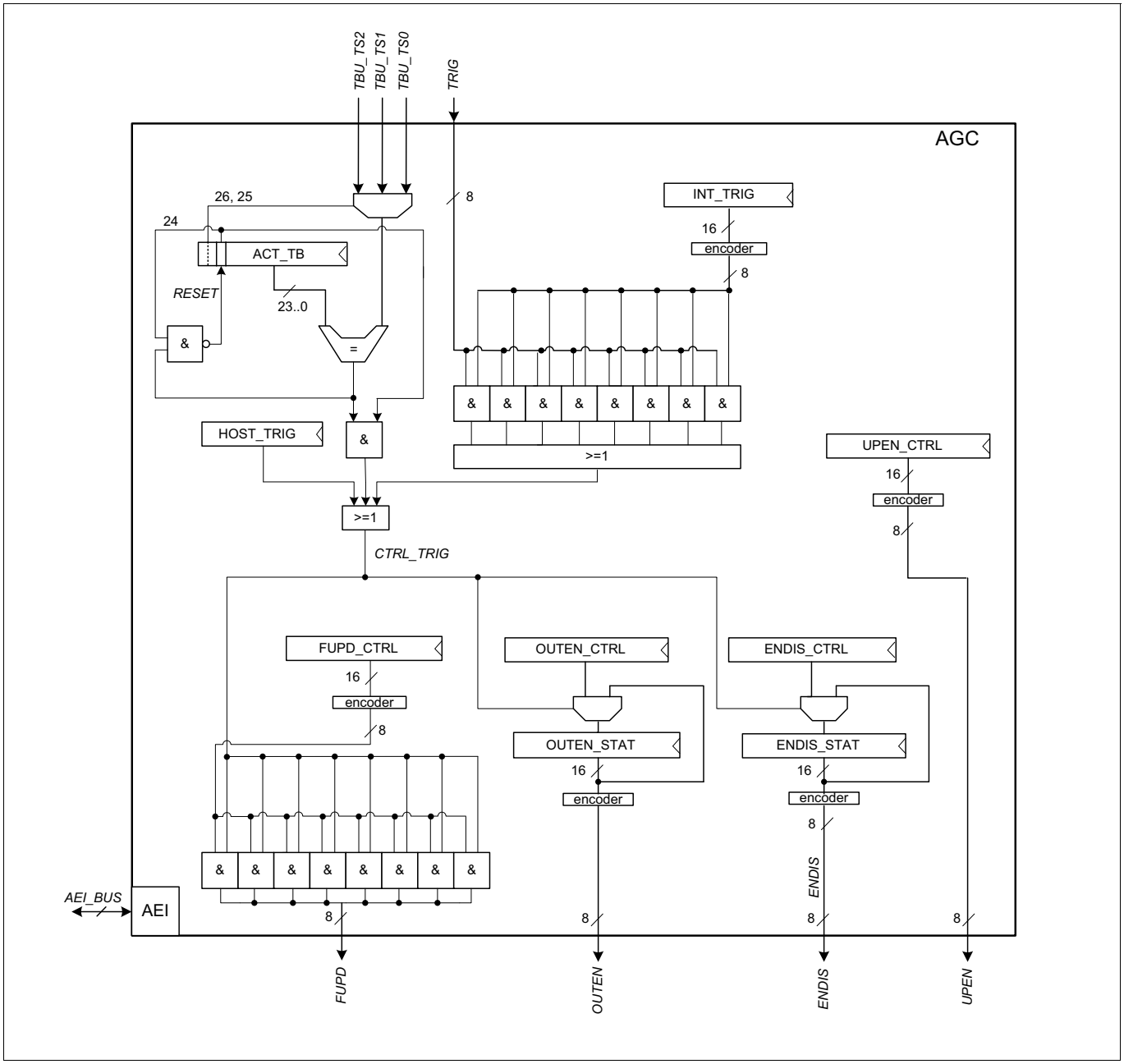


Figure 63 ATOM Global channel control mechanism

The output of the individual ATOM channels can be controlled using the register **ATOM[i]_AGC_OUTEN_CTRL** and **ATOM[i]_AGC_OUTEN_STAT**.

The register **ATOM[i]_AGC_OUTEN_STAT** controls directly the signal *OUTEN*. A write access to this register is possible.

The register **ATOM[i]_AGC_OUTEN_CTRL** is a shadow register that overwrites the value of register **ATOM[i]_AGC_OUTEN_STAT** if one of the three trigger conditions matches.

If an ATOM channel is disabled by the register **ATOM[i]_AGC_OUTEN_STAT**, the actual value of the channel output at *ATOM_CH[x]_OUT* is defined by the signal level bit (**SL**) defined in the channel control register **ATOM[i]_CH[x]_CTRL**.If the output is enabled, the output at *ATOM_CH[x]_OUT* depends on value of FlipFlop **SOUR**.

The register **ATOM[i]_AGC_FUPD_CTRL** defines which of the ATOM channels receive a *FORCE_UPDATE* event if the trigger signal *CTRL_TRIG* is raised. Note: In SOMP mode the force update request is stored and executed synchronized to the selected CMU_CLK. In all other modes the force update request is executed immediately.

The register bits **UPEN_CTRL[x]** defines for which ATOM channel the update of the working register **CM0**, **CM1** and **CLK_SRC** by the corresponding shadow register **SR0**, **SR1** and **CLK_SRC_SR** is enabled. If update is enabled, the register **CM0**, **CM1** and **CLK_SRC** will be updated on reset of counter register **CN0** (see **Figure 64**).

28.15.1.2 ATOM Channel Mode Overview

Each ATOM channel offers the following different operation modes:

In ATOM Signal Output Mode Immediate (**SOMI**), the ATOM channels generate an output signal immediately after receiving an ARU word according to the two signal level output bits of the ARU word received through the ACBI bit field. Due to the fact, that the ARU destination channels are served in a round robin order, the output signal can jitter in this mode with a jitter of the ARU round trip time.

In ATOM Signal Output Mode Compare (**SOMC**), the ATOM channel generates an output signal on behalf of time stamps that are located in the ATOM operation registers. These time stamps are compared with the time stamps, the TBU generates. The ATOM is able to receive new time stamps either by CPU or via the ARU. The new time stamps are directly loaded into the channels operation register. The shadow registers are used as capture registers for two time base values, when a compare match of the channels operation registers occurs.

In ATOM Signal Output Mode PWM (**SOMP**), the ATOM channel is able to generate simple and complex PWM output signals like the TOM sub-module by comparing its operation registers with a sub-module internal counter. In difference to the TOM, the ATOM shadow registers can be reloaded by the CPU and by the ARU in the background, while the channel operates on the operation registers.

In ATOM Signal Output Mode Serial (**SOMS**), the ATOM channel generates a serial output bit stream on behalf of a shift register. The number of bits shifted and the shift direction is configurable. The shift frequency is determined by one of the *CMU_CLKx* clock signals. Please refer to **Section 28.15.3.4** for further details.

In ATOM Signal Output Buffered Compare (**SOMB**), the ATOM channel generates an output signal on behalf of time stamps that located in the ATOM operation registers. These time stamps are compared with the time stamps, the TBU generates. The ATOM is able to receive new compare values either by CPU or via the ARU. The new compare values received via ARU are stored first in the shadow register and only if previous compare match is occurred, the operation register are updated with the content of the shadow register.

28.15.2 ATOM Channel Architecture

Each ATOM channel is able to generate output signals according to five operation modes. The architecture of the ATOM channels is similar to the architecture of the TOM channels. The general architecture of an ATOM channel is depicted in **Figure 64**.

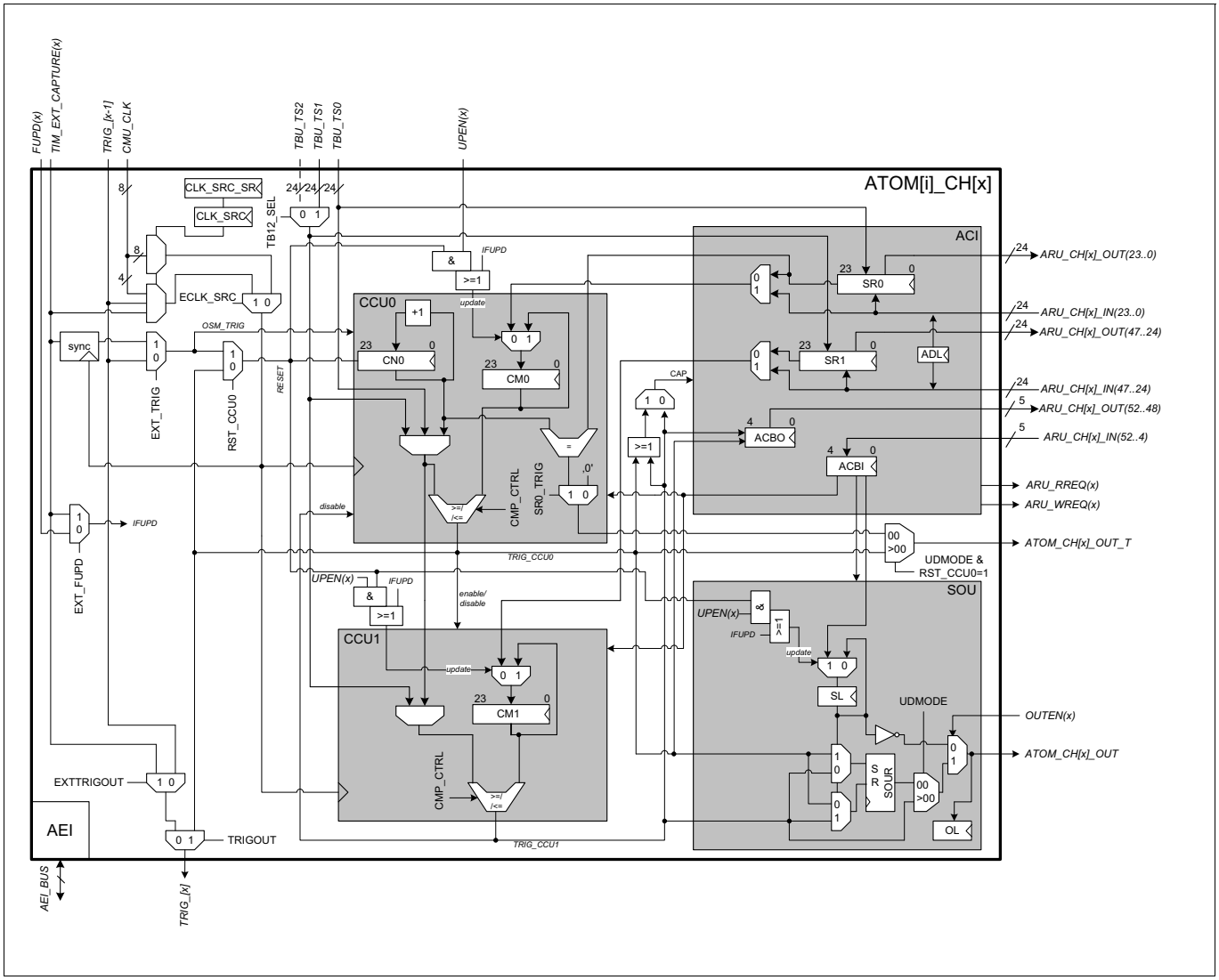


Figure 64 ATOM channel architecture

In all ATOM channels the operation registers **CN0**, **CM0** and **CM1** and the shadow registers **SR0** and **SR1** are the 24 bit width. The comparators inside CCU0 and CCU1 provide a selectable signed greater-equal or less-equal comparison to compare against the GTM time bases *TBU_TS0*, *TBU_TS1* and, if available, *TBU_TS2*. Please refer to TBU (chapter: “Time Base Unit”) for further details. The CCU0 and CCU1 units have different tasks for the different ATOM channel modes.

The cyclic event compare is used to detect time base overflows and to guarantee, that a compare match event can be set up for the future even when the time base will first overflow and then reach the compare value. Please note, that for a correct behavior of this cyclic event compare, the new compare value must not be specified larger/smaller than half of the range of the total time base value (0x7FFFFF).

In SOMC/SOMB mode, the two compare units CCUx can be used in combination to each other. When used in combination, the trigger lines *TRIG_CCU0* and *TRIG_CCU1* can be used to enable/disable the other compare unit on a match event. Please refer to **Section 28.15.3.2** and **Section 28.15.3.5** for further details.

The Signal Output Unit (SOU) generates the output signal for each ATOM channel. This output signal level depends on the ATOM channel mode and on the **SL** bit of the **ATOM[i]_CH[x]_CTRL** register in combination with the two control bits. These two control bits **ACB(1)** and **ACB(0)** can either be received via CPU in the ACB register field of the **ATOM[i]_CH[x]_CTRL** register or via ARU in the ACBI bit field of the **ATOM[i]_CH[x]_STAT** register.

The **SL** bit in the **ATOM[i]_CH[x]_CTRL** register defines in all modes the operational behavior of the ATOM channel.

When the channel and its output are disabled, the output signal level of the channel is the inverse of the **SL** bit.

In SOMI, SOMC and SOMB mode the output signal level depends on the **SL**, **ACB0** and **ACB1** bits. In SOMP mode the output signal level depends on the two trigger signals *TRIG_CCU0* and *TRIG_CCU1* since theses two triggers define the PWM timing characteristics and the **SL** bit defines the level of the duty cycle. In SOMS mode the output signal level is defined by the bit pattern that has to be shifted out by the ATOM channel. The bit pattern is located inside the **CM1** register.

The ARU Communication Interface (ACI) sub-unit is responsible for requesting data routed through ARU to the ATOM channel in SOMI, SOMP, SOMB and SOMS modes, and additionally for providing data to the ARU in SOMC mode.

In SOMC mode the ACI shadow registers have a different behavior and are used as output buffer registers for data send to ARU.

28.15.2.1 ARU Communication Interface

The ATOM channels have an ARU Communication Interface (ACI) sub-unit. This sub-unit is responsible for data exchange from and to the ARU. This is done with the two implemented registers **SR0**, **SR1**, and the **ACBI** and **ACBO** bit fields that are part of the **ATOM[i]_CH[x]_STAT** register. The ACI architecture is shown in **Figure 65**.

If the **ARU_EN** bit is set inside the **ATOM[i]_CH[x]_CTRL** register, the ATOM channel is enabled by setting the enable bits inside the **ATOM[i]_AGC_ENDIS_STAT** register and the CPU hasn't written data not equal to zero into the **CM0**, **CM1, SR0, SR1** register, the ATOM channel will first request data from the ARU before the signal generation starts in SOMP, SOMS, SOMC and SOMB mode.

Note: if in SOMP mode there is data inside the **CM0** or **SR0** register not equal to 0 the channel counter **CN0** will start counting immediately, regardless whether the channel has received ARU data yet.

Note: if in SOMS mode there is data inside the **CM0** or **SR0** register not equal to 0 the channel will start shifting immediately, regardless whether the channel has received ARU data yet.

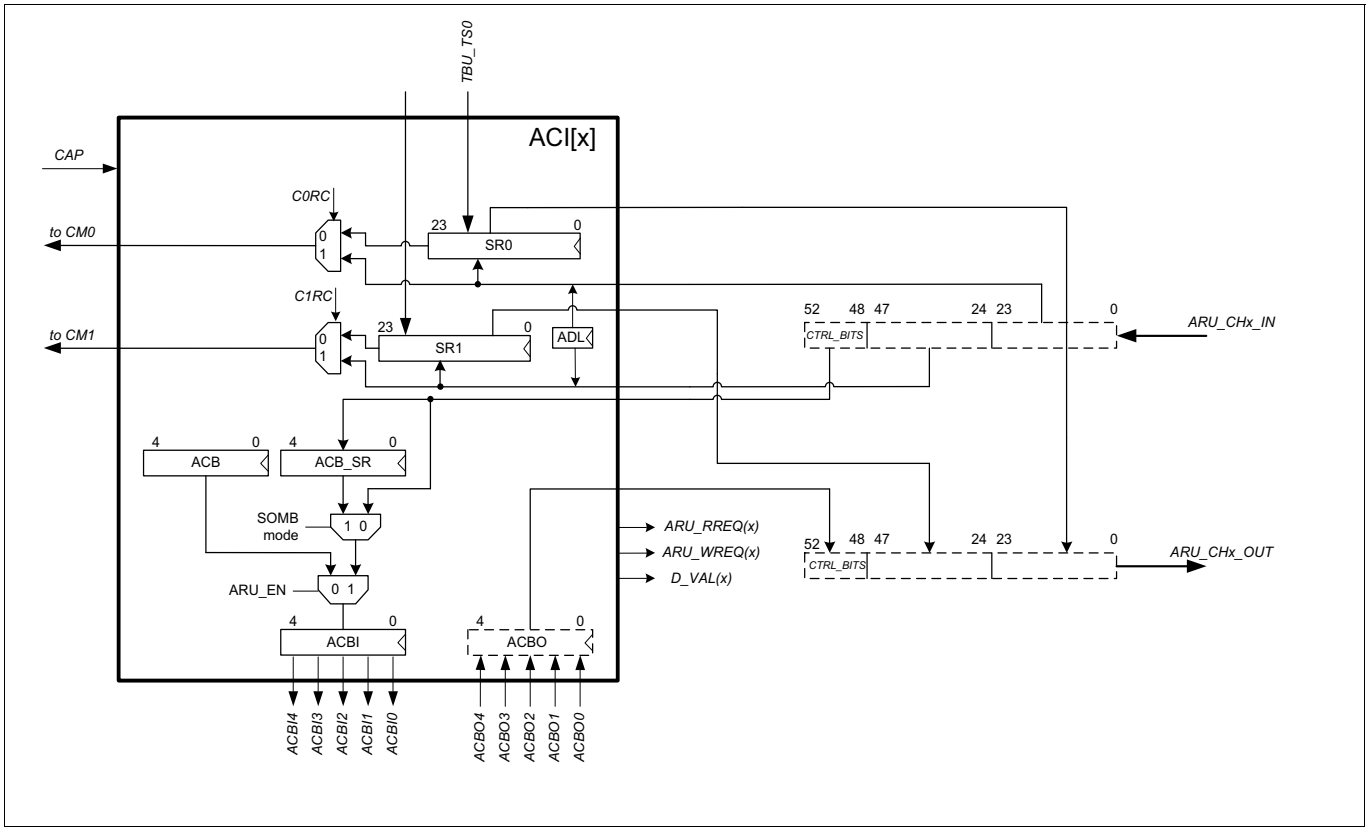


Figure 65 ACI architecture overview

Incoming ARU data (53 bit width signal *ARU_CHx_IN*) is split into three parts by the ACI and communicated to the ATOM channel registers. In SOMI, SOMP, SOMS and SOMB modes incoming ARU data *ARU_CHx_IN* is split in a way that the lower 24 bits of the ARU data (23 downto 0) are stored in the **SR0** register, the upper bits (47 downto 24) are stored in the **SR1** register. The bits 52 downto 48 (*CTRL_BITS*) are stored in SOMI, SOMP and SOMS mode in the **ACBI** bit field of the register **ATOM[i]_CH[x]_STAT**, in SOMB mode in the internal ACB_SR register.

The ATOM channel has to ensure, that in a case when the channel operation registers **CM0** and **CM1** are updated with the **SR0** and **SR1** register content and an ARU transfer to these shadow registers happens in parallel that either the old data in both shadow registers is transferred into the operation registers or both new values from the ARU are transferred.

In SOMC mode incoming ARU data *ARU_CHx_IN* is written directly to the ATOM channel operation register in the way that the lower 24 bits (23 down to 0) are written to **CM0**, and the bits 47 down to 24 are written to register **CM1**. The bits 52 down to 48 are stored in the **ACBI** bit field of the **ATOM[i]_CH[x]_STAT** register and control the behavior of the compare units and the output signal of the ATOM channel.

In SOMC mode the **SR0** and **SR1** registers serve as capture registers for the time stamps coming from TBU whenever a compare match event is signaled by the CCU0 and/or CCU1 sub-units via the *CAP* signal line. These two time stamps are then provided together with actual ATOM channel status information located in the **ACBO** bit field to the ARU at the dedicated ARU write address of the ATOM channel when the ARU is enabled.

The encoding of the ARU control bits in the different ATOM operation modes is described in more detail in the following chapters.

28.15.3 ATOM Channel Modes

As described above, each ATOM channel can operate independently from each other in one of five dedicated output modes:

- ATOM Signal Output Mode Immediate (SOMI)
- ATOM Signal Output Mode Compare (SOMC)
- ATOM Signal Output Mode PWM (SOMP)
- ATOM Signal Output Mode Serial (SOMS)
- ATOM Signal Output Mode Buffered Compare (SOMB)

The Signal Output Mode PWM (SOMP) is principally the same like the output mode for the TOM sub-module. In addition, it is possible to reload the shadow registers via the ARU without the need of a CPU interaction. The other modes provide additional functionality for signal output control. All operation modes are described in more detail in the following sections.

Note that in any output mode, if a channel is enabled, one-shot mode is disabled (**OSM**=0; only used in modes SOMP and SOMS) and CM0 >= CN0, the counter CN0 is incrementing until it reaches CM0.To avoid unintended counting of **CN0** after enabling a channel, it is recommended to reset a channel (or at least CN0 and CM0) before any change on the mode bits **MODE**, **ARU_EN** and **OSM**.

28.15.3.1 ATOM Signal Output Mode Immediate (SOMI)

In ATOM Signal Output Mode Immediate (SOMI), the ATOM channel generates output signals on the *ATOM[i]_CH[x]_OUT* output port immediate after update of the bit **ACBI(0)** of register **ATOM[i]_CH[x]_STAT** or **ACB(0)** bit of register **ATOM[i]_CH[x]_CTRL**.

If ARU access is enabled by setting bit ARU_EN in register **ATOM[i]_CH[x]_CTRL**, the update of the output *ATOM[i]_CH[x]_OUT* depends on the bit ACBI(0) of register **ATOM[i]_CH[x]_STAT** received at the ACI sub-unit and the bit **SL** bit of register **ATOM[i]_CH[x]_CTRL**. The remaining 48 ARU bits (47 downto 0) have no meaning in this mode.

If ARU access is disabled, the update of the output *ATOM[i]_CH[x]_OUT* depends on the bit **ACB(0)** and the bit **SL** of register **ATOM[i]_CH[x]_CTRL**.

The initial ATOM channel port pin *ATOM[i]_CH[x]_OUT* signal level has to be specified by the **SL** bit field of the **ATOM[i]_CH[x]_CTRL** register when **OUTEN_CTRL** register bit field **OUTEN_CTRLx** is disabled (see **Section 28.15.6.5**) for details.

In SOMI mode the output behavior depends on the **SL** bit of register **ATOM[i]_CH[x]_CTRL** and the bit **ACBI(0)** of the **ATOM[i]_CH[x]_STAT** register or the bit **ACB0** of register **ATOM[i]_CH[x]_CTRL**:

Table 50 Output behavior in SOMI mode

SL	ACBI(0)/ ACB(0)	Output behavior
0	0	Set output to inverse of SL (1)
0	1	Set output to SL (0)
1	0	Set output to inverse of SL (0)
1	1	Set output to SL (1)

The signal level bit **ACBI(0)** is transferred to the SOU sub-unit of the ATOM and made visible at the output port according to the table above immediately after the data was received by the ACI. This can introduce a jitter on the output signal since the ARU channels are served in a time multiplexed fashion.

28.15.3.1.1 Register ATOM[i]_CH[x]_CTRL in SOMI mode

Register ATOM[i]_CH[x]_CTRL in SOMI mode

GTM_ATOMi_CHx_SOMI (i=0-11; x=0-7)

ATOMi Channel x Control Register in SOMI Mode ($E8004_H$ + i*800_H+ x*80_H) **Reset Value: 00000800_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREEZE	Not_used	Not_used	Reserved	Not_used	Not_used	Not_used	Not_used	Not_used			Not_used	Not_used		Not_used	Not_used
rw	rw	rw	r	rw	rw	r	rw	rw			rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not_used	Not_used			SL	Not_used	Not_used	Not_used				ACB0	ARU_EN	Not_used	MODE	
rw	rw			rw	rw	rw	rw				rw	rw	rw	rw	

Field	Bits	Type	Description
MODE	1:0	rw	**ATOM channel mode select.** 00_B ATOM Signal Output Mode Immediate (SOMI)
Not_used	2	rw	**Not used** Note: Not used in this mode.
ARU_EN	3	rw	**ARU Input stream enable** 0_B ARU Input stream disabled 1_B ARU Input stream enabled
ACB0	4	rw	**ACB bit 0** 0_B Set output to inverse of SL bit 1_B Set output to SL bit
Not_used	8:5	rw	**Not used** Note: Not used in this mode.
Not_used	9	rw	**Not used** Note: Not used in this mode.
Not_used	10	rw	**Not used** Note: Not used in this mode.
SL	11	rw	**Initial signal level after channel is enabled** 0_B Low signal level 1_B High signal level Reset value depends on the hardware configuration chosen by silicon vendor. If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL. If FREEZE=0, following note is valid: If the channel is disabled, the output register of SOU unit is set to inverse value of SL. If FREEZE=1, following note is valid: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.

Field	Bits	Type	Description
Not_used	14:12	rw	**Not used** Note: Not used in this mode.
Not_used	15	rw	**Not used** Note: Not used in this mode.
Not_used	16	rw	**Not used** Note: Not used in this mode.
Not_used	17	rw	**Not used** Note: Not used in this mode.
Not_used	19:18	rw	**Not used** Note: Not used in this mode.
Not_used	20	rw	**Not used** Note: Not used in this mode.
Not_used	23:21	rw	**Not used** Note: Not used in this mode.
Not_used	24	rw	**Not used** Note: Not used in this mode.
Not_used	25	r	**Not used** Note: Not used in this mode.
Not_used	26	rw	**Not used** Note: Not used in this mode.
Not_used	27	rw	**Not used** Note: Not used in this mode.
Reserved	28	r	**Reserved** Read as zero, should be written as zero.
Not_used	29	rw	**Not used** Note: Not used in this mode.
Not_used	30	rw	**Not used** Note: Not used in this mode.
FREEZE	31	rw	**FREEZE** 0_B a channel disable/enable may change internal register and output register 1_B a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

28.15.3.2 ATOM Signal Output Mode Compare (SOMC)

28.15.3.2.1 Overview

In ATOM Signal Output Mode Compare (SOMC) the output action is performed in dependence of the comparison between input values located in **CM0** and/or **CM1** registers and the two (three) time base values *TBU_TS0* or *TBU_TS1* (or *TBU_TS2*) provided by the TBU. For a description of the time base generation please refer to the TBU specification in chapter “Time Base Unit”. It is configurable, which of the two (three) time bases is to be compared with one or both values in **CM0** and **CM1**.

The behavior of the two compare units CCU0 and CCU1 is controlled either with the bits 4 downto 2 of **ACB** bit field inside the **ATOM[i]_CH[x]_CTRL** register, when the ARU connection is disabled or with the ACBI bit field of the **ATOM[i]_CH[x]_STAT** register, when the ARU is enabled. In that case the **ACB** bit field is updated via the ARU control bits 52 downto 48.

The CCUx trigger signals *TRIG_CCU0* and *TRIG_CCU1* always create edges, dependent on the predefined signal level in **SL** bit in combination with two control bits that can be specified by either ARU or CPU within the aforementioned **ATOM[i]_CH[x]_CTRL** or **ATOM[i]_CH[x]_STAT** registers.

In SOMC mode the channel is always disabled after the specified compare match event occurred. The shadow registers are used to store two time stamp values at the match time. The channel compare can be re-enabled by first reading the shadow registers, either by CPU or ARU and by providing new data for CMx registers through CPU or ARU. For a detailed description please refer to the **Section 28.15.3.2.2** and **Section 28.15.3.2.3**.

If three time bases exist for the GTM there must be a preselection between *TBU_TS1* and *TBU_TS2* for the ATOM channel. This can be done with **TB12_SEL** bit in the **ATOM[i]_CH[x]_CTRL** register.

The comparison in CCU0/1 with time base TBU_TS1 or TBU_TS2 can be done on a greater-equal or less-equal compare according to the **CMP_CTRL** bit. This control bit has no effect to a compare unit CCU0 or CCU1 that compares against TBU_TS0. In this case always a greater-equal compare is done. The bit **CMP_CTRL** is part of the **ATOM[i]_CH[x]_CTRL** register.

When configured in SOMC mode, the channel port pin has to be initialized to an initial signal level. This initial level after enabling the ATOM channel is determined by the **SL** bit in the **ATOM[i]_CH[x]_CTRL** register. If the output is disabled, the signal level is set to the inverse level of the **SL** bit.

If the channel is disabled, the register SOUR is set to the **SL** bit in the **ATOM[i]_CH[x]_CTRL** register.

On a compare match event the shadow register **SR0** and **SR1** are used to capture the TBU time stamp values. **SR0** always holds *TBU_TS0* and **SR1** either holds *TBU_TS1* or *TBU_TS2* dependent on the **TB12_SEL** bit in the **ATOM[i]_CH[x]_CTRL** register.

Please note, that when the channel is disabled and the compare registers are written, the compare registers CMx are loaded with the written value and the channel starts with the comparison on behalf of this values, when the channel is enabled.

28.15.3.2.2 SOMC Mode under CPU control

As already mentioned above the ATOM channel can be controlled either by CPU or by ARU. When the channel should be controlled by CPU, the **ARU_EN** bit inside the **ATOM[i]_CH[x]_CTRL** register has to be reset.

The output of the ATOM channel is set on a compare match event depending on the **ACB10** bit field in combination with the **SL** bit both located in the **ATOM[i]_CH[x]_CTRL** register. The output behavior according to the **ACB10** bit field in the control register is shown in the following table:

Table 51 Output behavior according to the ACB10 bit field in the control register

SL	ACB10(5)	ACB10(4)	Output behavior
0	0	0	No signal level change at output (exception in table **Figure 67** mode ACB42=001)
0	0	1	Set output signal level to 1
0	1	0	Set output signal level to 0
0	1	1	Toggle output signal level (exception in table {REF:ATOM_1809} mode ACB42=001)
1	0	0	No signal level change at output (exception in table {REF:ATOM_1809} mode ACB42=001)

Table 51 Output behavior according to the ACB10 bit field in the control register (cont'd)

SL	ACB10(5)	ACB10(4)	Output behavior
1	0	1	Set output signal level to 0
1	1	0	Set output signal level to 1
1	1	1	Toggle output signal level (exception in table **Figure 67** mode ACB42=001)

The capture/compare strategy of the two CCUx units can be controlled with the **ACB42** bit field inside the **ATOM[i]_CH[x]_CTRL** register. The meaning of these bits is shown in the following table:

Table 52 Capture/compare strategy of the two CCUx units controlled by ACB42 bit field

ACB42(8)	ACB42(7)	ACB42(6)	CCUx control
0	0	0	Serve First: Compare in CCU0 using TBU_TS0 and in parallel in CCU1 using TBU_TS1 or TBU_TS2. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACB10(5) and ACB10(4). Details see table**Section 67**
0	0	1	Serve First: Compare in CCU0 using TBU_TS0 and in parallel in CCU1 using TBU_TS1 or TBU_TS2. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACB10(5) and ACB10(4). Details see **Section 67**
0	1	0	Compare in CCU0 only, use time base TBU_TS0. Output signal level is defined by combination of SL, ACB10(5) and ACB10(4) bits.
0	1	1	Compare in CCU1 only, use time base TBU_TS1 or TBU_TS2. Output signal level is defined by combination of SL, ACB10(5) and ACB10(4) bits.
1	0	0	Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS0. Output signal level when CCU0 matches is defined by combination of SL, ACB10(5) and ACB10(4). On the CCU1 match the output level is toggled.
1	0	1	Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU0 matches is defined by combination of SL, ACB10(5) and ACB10(4). On the CCU1 match the output level is toggled.
1	1	0	Serve Last: Compare in CCU0 using TBU_TS0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU1 matches is defined by combination of SL, ACB10(5) and ACB10(4).
1	1	1	Cancels pending comparison independent on ARU_EN

The behavior of the **ACBI**/**ACB42** bit combinations 0b000 and 0b001 is described in more detail in the tables of **Figure 66** and **Figure 67**.

ACB4	ACB3	ACB2	ACB1	ACB0	SL	CCU0 match	CCU1 match	Pin level new
0	0	0	0	0	0	0	1	hold
						1	0	hold
						1	1	hold
0	0	0	0	1	0	0	1	1
						1	0	1
						1	1	1
0	0	0	1	0	0	0	1	0
						1	0	0
						1	1	0
0	0	0	1	1	0	0	1	toggle
						1	0	toggle
						1	1	toggle
0	0	0	0	0	1	0	1	hold
						1	0	hold
						1	1	hold
0	0	0	0	1	1	0	1	0
						1	0	0
						1	1	0
0	0	0	1	0	1	0	1	1
						1	0	1
						1	1	1
0	0	0	1	1	1	0	1	toggle
						1	0	toggle
						1	1	toggle

Figure 66 ATOM CCUx Serve first definition ACB42 = 0b000

ACB4	ACB3	ACB2	ACB1	ACB0	SL	CCU0 match	CCU1 match	Pin level new
0	0	1	0	0	0	0	1	hold
						1	0	toggle
						1	1	hold
0	0	1	0	1	0	0	1	0
						1	0	1
						1	1	0
0	0	1	1	0	0	0	1	1
						1	0	0
						1	1	1
0	0	1	1	1	0	0	1	toggle
						1	0	hold
						1	1	toggle
0	0	1	0	0	1	0	1	hold
						1	0	toggle
						1	1	hold
0	0	1	0	1	1	0	1	1
						1	0	0
						1	1	1
0	0	1	1	0	1	0	1	0
						1	0	1
						1	1	0
0	0	1	1	1	1	0	1	toggle
						1	0	hold
						1	1	toggle

Figure 67 ATOM CCUx Serve first definition ACB42 = 0b001

If the ATOM channel is enabled, the **CM0** and/or **CM1** registers and the **ACB42** bit field of the **ATOM[i]_CH[x]_CTRL** register can be updated by the CPU as long as the first match event occurs in case of a ‘serve last’ compare strategy or as long as the overall match event in case of the other compare strategies.

After a compare match event that causes an update of the shadow registers **SR0**/**SR1** and before reading the **SR0** and/or **SR1** register via ARU, the update of the registers **CM0** and/or **CM1** is possible but has no effect.

To set up a new compare action, first the **SR0** and/or **SR1** register containing captured values have to be read and then new compare values have to be written into the register **CM0** and/or **CM1**.

Which **CMx** register has to be updated depends on the compare strategy defined in the **ACB42** bit field of the channel control register. Since the channel immediately starts with the comparison after the **CMx** register was/were written, the compare strategy has to be updated before the **CMx** registers are written.

For the ‘serve last’ compare strategies, if the register **CM0** and **CM1** are updated, it can happen that one or both compare values are already located in the past. In any way the ATOM channel will first wait until both compare values are written before it starts the time base comparisons to avoid a deadlock.

The CPU can check at any time if at least one of the ATOM channels' capture compare register contains valid data and waits for a compare event to happen. This is signaled by the **DV** bit inside the **ATOM[i]_CH[x]_STAT** register. Note, for ‘serve last’ compare strategies, if **DV** bit is currently not set, writing to **CM0** or **CM1** sets immediately the **DV** bit although the compare is only started if both values are written.

An exception for update of register **CM0**/**CM1** exists in SOMC mode and CCUx control mode 'serve last'. If in this mode the CCU0 compare match event occurred, the update of register **CM0**/**CM1** via CPU is blocked until the CCU1 compare match event.

In the ‘serve last’ mode (ACB42= 0b100 or ACB42=0b101) it is possible to generate very small spikes on the output pin by loading **CM0** and **CM1** with two time stamp values for *TBU_TS0*, *TBU_TS1* or *TBU_TS2* close together. The output pin will then be set or reset dependent on the **SL** bit and the specified **ACB10(5)** and **ACB10(4)** bits in the **ACB10** bit field of the **ATOM[i]_CH[x]_CTRL** register on the first match event and the output will toggle on the second compare event in the CCU1 compare unit.

It is important to note, that the bigger (smaller) time stamp has to be loaded into the **CM1** register, since the CCU0 will enable the CCU1 once it has reached its comparison time stamp. The order of the comparison time stamps depends on the defined greater-equal or less-equal comparison of the CCUx units.

In addition to storing the captured time stamps in the shadow registers, the ATOM channel provides the result of the compare match event in the **ACBO(4)** and **ACBO(3)** bits of the **ATOM[i]_CH[x]_STAT** register. The meaning of the bits is shown in the following table:

Table 53 Compare match event ACBO(4) and ACBO(3) bits of ATOM[i]_CH[x]_STAT

ACBO(4)	ACBO(3)	Indication
0	1	CCU0 compare match occurred
1	0	CCU1 compare match occurred

Please note, that in case of the 'serve last' compare strategy, when the bit **SLA** in the **ATOM[i]_CH[x]_CTRL** register is not set, the **ACBO(4)** bit is always set and the **ACBO(3)** bit is always reset after the compare match event occurred.

The **ACBO** bit field is reset, when the **DV** bit is set.

Depending on the capture compare unit where the time base matched the interrupt *CCU0TCx_IRQ* or *CCU1TCx_IRQ* is raised. Note that in case of 'serve first' compare strategy, if both events CCU0 and CCU1 occur at the same point in time, both interrupts will be raised.

The behavior of an ATOM channel in SOMC mode under CPU control is depicted in **Figure 68**.

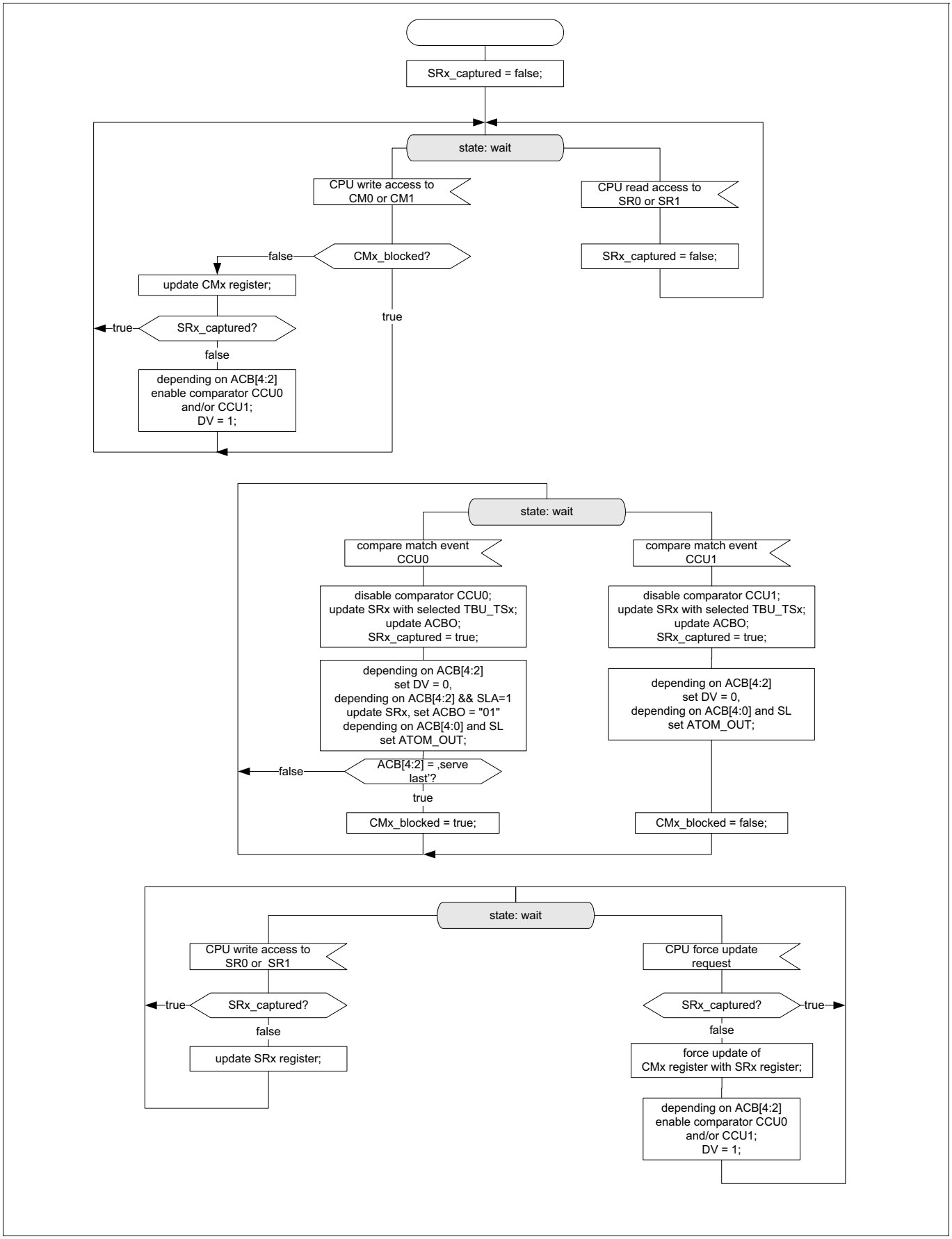


Figure 68 SOMC state diagram for channel under CPU control

28.15.3.2.3 SOMC Mode under ARU control

When the channel should be controlled by ARU, the **ARU_EN** bit inside the **ATOM[i]_CH[x]_CTRL** register has to be set.

In case, the ATOM channel is under ARU control the content for the compare registers **CM0** and **CM1** as well as the update of the compare strategy can be loaded via the 53 bit ARU word.

The ARU word 23 to 0 is loaded into the **CM0** register while the ARU word 47 to 24 is loaded into the **CM1** register. The five ARU control bits 52 to 48 are loaded into the **ACBI** bit field of the **ATOM[i]_CH[x]_STAT** register and control the channel compare strategy as well as the output behavior in case of compare match events.

For the five ARU control bits 52 to 48 the bits 49 and 48 are loaded into the **ACBI** bits 1 and 0. The output behavior also depends on the setting of the **SL** bit inside of the **ATOM[i]_CH[x]_CTRL** register and is shown in the following table:

Table 54 Output behavior depends on SL bit inside of the ATOM[i]_CH[x]_CTRL and ACBI bits 1 and 0

SL	ACBI(1)	ACBI(0)	Output behavior
0	0	0	No signal level change at output (exception in **Figure 66** and **Figure 67** mode ACB42=001)
0	0	1	Set output signal level to 1
0	1	0	Set output signal level to 0
0	1	1	Toggle output signal level (exception in **Figure 66** and **Figure 67** mode ACB42=001)
1	0	0	No signal level change at output (exception in **Figure 66** and **Figure 67** mode ACB42=001)
1	0	1	Set output signal level to 0
1	1	0	Set output signal level to 1
1	1	1	Toggle output signal level (exception in **Figure 66** and **Figure 67** mode ACB42=001)

For the five ARU control bits 52 to 48 the bits 52 to 50 are loaded into the **ACBI** bits 4 to 2. With these three bits the capture/compare units CCUx can be controlled as shown in the following table:

Table 55 Capture/compare units CCUx controlled by ACBI bits 4 to 2

ACBI(4)	ACBI(3)	ACBI(2)	CCUx control
0	0	0	Serve First: Compare in CCU0 using TBU_TS0 and in parallel in CCU1 using TBU_TS1 or TBU_TS2. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACBI(1) and ACBI(0). Details see **Figure 67**
0	0	1	Serve First: Compare in CCU0 using TBU_TS0 and in parallel in CCU1 using TBU_TS1 or TBU_TS2. Disable other CCUx on compare match. Output signal level on the compare match of the matching CCUx unit is defined by combination of SL, ACBI(1) and ACBI(0). Details see **Figure 66**
0	1	0	Compare in CCU0 only, use time base TBU_TS0. Output signal level is defined by combination of SL, ACBI(1) and ACBI(0) bits.

Table 55 Capture/compare units CCUx controlled by ACBI bits 4 to 2 (cont'd)

ACBI(4)	ACBI(3)	ACBI(2)	CCUx control
0	1	1	Compare in CCU1 only, use time base TBU_TS1 or TBU_TS2. Output signal level is defined by combination of SL, ACBI(1) and ACBI(0) bits.
1	0	0	Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS0. Output signal level when CCU0 matches is defined by combination of SL, ACBI(1) and ACBI(0). On the CCU1 match the output level is toggled.
1	0	1	Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU0 matches is defined by combination of SL, ACBI(1) and ACBI(0). On the CCU1 match the output level is toggled.
1	1	0	Serve Last: Compare in CCU0 using TBU_TS0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU1 matches is defined by combination of SL, ACBI(1) and ACBI(0).
1	1	1	Change ARU read address to ATOM_RDADDR1 DV flag is not set. Neither ACBI(1) nor ACBI(0) is evaluated.

It is important to note that the bit combination 0b111 for the **ACBI(4)**, **ACBI(3)** and **ACBI(2)** bits forces the channel to request new compare values from another destination read address defined in the **ATOM_RDADDR1** bit field of the **ATOM[i]_CH[x]_RDADDR** register. After data was successfully received and the compare event occurred the ATOM channel switches back to **ATOM_RDADDR0** to receive the next data from there.

After the specified compare match event, the captured time stamps are stored in **SR0** and **SR1** and the compare result is stored in the **ACBO** bit field of the **ATOM[i]_CH[x]_STAT** register. The meaning of the **ACBO(4)** and **ACBO(3)** bits of the **ATOM[i]_CH[x]_STAT** is shown in the following table:

Table 56 Compare match event ACBO(4) and ACBO(3) bits of ATOM[i]_CH[x]_STAT

ACBO(4)	ACBO(3)	Return value to ARU
0	1	CCU0 compare match occurred
1	0	CCU1 compare match occurred

Please note, that in case of the 'serve last' compare strategy, when the bit **SLA** in the **ATOM[i]_CH[x]_CTRL** register is not set, the **ACBO(4)** bit is always set and the **ACBO(3)** bit is always reset after the compare match event occurred.

The **ACBO** bit field is reset, when the **DV** bit is set.

Depending on the capture compare unit where the time base matched the interrupt *CCU0TCx_IRQ* or *CCU1TCx_IRQ* is raised.

When CCU0 and CCU1 is used for comparison it is possible to generate very small spikes on the output pin by loading **CM0** and **CM1** with two time stamp values for *TBU_TS0*, *TBU_TS1* or *TBU_TS2* close together. The output pin will then be set or reset dependent on the **SL** bit and the specified **ACBI(0)** and **ACBI(1)** bits in the **ACBI** bit field of the **ATOM[i]_CH[x]_STAT** register on the first match event and the output will toggle on the second match event.

It is important to note, that the bigger (smaller) time stamp has to be loaded into the **CM1** register, since the CCU0 will enable the CCU1 once it has reached its comparison time stamp. The order of the comparison time stamps depends on the defined greater-equal or less-equal comparison of the CCUx units.

For compare strategy 'serve last' the CCU0 and CCU1 compare match may occur sequentially. During different phases of compare match the CPU access rights to register **CM0** and **CM1** as well as to **WR_REQ** bit is different.

For the case of bit **ABM**=0 and EUPM=0 (register **ATOM[i]_CH[x]_CTRL**) these access rights by CPU to register **CM0** and **CM1** and the **WR_REQ** are depicted in the following figure.

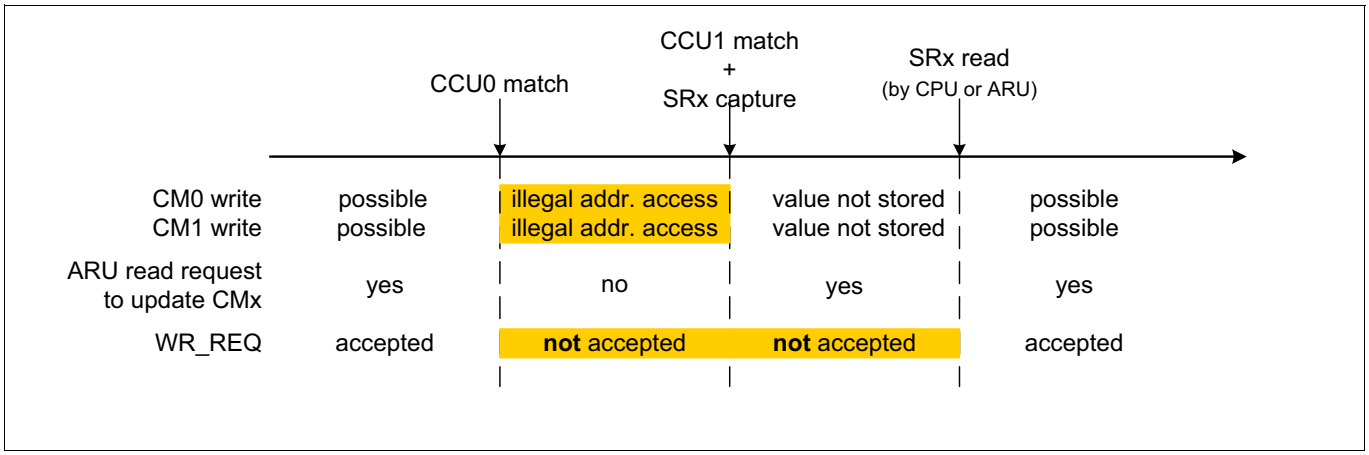


Figure 69 CPU access rights in case of compare strategy 'serve last', ABM=0 and EUPM=0

For the case of bit **ABM**=1 and **EUPM**=0 (register **ATOM[i]_CH[x]_CTRL**) these access rights by CPU to register **CM0** and **CM1** and the **WR_REQ** are depicted in the following figure.

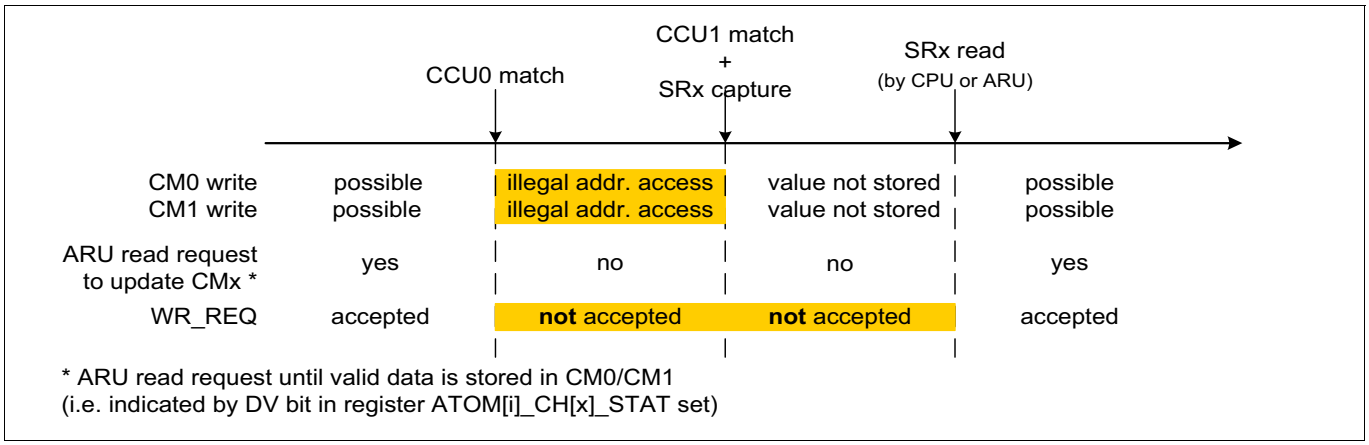


Figure 70 CPU access rights in case of compare strategy 'serve last', ABM=1 and EUPM=0

For the case of bit **EUPM**=1 (register **ATOM[i]_CH[x]_CTRL**), after CCU0 compare match (and before CCU1 compare match) an update of **CM1** as well as a late update via **WR_REQ** is possible. The value is used for compare. After CCU0 compare match an update of **CM0** is not possible, means the value is not stored. The ARU read request is not paused between the compare matches. This behavior is depicted in the following figures.

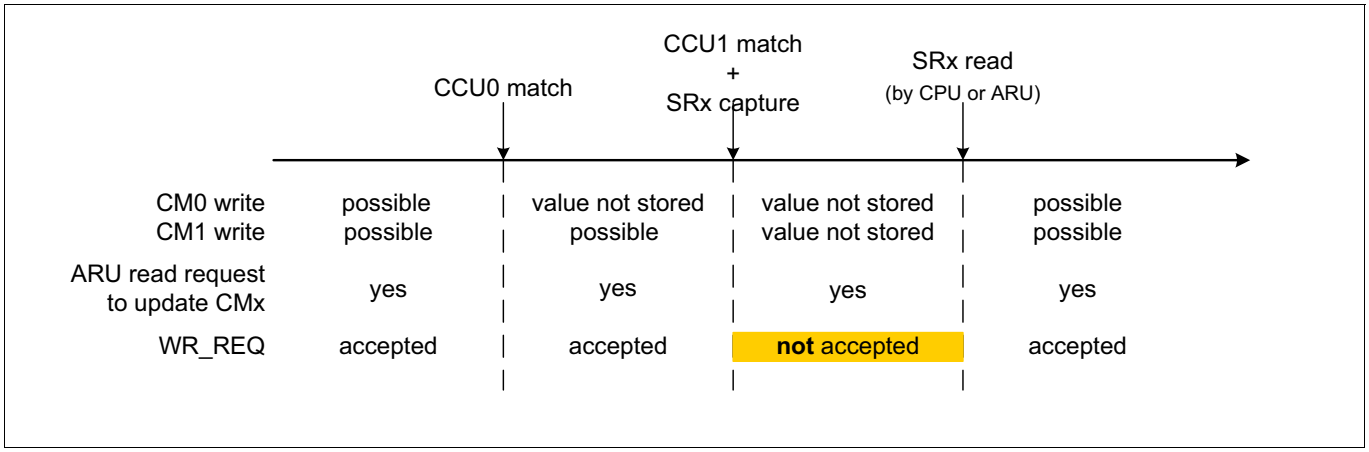


Figure 71 CPU access rights in case of compare strategy 'serve last' ,ABM=0 and EUPM=1

The behavior in case of EUPM=1 and ABM=1 is depicted in the following figure:

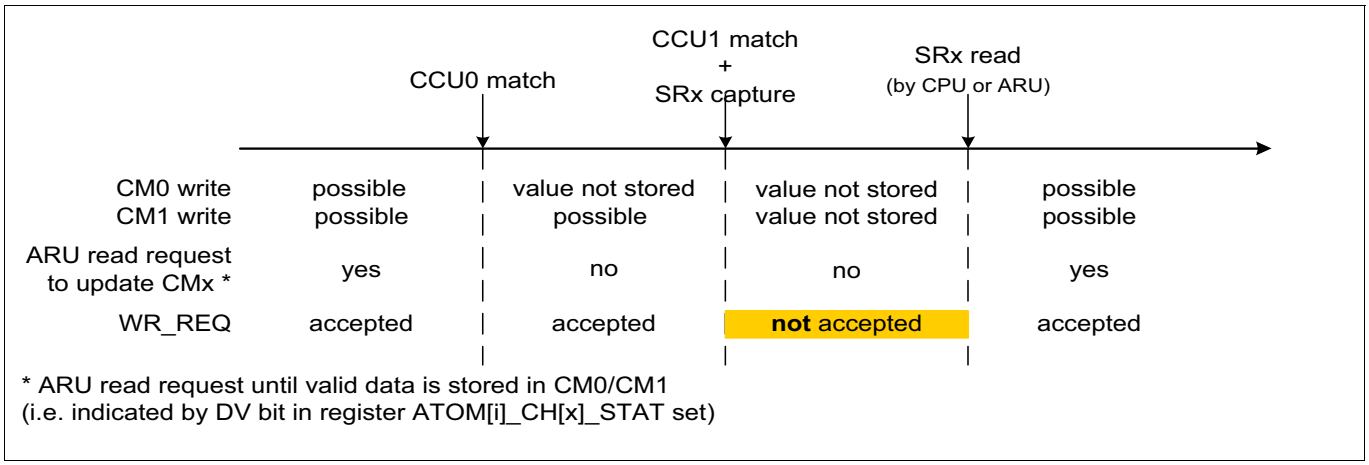


Figure 72 CPU access rights in case of compare strategy 'serve last' ,ABM=1 and EUPM=1

In case of EUPM=1 a write access to CM0 or CM1 never causes an AEI write status 0b10.

ARU Non-Blocking mode

When the compare registers are updated via ARU the update behavior of the channel is configurable with the ABM bit inside the **ATOM[i]_CH[x]_CTRL** register. When the **ABM** bit is reset, the ATOM channel is in ARU non-blocking mode.

In the ARU non-blocking mode, data received via ARU is continuously transferred to the registers **CM0** and **CM1** and the bit field **ACBI** of register **ATOM[i]_CH[x]_STAT** as long as no specified compare match event occurs.

After a compare match event that causes an update of the shadow register **SR0**/**SR1** and before reading the **SR0**/**SR1** register via CPU or ARU, the update of the registers **CM0**/ **CM1** via CPU or ARU is possible but the data is not accepted to be valid (no DV bit set in register ATOM[i]_CH[x]_CTRL).

To set up a new compare action, first the **SR0**/**SR1** register containing captured values have to be read and then new compare values have to be written into the register **CM0**/**CM1**. This can be done either by ARU or by CPU.

When the CPU does the register accesses, only one of the shadow registers has to be read. Dependent on the compare strategy, the CPU has to write one or both of the compare registers.

An exception for update of register **CM0**/**CM1** exists in SOMC mode and CCUx control mode 'serve last' if EUPM=0. If in this mode the CCU0 compare match event occurred, the update of register **CM0**/**CM1** via CPU is not possible until the CCU1 compare match event occurs.

Note that a write access to either CM0 or CM1 in this case leads to a write status 0b10.

The CPU can check at any time if the ATOM channel has received valid data from the ARU and waits for a compare event to happen. This is signaled by the **DV** bit inside the **ATOM[i]_CH[x]_STAT** register.

The behavior of an ATOM channel in SOMC mode, when ARU is enabled and ARU blocking mode is disabled is shown in **Figure 73**.

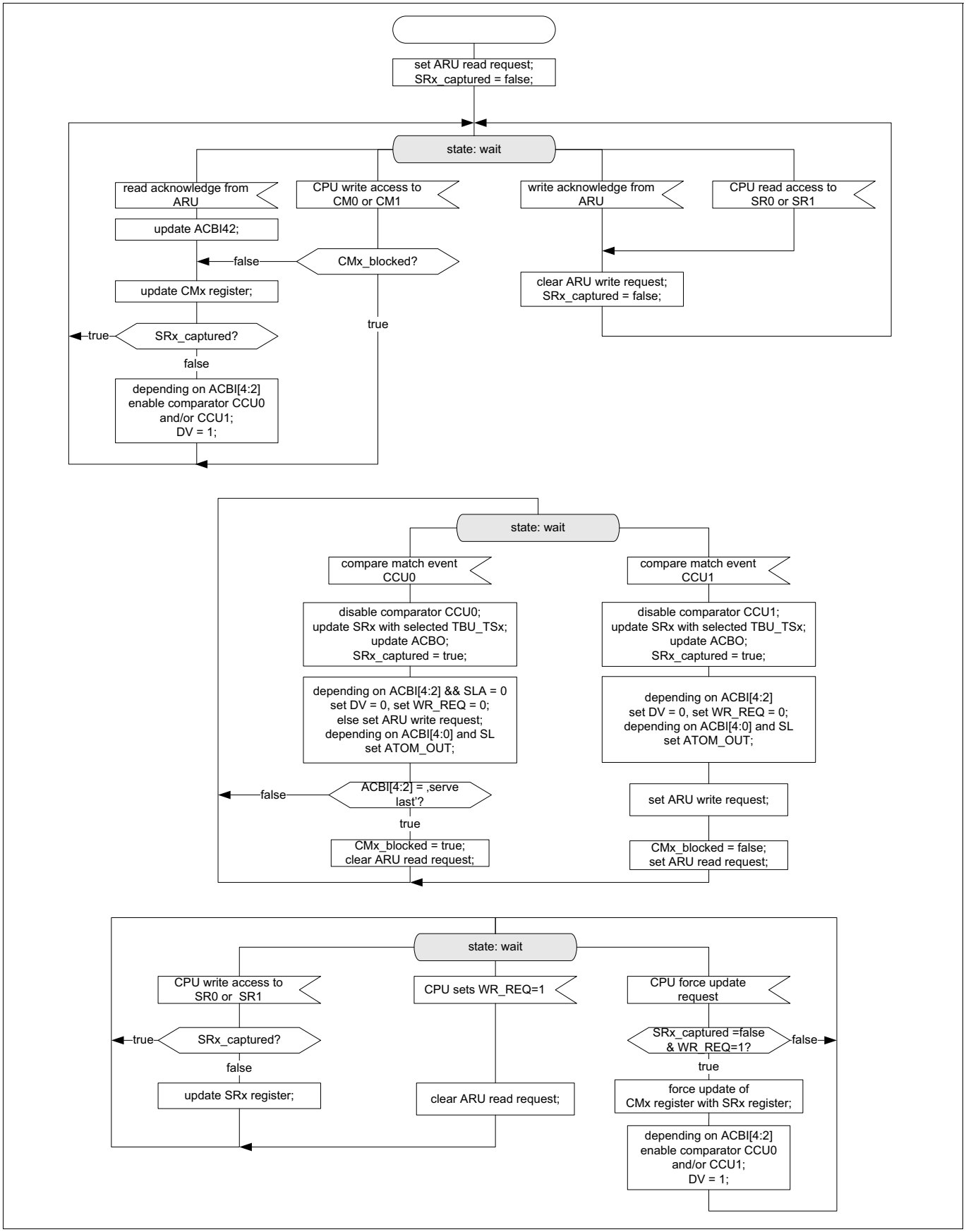


Figure 73 SOMC State diagram for SOMC mode, ARU enabled, ABM disabled

ARU Blocking mode

When the compare registers are updated by ARU, the ATOM channel can be configured to receive ARU data in a blocking manner. This can be configured by setting the **ABM** bit in the **ATOM[i]_CH[x]_CTRL** register.

If the **ABM** and **ARU_EN** bits are set, depending on compare strategy,**CM0** and/or **CM1** can be updated via ARU with new compare values. If the compare registers **CM0** and/or **CM1** are accepting these new data to be valid (indicated by bit DV in register ATOM[i]_CH[x]_STAT), the ATOM channel stops requesting new data via ARU and waits for the compare match event to happen.

When the specified compare match event happens, the shadow registers **SR0** and **SR1** are updated together with the **ACBO** bits in the **ATOM[i]_CH[x]_STAT** register. The data in the shadow registers is marked as valid for the ARU and the **DV** bit or register **ATOM[i]_CH[x]_CTRL** is reset.

If the register **SR0** and **SR1** holding the captured TBU time stamp values are read by either the ARU or the CPU, the next write access to or update of the register **CM0** or **CM1** via ARU or the CPU enables the new compare match check again.

At least one of the registers **SR0** or **SR1** has to be read either via ARU or by CPU, before new data is requested via ARU.

Note that in case of ABM=1 the application has to handle the situation that the ATOM does not request update of new data for CM0/CM1 until the captured values are read. E.g. if an MCS task starts to write via ARU new data (with AWR(I) command) after capture the data in SR0/SR1, the task sticks in the command until captured data is read by another task via ARU or via the CPU interface.

The CPU can check at any time if the ATOM channel has received valid data from the ARU and waits for a compare event to happen. This is signaled by a set **DV** bit inside the **ATOM[i]_CH[x]_STAT** register.

The behavior of an ATOM channel in SOMC mode, when ARU is enabled and ARU blocking mode is enabled is shown in **Figure 74**.

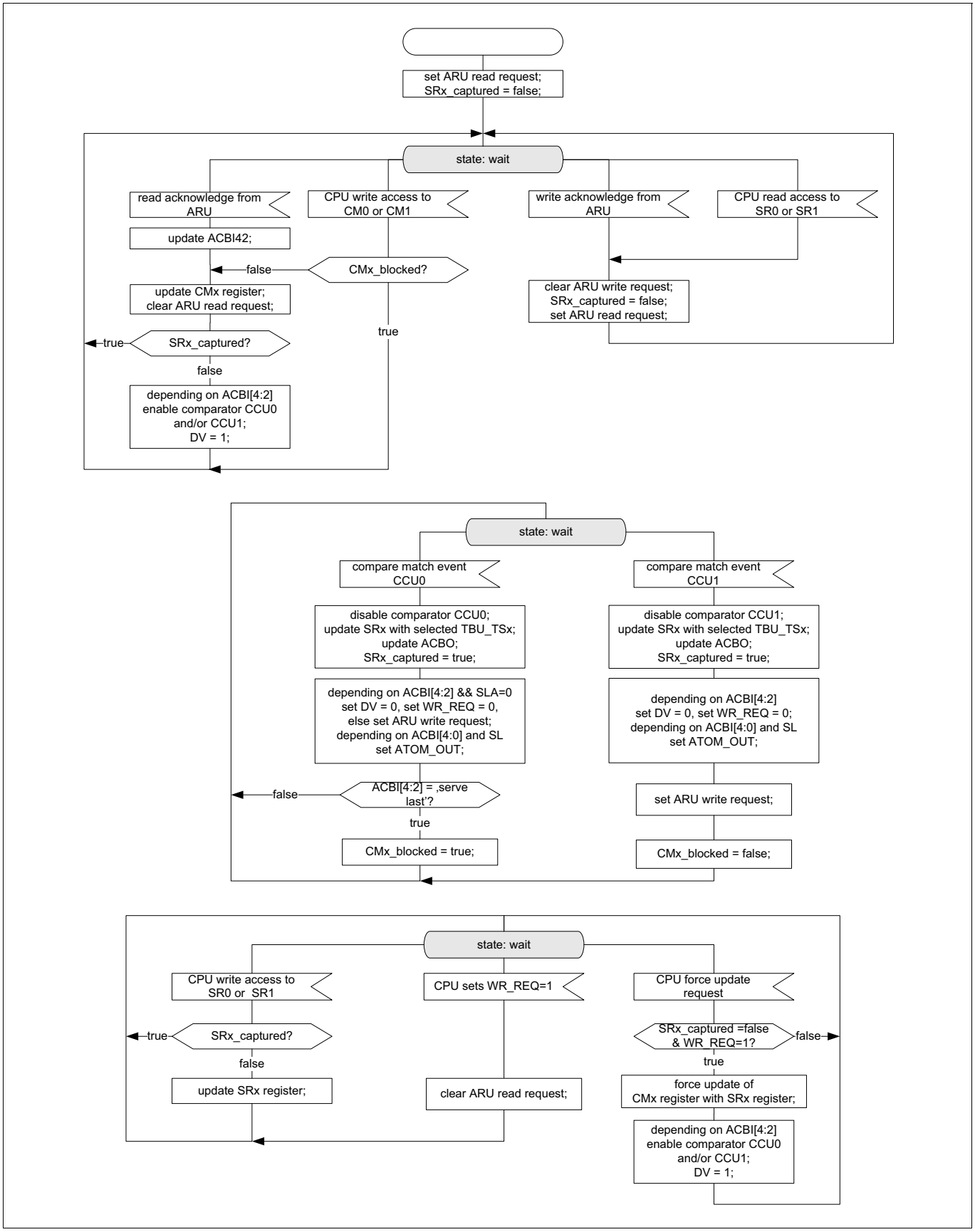


Figure 74 SOMC State diagram for SOMC mode, ARU enabled and ABM enabled

ATOM SOMC Late update mechanism

Although, the ATOM channel may be controlled by data received via the ARU, the CPU is able to request at any time a late update of the compare register. This can be initiated by setting the **WR_REQ** bit inside the **ATOM[i]_CH[x]_CTRL** register. By doing this, the ATOM will request no further data from ARU (if ARU access was enabled). The channel will in any case continue to compare against the values stored inside the compare registers (if bit **DV** was set). The CPU can now update the new compare values until the compare event happens by writing to the shadow registers, and force the ATOM channel to update the compare registers by writing to the force update register bits in the **AGC** register.

If the **WR_REQ** bit is set and a compare match event happens, any further access to the shadow registers **SR0**, **SR1** is blocked and the force update of this channel is blocked. In addition, the **WRF** bit is set in the **ATOM[i]_CH[x]_STAT** register. Thus, the CPU can determine that the late update failed by reading the **WRF** bit.

In case of bit **EUPM**=0 (register **ATOM[i]_CH[x]_CTRL**) the following statements are true:

If a compare match event already happened, the **WR_REQ** bit could not be set until the channel is unlocked for a new compare match event by reading the shadow registers. In addition, the **WRF** bit is set if the CPU tries to write the **WR_REQ** bit in that case.

In case of bit **EUPM**=1 (register **ATOM[i]_CH[x]_CTRL**) the following statements are true:

If in case of serve last strategy a CCU1 or in any other compare strategy a CCU0 or CCU1 compare match event already happened, the **WR_REQ** bit could not be set until the channel is unlocked for a new compare match event by reading the shadow registers. In addition, the **WRF** bit is set if the CPU tries to write the **WR_REQ** bit in that case.

In general, for a late update the following has to be taken into account:

If between a correct **WR_REQ** bit set, a correct shadow register write, and before the force update is requested by the AGC a match event occurs on the old compare values, the **WRF** bit will be set. The force update will be blocked.

The **WRF** bit will be set in any case if the CPU tries to write to a blocked shadow register.

The **WR_REQ** bit and the **DV** bit will be reset on a compare match event.

After a capture event for register **SR0** and/or **SR1** the force update mechanism will be blocked until a read access to the register **SR0** or **SR1** by either the ARU or the CPU happens.Writing to **SR0** or **SR1** after compare match causes an AEI write status 0b10.

The ATOM SOMC late update mechanism from CPU is shown in **Figure 75**.

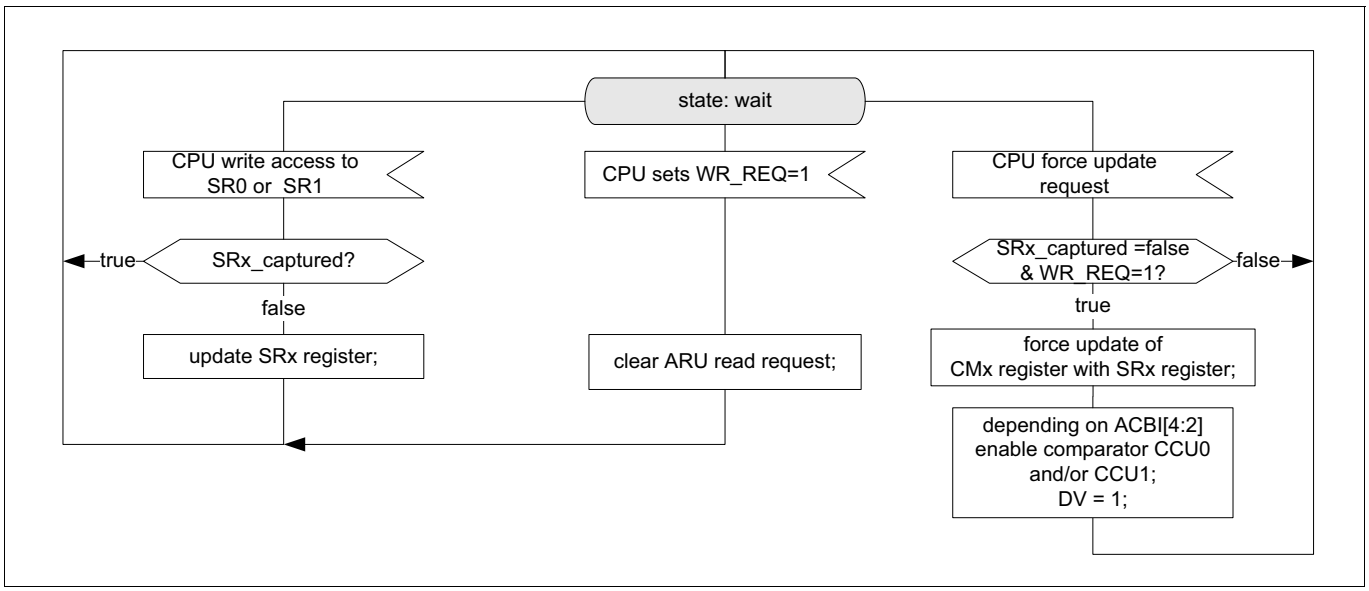


Figure 75 SOMC State diagram for late update requests by CPU

28.15.3.2.4 Register ATOM[i]_CH[x]_CTRL in SOMC mode

Register ATOM[i]_CH[x]_CTRL in SOMC mode

GTM_ATOMi_CHx_SOMC (i=0-11; x=0-7)
ATOMi Channel x Control Register in SOMC Mode ($E8004_H$ + i*800_H+ x*80_H) **Reset Value: 00000800_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREEZE	Not_used	Not_used	Reserved	ABM	Not_used	SLA	TRIGOUT	EXTTRIGOUT	Not_used		Not_used	Not_used		Not_used	WR_REQ
rw	rw	rw	r	rw	rw	rw	rw	rw	r		rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not_used	Not_used			SL	EUPM	CMP_CTRL	ACB42			ACB10		ARU_EN	TB12_SEL	MODE	
rw	rw			rw	rw	rw	rw			rw		rw	rw	rw	

Field	Bits	Type	Description
MODE	1:0	rw	**ATOM channel mode select** 01_B ATOM Signal Output Mode Compare (SOMC)
TB12_SEL	2	rw	**Select time base value TBU_TS1 or TBU_TS2** 0_B T*BU_TS1* selected for comparison 1_B T*BU_TS2* selected for comparison Note: This bit is only applicable if three time bases are present in the GTM. Otherwise, this bit is reserved.
ARU_EN	3	rw	**ARU Input stream enable** 0_B ARU Input stream disabled 1_B ARU Input stream enabled
ACB10	5:4	rw	**Signal level control bits** 00_B No signal level change at output (exception in **Figure 66** and **Figure 67** mode ACB42=001). 01_B Set output signal level to 1 when SL bit = 0 else output signal level to 0. 10_B Set output signal level to 0 when SL bit = 0 else output signal level to 1. 11_B Toggle output signal level (exception in **Figure 66** and **Figure 67** mode ACB42=001). These bits are only applicable if ARU_EN = '0'.

Field	Bits	Type	Description
ACB42	8:6	rw	**ATOM control bits ACB(4), ACB(3), ACB(2)** 000_B Compare in CCU0 and CCU1 in parallel, disable the CCUx on a compare match on either of compare units. Use *TBU_TS0* in CCU0 and *TBU_TS1* or *TBU_TS2* in CCU1. 001_B Compare in CCU0 and CCU1 in parallel, disable the CCUx on a compare match on either compare units. Use *TBU_TS0* in CCU0 and *TBU_TS1* or *TBU_TS2* in CCU1. 010_B Compare in CCU0 only against *TBU_TS0*. 011_B Compare in CCU1 only against *TBU_TS1* or *TBU_TS2*. 100_B Compare first in CCU0 and then in CCU1. Use *TBU_TS0*. 101_B Compare first in CCU0 and then in CCU1. Use *TBU_TS1* or *TBU_TS2*. 110_B Compare first in CCU0 and then in CCU1. Use *TBU_TS0* in CCU0 and *TBU_TS1* or *TBU_TS2* in CCU1. 111_B Cancel pending compare events. Note: These bits are defining the compare strategy only if ARU_EN = 0. Independent of ARU_EN, a writing of 0b111 cancels any pending CCU0 or CCU1 compare.
CMP_CTRL	9	rw	**CCUx compare strategy select** 0_B Greater-equal compare against TBU time base values (TBU_TS1/2 >;= CM0/1) 1_B Less-equal compare against TBU time base values (TBU_TS1/2 <= CM0/1) The compare unit CCU0 or CCU1 that compares against TBU_TS0 (depending on CCUx control mode defined by ACBI(4:2) or ACB42) always performs a greater-equal comparison, independent on CMP_CTRL bit.
EUPM	10	rw	**Extended Update Mode** 0_B No extended update of CM0 and CM1 via CPU or ARU; 1_B Extended update mode: in case of compare strategy ‘serve last’: update of CM1 after CCU0 compare match possible, via ARU or CPU. Note: If EUPM=1, write access to CM0 never causes an AEI write status 0b10. This bit is only applicable in SOMC and SOMB mode.
SL	11	rw	**Initial signal level after channel enable** 0_B Low signal level 1_B High signal level Note: Reset value depends on the hardware configuration chosen by silicon vendor. Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL. If FREEZE=0, following note is valid: If the channel is disabled, the output register of SOU unit is set to value of SL. If FREEZE=1, following note is valid: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.
Not_used	14:12	rw	**Not used** Note: Not used in this mode.

Field	Bits	Type	Description
Not_used	15	rw	**Not used** Note: Not used in this mode.
WR_REQ	16	rw	**CPU write request bit** 0_B No late update requested by CPU 1_B Late update requested by CPU Note: The CPU can disable subsequent ARU read requests by the channel and can update the shadow registers with new compare values, while the compare units operate on old compare values received by former ARU accesses, if occurred. Note: On a compare match event, the WR_REQ bit will be reset by hardware. Note: At the point of the force update only the shadow registers SR0 and SR1 are transferred into the CM0, CM1 registers. The output action is still defined by the ACBI bit field described by the ARU together with the old compare values for CM0/CM1.
Not_used	17	rw	**Not used** Note: Not used in this mode.
Not_used	19:18	rw	**Not used** Note: Not used in this mode.
Not_used	20	rw	**Not used** Note: Not used in this mode.
Not_used	22:21	r	**Not used** Note: Not used in this mode.
EXTTRIGOUT	23	rw	**Select TIM_EXT_CAPTURE(x) as potential output signal TRIG_[x]** 0_B signal *TRIG_[x-1]* is selected as output on *TRIG_[x]* (if TRIGOUT=0) 1_B signal *TIM_EXT_CAPTURE(x)* is selected as output on *TRIG_[x]* (if TRIGOUT=0)
TRIGOUT	24	rw	**TRIGOUT: Trigger output selection (output signal TRIG_CHx) of module ATOM_CHx** 0_B *TRIG_[x]* is *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)* 1_B *TRIG_[x]* is *TRIG_CCU0*

Field	Bits	Type	Description
SLA	25	rw	**'Serve last' ARU communication strategy** 0_B Capture SRx time stamps after CCU0 match event not provided to ARU 1_B Capture SRx time stamps after CCU0 match event provided to ARU Note: Please note, that setting of this bit has only effect, when ACBI(4:2) is configured for serve last compare strategy ("100", "101", or "110"). Note: When this bit is not set, the captured time stamps in the shadow registers SRx are only provided after the CCU1 match occurred. The ACBO(4:3) bits always return "10" in that case. Note: By setting this bit, the ATOM channel also provides the captured time stamps after the CCU0 match event to the ARU. The ACBO(4:3) bits are set to "01" in that case. After the CCU1 match event, the time stamps are captured again in the SRx registers and provided to the ARU. The ACBO(4:3) bits are set to "10". When the data in the shadow registers after the CCU0 match was not consumed by an ARU destination and the CCU1 match occurs, the data in the shadow registers is overwritten by the new captured time stamps. The ATOM channel does not request new data from the ARU when the CCU0 match values are read from an ARU destination.
Not_used	26	rw	**Not used** Note: Not used in this mode.
ABM	27	rw	**ARU blocking mode** 0_B ARU blocking mode disabled: ATOM reads continuously from ARU and updates CM0, CM1 and ACB bits independent of pending compare match event. 1_B ARU blocking mode enabled: after update of CM0, CM1 and ACB bit via ARU, no new data is read via ARU until compare match event occurred and SR0 and/or SR1 are read.
Reserved	28	r	**Reserved** Read as zero, should be written as zero.
Not_used	29	rw	**Not used** Note: Not used in this mode.
Not_used	30	rw	**Not used** Note: Not used in this mode.
FREEZE	31	rw	**FREEZE** 0_B a channel disable/enable may change internal register and output register 1_B a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

28.15.3.3 ATOM Signal Output Mode PWM (SOMP)

In ATOM Signal Output Mode PWM (SOMP) the ATOM sub-module channel is able to generate complex PWM signals with different duty cycles and periods. Duty cycles and periods can be changed synchronously and asynchronously. Synchronous change of the duty cycle and/or period means that the duty cycle or period

duration changes after the end of the preceding period. An asynchronous change of period and/or duty cycle means that the duration changes during the actual running PWM period.

The signal level of the pulse generated inside the period can be configured inside the channel control register (**SL** bit of **ATOM[i]_CH[x]_CTRL** register). The initial signal output level for the channel is the inverse pulse level defined by the **SL** bit. **Figure 76** depicts this behavior.

The counter **CN0** of each channel can run in two different modes depending on configuration of **UDMODE** in register **ATOM[i]_CH[x]_CTRL.**By default the counter counts only up until it reaches **CM0** and is then reset to 0.In the up down counter mode **CN0** switches between counting up and counting down.

28.15.3.3.1 Continuous Counting Up Mode

In SOMP mode with **UDMODE**=0b00 (i.e. **CN0** counts only up), depending on configuration bits RST_CCU0 of register **ATOM[i]_CH[x]_CTRL** the counter register **CN0** can be reset either when the counter value is equal to the compare value **CM0** (i.e. CN0 counts only from 0 to CM0-1 and is then reset to 0) or when signaled by the ATOM[i] trigger signal *TRIG_[x-1]* of the preceding channel [x-1] (which can also be the last channel of preceding instance TOM[i-1]) or the trigger signal *TIM_EXT_CAPTURE*(x) of the assigned TIM channel [x].

In this case, if **UPEN_CTRL[x]**=1, also the working register **CM0,CM1** and **CLK_SRC** are updated.

Note: As an exception, the input TRIG_[0] of instance ATOM0 is triggered by its own last channel cCATO via signal TRIG_[cCATO].

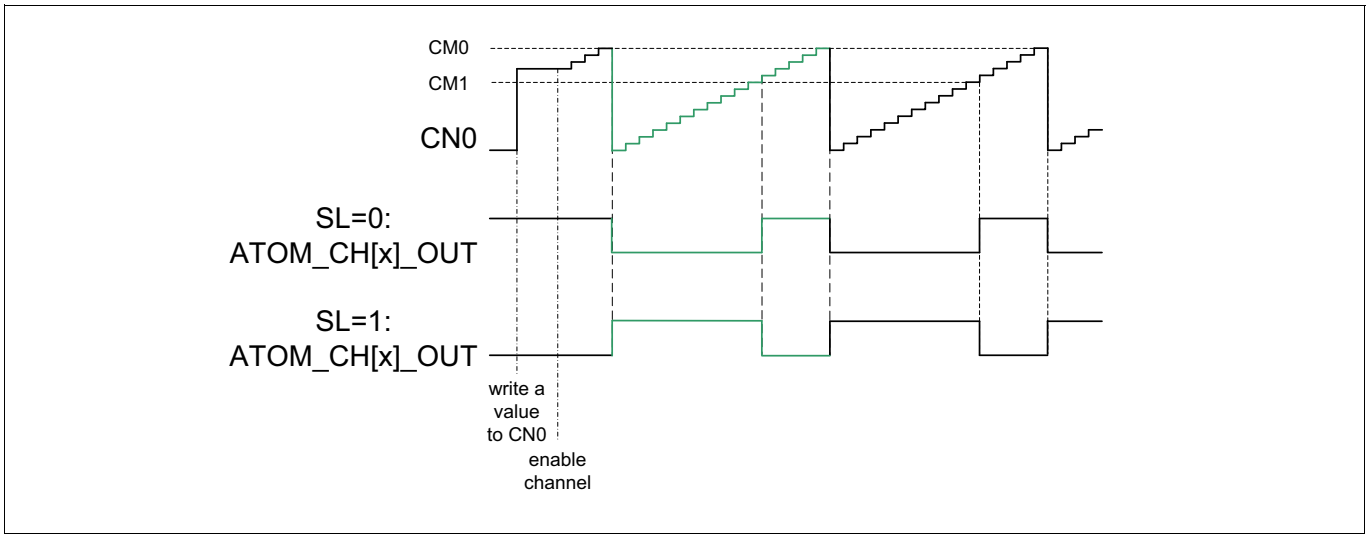


Figure 76 PWM Output behavior with respect to the SL bit in the ATOM[i]_CH[x]_CTRL register if UDMODE = 0b00

On an asynchronous update, it is guaranteed, that no spike occurs at the output port of the channel due to a too late update of the operation registers. The behavior of the output signal due to the different possibilities of an asynchronous update during a PWM period is shown in **Figure 77**.

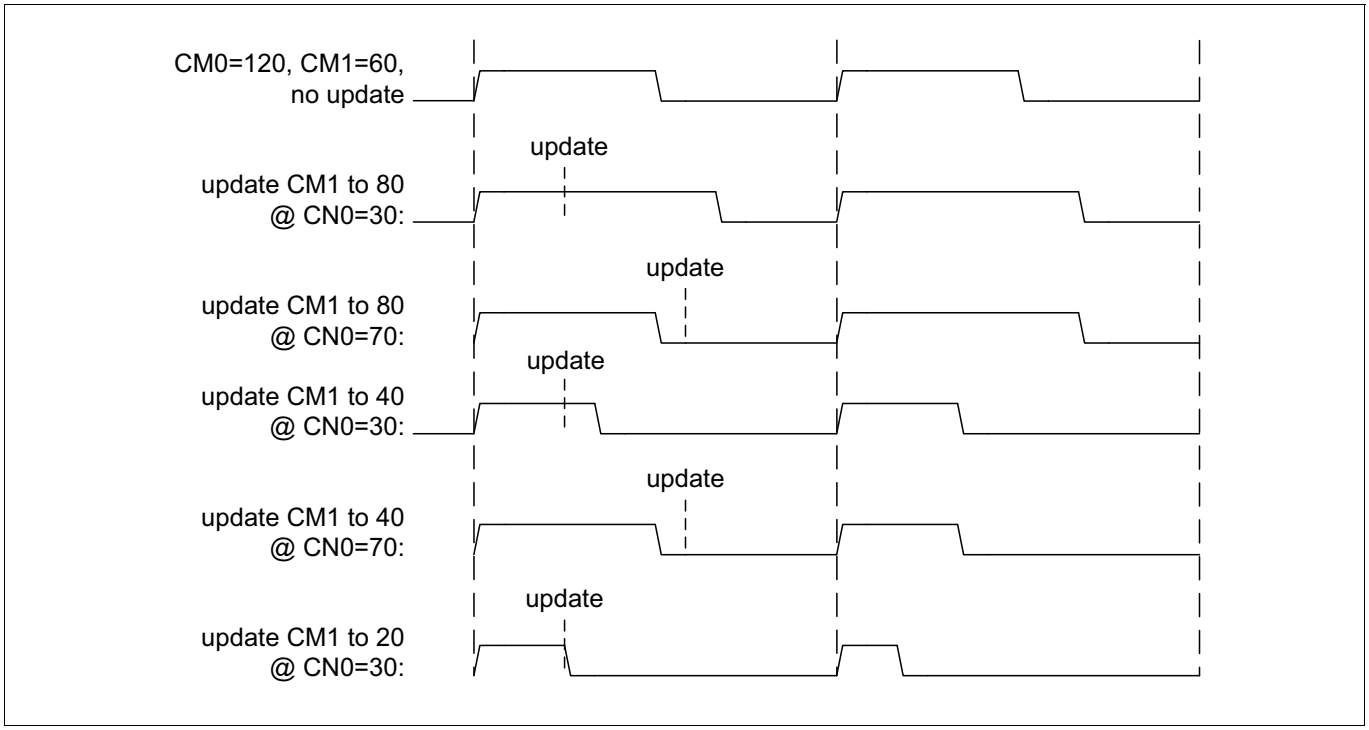


Figure 77 PWM Output behavior in case of an asynchronous update of the duty cycle

The duration of the pulse high or low time and period is measured with the counter in sub-unit CCU0. The trigger of the counter is one of the eight CMU clock signals configurable in the channel control register **ATOM[i]_CH[x]_CTRL**. The register **CM0** holds the duration of the period and the register **CM1** holds the duration of the duty cycle in clock ticks of the selected CMU clock.

If counter register **CN0** of channel x is reset by its own CCU0 unit (i.e. the compare match of **CN0** >=**CM0-1** configured by **RST_CCU0**=0), following statements are valid:

- **CN0** counts from 0 to **CM0**-1 and is then reset to 0
- When CN0 is reset from CM0 to 0, an edge to SL is generated.
- When CN0 is incrementing and reaches CN0 > CM1, an edge to !SL is generated.
- if **CM0**=0 or **CM0**=1, the counter **CN0** is constant 0.
- if **CM1**=0, the output is !SL = 0% duty cycle
- if **CM1** >= **CM0** and **CM0**>1, the output is **SL** = 100% duty cycle

If the counter register **CN0** of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by **RST_CCU0**=1), following statements are valid:

- **CN0** counts from 0 to MAX-1 and is then reset to 0 by trigger signal
- **CM0** defines the edge to **SL** value, **CM1** defines the edge to !**SL** value.
- if **CM0**=**CM1**, the output switches to **SL** if **CN0**=**CM0**=**CM1** (**CM0** has higher priority)
- if **CM0**=0 and **CM1**=MAX, the output is **SL** = 100% duty cycle
- if **CM0** > MAX, the output is !**SL** = 0% duty cycle, independent of **CM1**.

In case the counter value **CN0** reaches the compare value in register **CM0** (in fact CM0-1) or the channel receives an external update trigger via the *FUPD(x)* signal, a synchronous update is performed. A synchronous update means that the registers **CM0** and **CM1** are updated with the content of the shadow registers **SR0** and **SR1** and the **CLK_SRC** register is updated with the value of the **CLK_SRC_SR** register.

The clock source for the counter can be changed synchronously at the end of a period. If ARU access is disabled, this is done by using the bit field **CLK_SRC_SR** of register **ATOM[i]_CH[x]_CTRL** as shadow registers for the next CMU clock source.

28.15.3.3.2 Continuous Counting Up-Down Mode

In SOMP mode, if **CN0** counts up and down (**UDMODE** != 0b00), depending on configuration bit **RST_CCU0** of register **ATOM[i]_CH[x]_CTRL** the counter register **CN0** changes the direction either when the counter value is equal to the compare value **CM0** (in fact CM0-1), has counted down to 0 or when triggered by the ATOM[i] trigger signal *TRIG_[x-1]* of the preceding channel [x-1] (which can also be the last channel of preceding instance ATOM[i-1]) or the trigger signal *TIM_EXT_CAPTURE(x)* of the assigned TIM channel [x].

In this case, if **UPEN_CTRL[x]**=1, also the working register **CM0**, **CM1** and **CLK_SRC** are updated depending on **UDMODE**.

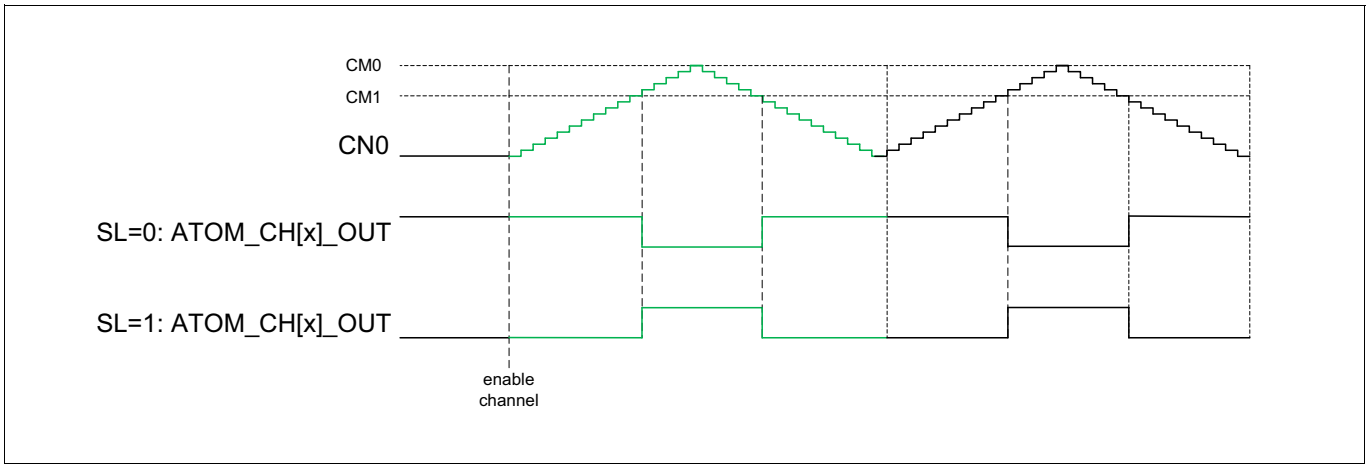


Figure 78 PWM Output behavior with respect to the SL bit in the ATOM[i]_CH[x]_CTRL register if UDMODE != 0b00

The clock of the counter register **CN0** can be one of the CMU clocks CMU_CLKx. If **ARU_EN**=0, the clock for **CN0** is defined by **CLK_SRC_SR** value in register **ATOM[i]_CH[x]_CTRL**.

If **ARU_EN**=1, the clock for **CN0** is defined by **CLK_SRC** value received via ARU.The duration of a period in multiples of selected **CN0** counter clock ticks is defined by the **CM0** configuration value (i.e. **CM0** defines half of period in up-down mode).

CM1 defines the duty cycle value in clock ticks of selected **CN0** counter clock (i.e. **CM0** defines half of duty cycle in up-down mode).

If counter register **CN0** of channel x is reset by its own CCU0 unit (i.e. the compare match of **CN0**>=**CM0-1** configured by **RST_CCU0**=0), following statements are valid:

- **CN0** counts continuously first up from 0 to **CM0**-1 and then down to 0
- if **CN0** >= **CM1**, the output is set to **SL**
- if **CM1**=0, the output is **SL** (i.e. 100% duty cycle)
- if **CM1** >= **CM0**, the output is !**SL** (i.e. 0% duty cycle)
- On output *ATOM[i]_CHx]_OUT* a PWM signal is generated. The period is defined by **CM0**, the duty cycle is defined by **CM1**.

This behavior is depicted in **Figure 78**.

If the counter register **CN0** of channel x is reset by the trigger signal coming from another channel or the assigned TIM module (configured by **RST_CCU0**=1), following statements are valid:

- **CN0** counts continuously first up. On a trigger signal the counter switches to count down mode. If **CN0** has reached 0, it counts up again.
- if **CN0** >= **CM1**, the output is set to **SL**
- if **CM1**=0, the output is **SL** (i.e. 100% duty cycle)
- if **CM1** >= **CM0**, the output is !**SL** (i.e. 0% duty cycle)
- On output *ATOM[i]_CHx]_OUT* a PWM signal is generated. The period is defined by the CCU0 trigger of triggering channel, the duty cycle is defined by **CM1**.
- On output *ATOM[i]_CHx]_OUT_T* a PWM signal is generated. The period is defined by the CCU0 trigger of triggering channel, the duty cycle is defined by **CM0**.

This behavior is depicted in **Figure 79**.

Note that in case of up-down counter mode and RST_CCU0=1 it is recommended that:

- The triggering channel and the triggered channel are both running in up-down mode.
- The time between two triggers signals is equal to the time needed for CN0 of triggered channel to count back to 0 and again up to the same upper value.

The second recommendation can be reached by synchronizing the start of triggering channel and of the triggered channel, i.e. let both channel start with a CN0 value 0.

Note that if there is a synchronization register in the trigger chain (indicated by value ATOM_TRIG_CHAIN in register CCM[i]_HW_CONF), the additional delay of the trigger by one clock period has to be taken into account by starting at triggering channel with a CN0 value 1 (+1 compared to CN0 of triggered channel).

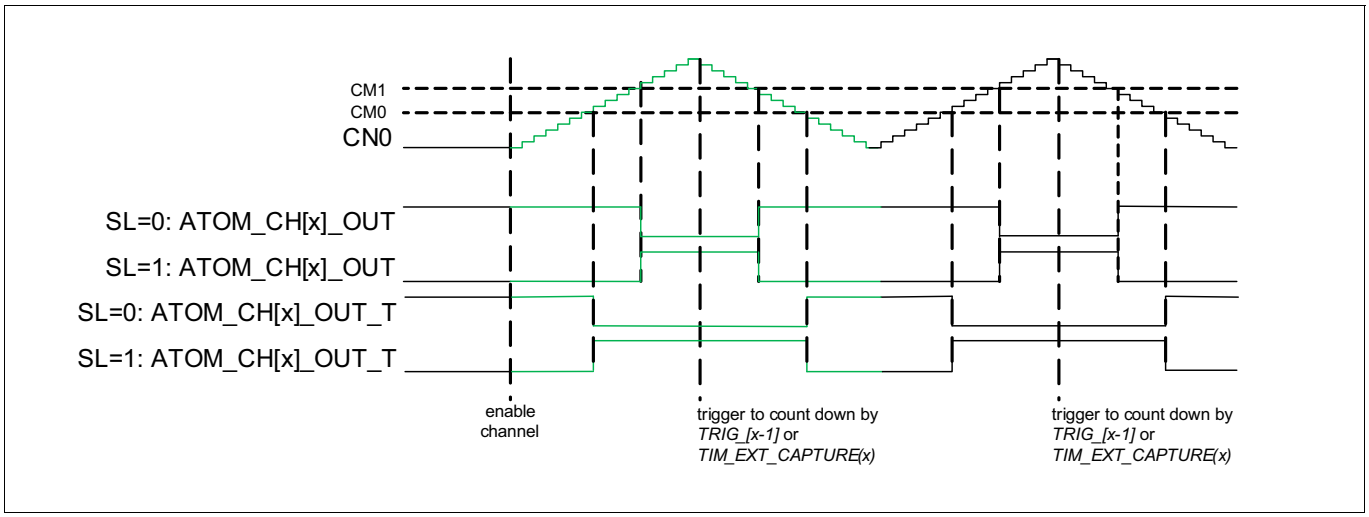


Figure 79 PWM Output behavior in case of RST_CCU0=1

28.15.3.3.3 ARU controlled update

If ARU access is enabled, the bits **ACBI(4)**, **ACBI(3)** and **ACBI(2)** received via ARU and stored in register **ATOM_[i]_CH[x]_STAT** are used as shadow register for the update of the CMU clock source register **CLK_SRC**.

For the synchronous update mechanism the generation of a complex PWM output waveform is possible without CPU interaction by reloading the shadow registers **SR0**, **SR1** and the **ACBI** bit field over the ACI sub-unit from the ARU, while the ATOM channel operates on the **CM0** and **CM1** registers.

This internal update mechanism is established, when the old PWM period ends. The shadow registers are loaded into the operation registers, the counter register is reset, the new clock source according to the **CLK_SRC_SR** and **ACBI(4)**, **ACBI(3)** and **ACBI(2)** bits is selected and the new PWM generation starts.

In parallel, the ATOM channel issues a read request to the ARU to reload the shadow registers with new values while the ATOM channel operates on the operation registers. To guarantee the reloading, the PWM period must not be smaller than the worst case ARU round trip time and source for the PWM characteristic must provide the new data within this time. Otherwise, the old PWM values are used from the shadow registers.

When updated over the ARU the user has to ensure that the new period duration is located in the lower (bits 23 to 0) and the duty cycle duration is located in the upper (bits 47 to 24) ARU data word and the new clock source is specified in the ARU control bits 52 to 50.

This pipelined data stream character is shown in **Figure 80**.

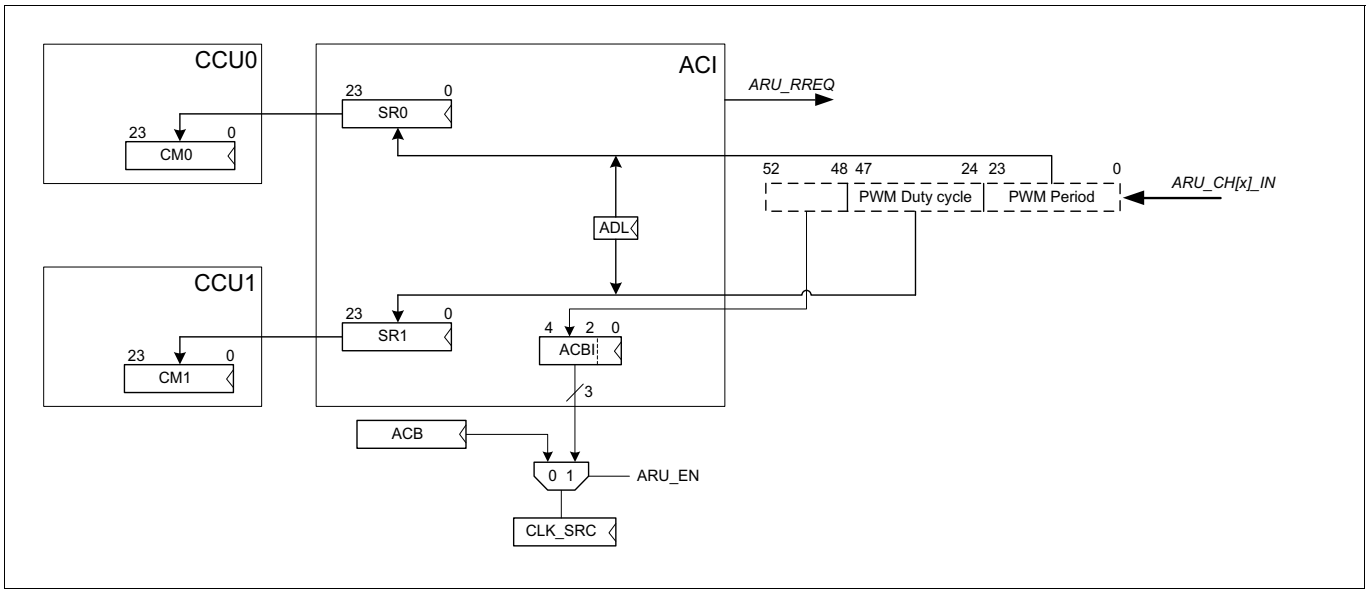


Figure 80 ARU Data input stream pipeline structure for SOMP mode

When an ARU transfer is in progress which means the *ARU_RREQ* is served by the ARU, the ACI locks the update mechanism of **CM0**, **CM1** and **CLK_SRC** until the read request has finished. The CCU0 and CCU1 operate on the old values when the update mechanism is locked.

28.15.3.3.4 CPU controlled update

The shadow registers **SR0** and **SR1** can also be updated over the AEI bus interface. In this case, **ARU_EN** hat to be set to 0.

When updated via the AEI bus the **CM0** and **CM1** update mechanism has to be locked via the **AGC_GLB_CTRL** register with the *UPENx* signal in the AGC sub-unit. To select the new clock source in this case, the CPU has to write to the CLK_SRC_SR bit field of the **ATOM[i]_CH[x]_CTRL** register.

For an asynchronous update of the duty cycle and/or period the new values must be written directly into the compare registers **CM0** and/or **CM1** while the counter **CN0** continues counting. This update can be done only via the AEI bus interface immediately by the CPU or by the *FUPD(x)* trigger signal triggered from the AGC global trigger logic. Values received through the ARU interface are never loaded asynchronously into the operation registers **CM0** and **CM1**. Therefore, the ATOM channel can generate a PWM signal on the output port pin *ATOM[i]_CH[x]_OUT* on behalf of the content of the **CM0** and **CM1** registers, while it receives new PWM values via the ARU interface ACI in its shadow registers.

On a compare match of **CN0** and **CM0** or **CM1** the output signal level of *ATOM[i]_CH[x]_OUT* is toggled according to the signal level output bit **SL** in the **ATOM[i]_CH[x]_CTRL** register.

Thus, the duty cycle output level can be changed during runtime by writing the new duty cycle level into the **SL** bit of the channel configuration register. The new signal level becomes active for the next trigger *CCU_TRIGx* (since bit **SL** is written).

Since the *ATOM[i]_CH[x]_OUT* signal level is defined as the reverse duty cycle output level when the ATOM channel is enabled, a PWM period can be shifted earlier by writing an initial offset value to **CN0** register. By doing this, the ATOM channel first counts until **CN0** reaches **CM0** and then it toggles the output signal at *ATOM[i]_CH[x]_OUT*.

28.15.3.3.5 One-shot Counting Up Mode

The ATOM channel can operate in One-shot mode when the **OSM** bit is set in the channel control register. One-shot mode means that a single pulse with the pulse level defined in bit **SL** is generated on the output line.

First the channel has to be enabled by setting the corresponding **ENDIS_STAT** value.

In one-shot mode the counter **CN0** will not be incremented once the channel is enabled.

A write access to the register **CN0** triggers the start of pulse generation (i.e. the increment of the counter register **CN0**).

If the counter **CN0** is reset from **CM0-1** back to zero, the first edge at *ATOM[i]_CH[x]_OUT* is generated.

To avoid an update of **CMx** register with content of **SRx** register at this point in time, the automatic update should be disabled by setting **UPEN_CTRL[x]** = 0b00 (in register **ATOM[i]_CH[x]_CTRL**)

The second edge is generated if **CN0** is greater or equal than **CM1** (i.e. **CN0** was incremented until it has reached **CM1** or **CN0** is greater than **CM1** after an update of **CM1**).

If the counter **CN0** has reached the value of **CM0-1** a second time, the counter stops.

The new value of **CN0** determines the start delay of the first edge. The delay time of the first edge is given by (**CM0-CN0**) multiplied with period defined by current value of **CLK_SRC**.

Figure 81 depicts the pulse generation in SOMP one-shot mode.

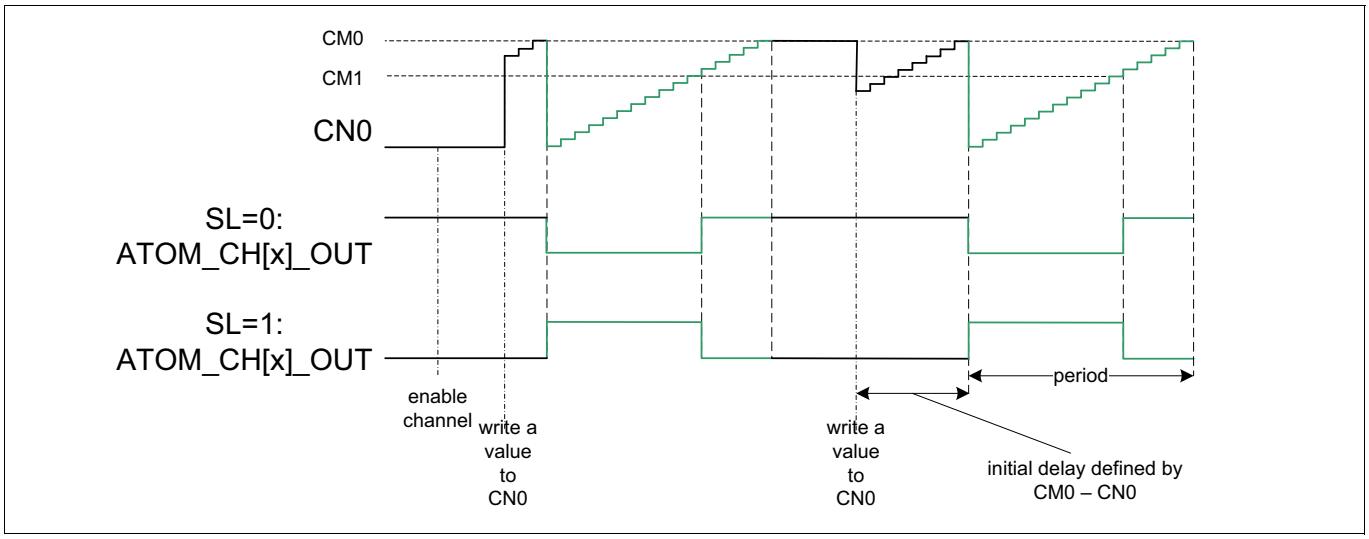


Figure 81 PWM Output with respect to configuration bit SL in One-shot counting up mode: trigger by writing to CN0

Further output of single pulses can be started by a write access to register **CN0**.

If **CN0** is already incrementing (i.e. started by writing to **CN0** a value CN0start < **CM0**), the effect of a second write access to **CN0** depends on the phase of **CN0**:

- phase 1: update of **CN0** before **CN0** reaches first time **CM0** (in fact CM0-1)
- phase 2: update of **CN0** after **CN0** has reached first time **CM0** but is less than **CM1**

- phase 3: update of **CN0** after **CN0** has reached first time **CM0** (in fact CM0-1) and **CN0** is greater than or equal **CM1**

In phase 1: writing to counter **CN0** a value CN0new < **CM0** leads to a shift of first edge (generated if **CN0** is reset first time from **CM0-1 back to 0**) by the time **CM0**-CN0new.

In phase 2: writing to incrementing counter **CN0** a value CN0new < **CM1** while CN0old is below **CM1** leads to a lengthening of the pulse. The counter **CN0** stops if it reaches **CM0**.

In phase 3: Writing to incrementing counter **CN0** a value CN0new while CN0old is already greater than or equal **CM1** leads to an immediate restart of a single pulse generation inclusive the initial delay defined by **CM0** - CN0new.

If a channel is configured to one-shot mode and configuration bit **OSM_TRIG** is set to 1, the trigger signal *OSM_TRIG* (i.e. *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*) triggers start of one pulse generation.

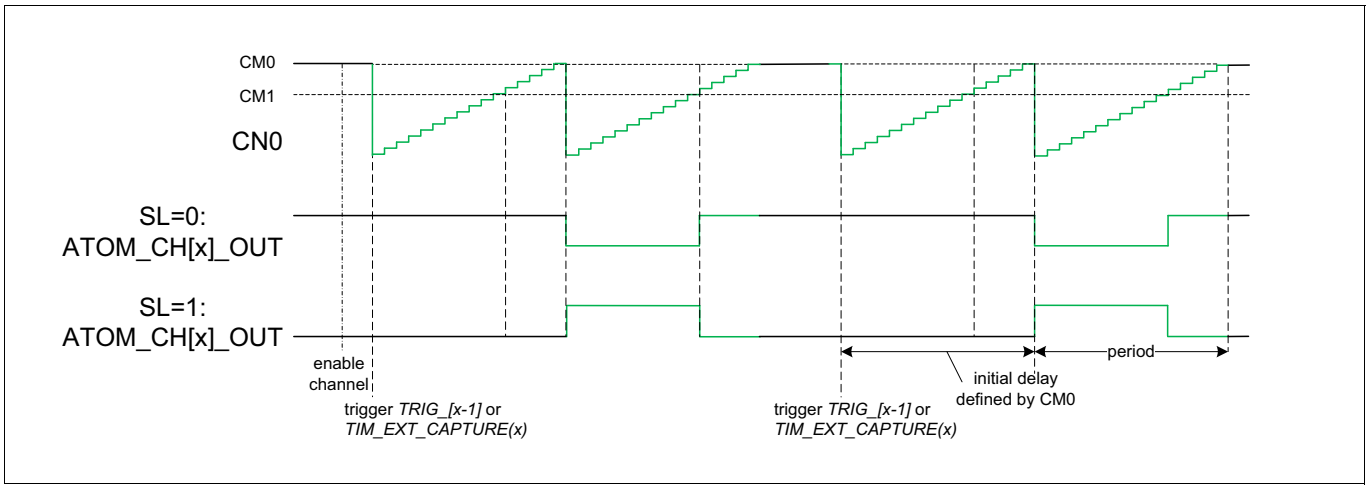


Figure 82 PWM Output with respect to configuration bit SL in one-shot mode: trigger by TRIG_[x-1] or TIM_EXT_CAPTURE(x)

28.15.3.3.6 One-shot Counting Up-Down Mode

The ATOM channel can operate in one-shot counting up-down mode when the bit **OSM** = 1 and the **UDMODE** != 0b00. One-shot mode means that a single pulse with the pulse level defined in bit **SL** is generated on the output line.

First the channel has to be enabled by setting the corresponding **ENDIS_STAT** value.

In one-shot mode the counter **CN0** will not be incremented once the channel is enabled.

A write access to the register **CN0** triggers the start of pulse generation (i.e. the increment of the counter register **CN0**).

To avoid an update of **CMx** register with content of **SRx** register at this point in time, the automatic update should be disabled by writing **UPEN_CTRL[x]** = 0b01 (see register **ATOM[i]_AGC_GLB_CTRL**)

If the counter **CN0** is greater or equal than **CM1**, the output *ATOM[i]_CH[x]_OUT* is set to **SL** value.

If the counter **CN0** is less than **CM1**, the output *ATOM[i]_CH[x]_OUT* is set to !**SL** value.

If the counter **CN0** has reached the value 0 (by counting down), it stops.

The new value of **CN0** determines the start delay of the first edge. The delay time of the first edge is given by (**CM1-CN0**) multiplied with period defined by current value of **CLK_SRC**.

Figure 83 depicts the pulse generation in SOMP one-shot mode.

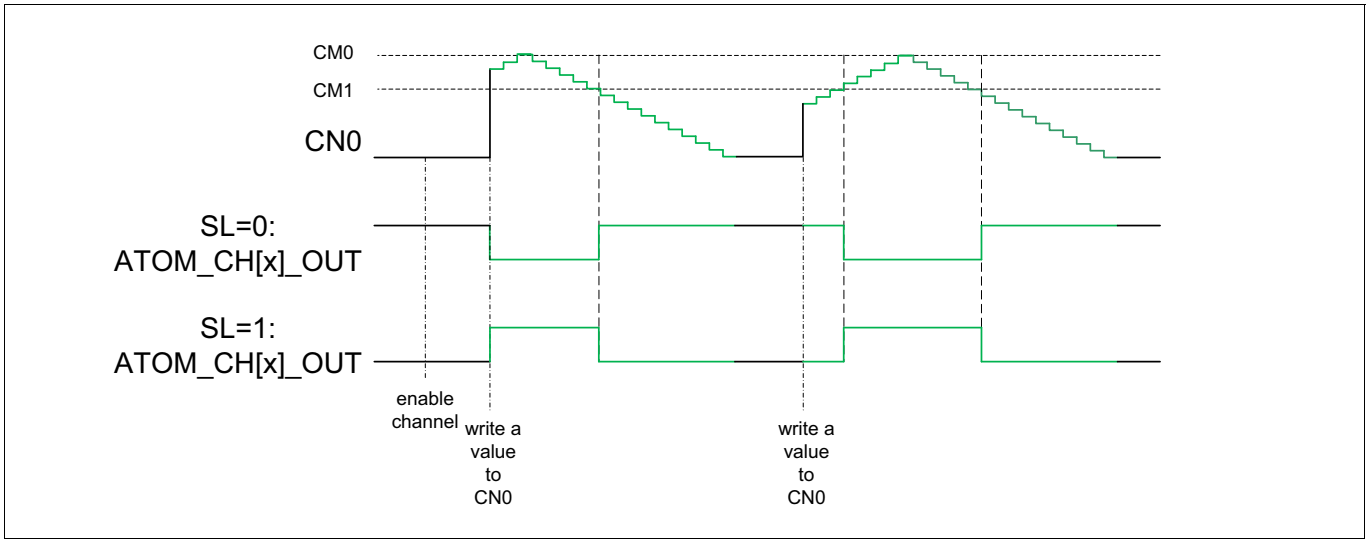


Figure 83 PWM Output with respect to configuration bit SL in one-shot counting up-down mode: trigger by writing to CN0

Further output of single pulses can be started by writing to register **CN0**.

If a channel is configured to one-shot counting up-down mode and configuration bit **OSM_TRIG** is set to 1, the trigger signal *OSM_TRIG* (i.e. *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*) triggers start of one pulse generation.

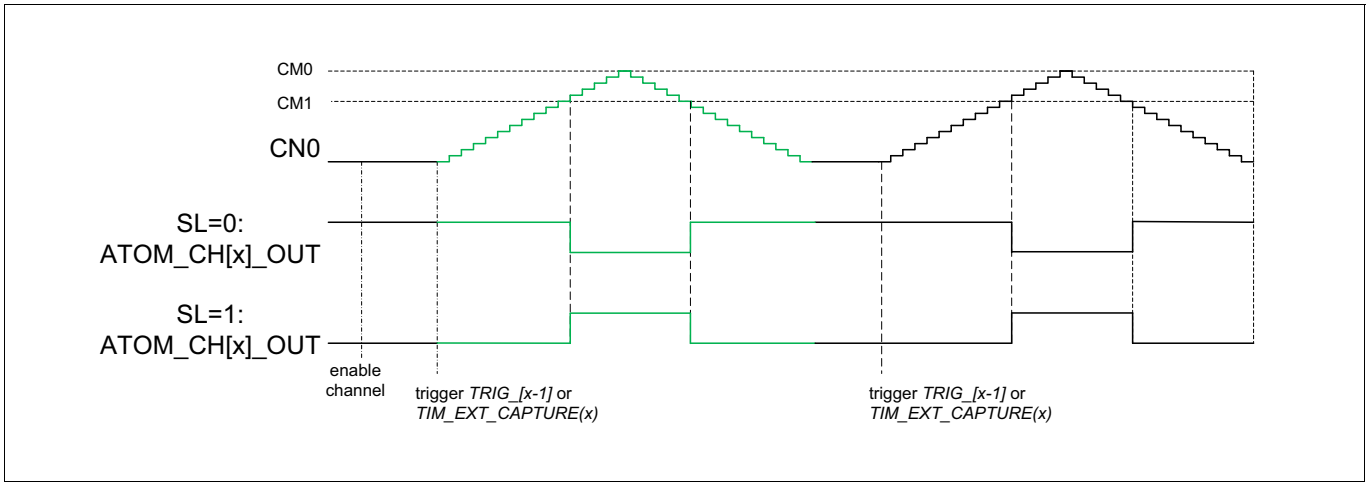


Figure 84 PWM Output with respect to configuration bit SL in one-shot counting up-down mode: trigger by TRIG_[x-1] or TIM_EXT_CAPTURE(x)

28.15.3.3.7 Pulse Count Modulation Mode

At the output *ATOM[i]_CH[x]_OUT* a pulse count modulated signal can be generated instead of the simple PWM output signal in SOMP mode.

The PCM mode is enabled by setting bit **BITREV** to 1 (bit 6 in **ATOM[i]_CH[x]_CTRL** register).Please note that it is device specific, in which channel the PCM mode is available. Please refer to device specific device specific appendix for this information.

With the configuration bit **BITREV**=1 a bit-reversing of the counter output **CN0** is configured. In this case the bits LSB and MSB are swapped, the bits LSB+1 and MSB-1 are swapped, the bits LSB+2 and MSB-2 are swapped and so on.

The effect of bit-reversing of the **CN0** register value is shown in the following **Figure 85**.

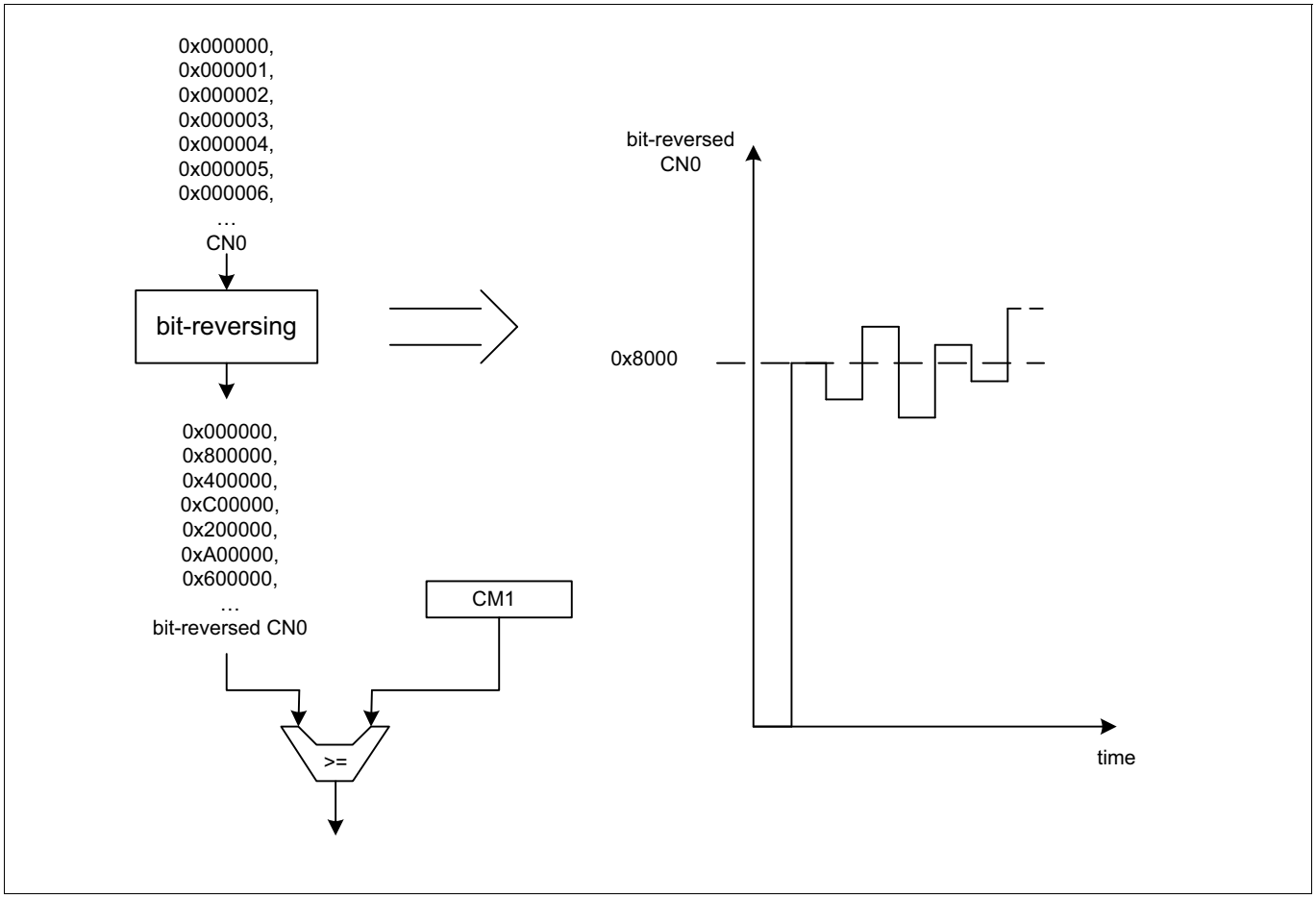


Figure 85 Bit reversing of counter CN0 output

In the PCM mode the counter register **CN0** is incremented by every clock tick depending on configured CMU clock (*CMU_CLK*).

The output of counter register **CN0** is first bit-reversed and then compared with the configured register value **CM1**.

If the bit-reversed value of register **CN0** is greater or equal than **CM1**, the SR-FlipFlop of sub-module SOU is set (i.e. set to inverse value of **SL**) otherwise the SR-FlipFlop is reset (i.e. to the value of **SL**). This generates at the output *ATOM[i]_CH[x]_OUT* a pulse count modulated signal.

In PCM mode the **CM0** register - in which the period is defined - normally has to be set to its maximum value 0xFFFFFF.

To reduce time period of updating duty cycle value in **CM1** register, it is additionally possible to setup period value in **CM0** register to smaller values than maximum value as described before.

Possible values for **CM0** register are each even numbered values to the power of 2 e.g. 0x800000, 0x400000, 0x200000….

In this case the duty cycle has to be configured in the following manner.

Depending on how much the period in **CM0** register is decreased - means shifted right starting from 0x1000000 - the duty cycle in **CM1** register has to be shifted left (= rotated: shift MSB back into LSB) with same value, e.g.:

period **CM0** = 0x001000 -> shifted 8 bits right from 0x1000000

--> so duty cycle has to be shifted left 8 bit:

e.g. 50% duty cycle = 0x0008000 -> shift 8 bits left -> **CM1** = 0x800000

More examples:

period CM0	-->	duty cycle	-->	no shift	-->	CM1
0xFFFFFF	-->	0x800000	-->	no shift	-->	0x800000
0x800000	-->	0x400000	-->	shift 1 bit left	-->	0x800000
0x400000	-->	0x100000	-->	shift 2 bits left	-->	0x400000
0x200000	-->	0x0FFFFF	-->	shift 3 bits left	-->	0x7FFFF8
0x100000	-->	0x033333	-->	shift 4 bits left	-->	0x333330
0x080000	-->	0x005555	-->	shift 5 bits left	-->	0x0AAAA0
…						
0x000020	-->	0x000008	-->	shift 19 bits left	-->	0x400000
0x000010	-->	0x000005	-->	shift 20 bits left	-->	0x500000
…						

In this mode the interrupt CCU1TC (see register **ATOM[i]_CH[x]_IRQ_NOTIFY**) is set every time if bit reverse value of **CN0** is greater or equal than **CM1** which may be multiple times during one period. Therefore, from application point of view it is not useful to enable this interrupt.

28.15.3.3.8 Trigger generation

For applications with constant PWM period defined by **CM0**, it is not necessary to update regularly the **CM0** register with **SR0** register. For these applications the **SR0** register can be used to define an additional output signal and interrupt trigger.

If bit **SR0_TRIG** in register **ATOM[i]_CH[x]_CTRL** is set, the register **SR0** is no longer used as a shadow register for register **CM0**. Instead, **SR0** is compared against **CN0** and if both are equal, a pulse of signal level 1 is generated at the output *ATOM[i]_CH[x]_OUT_T*.

The bit **SR0_TRIG** should only be set if bit RST_CCU0 of this channel is 0.

Note: If **ARU_EN**=1 and both **SR0** and **SR1** are updated via ARU, the new **SR0** value is used immediately after update. Update of **SR0** via ARU can be suppressed by ADL configuration in register **ATOM[i]_CH[x]_CTRL**.

If bit **SR0_TRIG** is set the interrupt notify flag **CCU1TC** is no longer set on a compare match of **CM1** and **CN0**. Instead, the **CCU1TC** interrupt notify flag is set in case of a compare equal match of **SR0** and **CN0**.

With configuration bit **TRIG_PULSE** one can select if the output *ATOM[i]_CH[x]_OUT_T* is high as long as **CN0**=**SR0** (**TRIG_PULSE**=0) or if there will be only one pulse of length one *SYS_CLK* period when **CN0** becomes **SR0** (**TRIG_PULSE**=1).

The ATOM output signal routing to DTM or GTM top level is described in subchapter “DTM connections on GTM-IP top level“.

28.15.3.3.9 Register ATOM[i]_CH[x]_CTRL in SOMP mode

Register ATOM[i]_CH[x]_CTRL in SOMP mode

GTM_ATOMi_CHx_SOMP (i=0-11; x=0-7)
ATOMi Channel x Control Register in SOMP Mode ($E8004_H$ + i*800_H+ x*80_H) **Reset Value: 00000800_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREEZE	Not_used	EXT_FUPD	Reserved	Not_used	OSM	Not_used	TRIGOUT	EXTTRIGOUT	EXT_TRIG	OSM_TRIG	RST_CCU0	UDMODE		TRIG_PULSE	Not_used
rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECLK_SRC	CLK_SRC_SR			SL	Not_used	Not_used	Not_used	SR0_TRIG	BITREV	ADL		ARU_EN	Not_used	MODE	
rw	rw			rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	

Field	Bits	Type	Description
MODE	1:0	rw	**ATOM channel mode select** 10_B ATOM Signal Output Mode PWM (SOMP)
Not_used	2	rw	**Not used** Note: Not used in this mode.
ARU_EN	3	rw	**ARU Input stream enable** 0_B ARU Input stream disabled 1_B ARU Input stream enabled
ADL	5:4	rw	**ARU data select for SOMP** 00_B Load both ARU words into shadow registers 01_B Load ARU low word (Bits 23…0) into shadow register SR0 10_B Load ARU high word (Bits 47…24) into shadow register SR1 11_B Reserved Note: This bit field is only relevant in SOMP mode to select the ARU data source.
BITREV	6	rw	**Bit-reversing of output of counter register CN0. This bit enables the PCM mode** It is device specific, in which channel the PCM mode is available. Please refer to device specific device specific appendix for this information.
SR0_TRIG	7	rw	**SR0 is used to generate a trigger on output ATOM[i]_CH[x]_OUT_T if equal to CN0** 0_B SR0 is used as a shadow register for register CM0. 1_B SR0 is not used as a shadow register for register CM0. SR0 is compared with CN0 and if both are equal, a trigger pulse is generated at output *ATOM[i]_CH[x]_OUT_T*. Note: This bit is only relevant in SOMP mode. Note: This bit should only be set if RST_CCU0 of this channel is 0.
Not_used	8	rw	**Not used** Note: Not used in this mode.

Field	Bits	Type	Description
Not_used	9	rw	**Not used** Note: Not used in this mode.
Not_used	10	rw	**Not used** Note: Not used in this mode.
SL	11	rw	**Signal level for pulse of PWM** 0_B Low signal level 1_B High signal level Note: Reset value depends on the hardware configuration chosen by silicon vendor. Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL FREEZE=0, the following note is valid: If the channel is disabled, the output register of SOU unit is set to inverse value of SL. If FREEZE=1, the following note is valid: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.
CLK_SRC_SR	14:12	rw	**Shadow register for CMU clock source register CLK_SRC** If ECLK_SRC=0 / ECLK_SRC=1: 000_B *CMU_CLK0* selected / *CMU_CLK0* selected 001_B *CMU_CLK1* selected / *CMU_CLK1* selected 010_B *CMU_CLK2* selected / *CMU_CLK2* selected 011_B *CMU_CLK3* selected / Reserved 100_B *CMU_CLK4* selected / clock stopped 101_B *CMU_CLK5* selected / *TRIG[x-1]* selected 110_B *CMU_CLK6* selected / *TIM_EXT_CAPTURE[x]* selected 111_B *CMU_CLK7* selected / *CMU_CLK7* selected Note: This register is a shadow register for the register CLK_SRC. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done either by an end of a period or a FORCE_UPDATE. Note: After (channel) reset the selected CLK_SRC value is the SYS_CLK (input of Global Clock Divider). To use in SOMP mode one of the CMU_CLKx, it is recommended to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel. Note: In case of ECLK_SRC=1 and CLK_SRC_SR = 0b11 / 0b100 / 0b101 / 0b110 a force update leads to an immediate update of CM0, CM1 and CLK_SRC.
ECLK_SRC	15	rw	**Extend CLK_SRC** 0_B CLK_SRC_SR set 1 selected 1_B CLK_SRC_SR set 2 selected See bit CLK_SRC_SR description for details.
Not_used	16	rw	**Not used** Note: Not used in this mode.

Field	Bits	Type	Description
TRIG_PULSE	17	rw	**Trigger output pulse length of one SYS_CLK period** 0_B output on TOM[i]_OUT[x]_T is 1 as long as CN0=SR0 (if SR=_TRIG=1) 1_B output on TOM[i]_OUT[x]_T is 1 for only one SYS_CLK period if CN0=SR0 (if SR=_TRIG=1)
UDMODE	19:18	rw	**Up/down counter mode** 00_B up/down counter mode disabled: CN0 counts always up 01_B up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches 0 (i.e. changes from down to up) 10_B up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches CM0 (i.e. changes from up to down) 11_B up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches 0 or CM0 (i.e. changes direction) Note: This mode is only applicable in SOMP mode.
RST_CCU0	20	rw	**Reset source of CCU0** 0_B Reset counter register CN0 to 0 on matching comparison with CM0 1_B Reset counter register CN0 to 0 on trigger *TRIG_[x-1]* or *TIM_EXT_CAPTURE(x)*. Note: If RST_CCU0=1 and UPEN_CTRLx=1 are set, *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)* triggers also the update of work register (CM0, CM1 and CLK_SRC).
OSM_TRIG	21	rw	**Enable trigger of one-shot pulse by trigger signal OSM_TRIG** 0_B signal OSM_TRIG cannot trigger start of single pulse generation 1_B signal OSM_TRIG can trigger start of single pulse generation (if bit OSM = 1) Note: This bit should only be set if bit OSM=1 and bit RST_CCU0=0.
EXT_TRIG	22	rw	**Select TIM_EXT_CAPTURE(x) as trigger signal** 0_B signal *TIM_[x-1]* is selected as trigger to reset CN0 or to start single pulse generation. 1_B signal *TIM_EXT_CAPTURE(x)* is selected
EXTTRIGOUT	23	rw	**Select TIM_EXT_CAPTURE(x) as potential output signal TRIG_[x]** 0_B signal *TRIG_[x-1]* is selected as output on *TRIG_[x]* (if TRIGOUT=0) 1_B signal *TIM_EXT_CAPTURE(x)* is selected as output on *TRIG_[x]* (if TRIGOUT=0)
TRIGOUT	24	rw	**Trigger output selection (output signal TRIG_CHx) of module ATOM_CHx** 0_B *TRIG_[x]* is *TRIG_[x-1*] or *TIM_EXT_CAPTURE(x)*. 1_B *TRIG_[x]* is *TRIG_CCU0*
Not_used	25	rw	**Not used** Note: Not used in this mode.
OSM	26	rw	**One-shot mode** 0_B Continuous PWM generation after channel enable 1_B A single pulse is generated

Field	Bits	Type	Description
Not_used	27	rw	**Not used** Note: Not used in this mode.
Reserved	28	r	**Reserved** Read as zero, should be written as zero.
EXT_FUPD	29	rw	**External forced update** 0_B use FUPD(x) signal from AGC to force update 1_B use TIM_EXT_CAPTURE signal to force update This bit is only applicable in SOMP and SOMS mode.
Not_used	30	rw	**Not used** Note: Not used in this mode.
FREEZE	31	rw	**FREEZE** 0_B a channel disable/enable may change internal register and output register 1_B a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode) *Note: If channel is disabled and output is enabled, in SOMP mode with UDMODE!=0b00 the output is depending directly on SL bit, independent on FREEZE mode.*

28.15.3.4 ATOM Signal Output Mode Serial (SOMS)

In ATOM Signal Output Mode Serial (SOMS) the ATOM channel acts as a serial output shift register where the content of the **CM1** register in the CCU1 unit is shifted out whenever the unit is triggered by the selected *CMU_CLK* input clock signal. The shift direction is configurable with the **ACB(0)** bit inside the **ATOM[i]_CH[x]_CTRL** register when ARU is disabled and the **ACBI(0)** bit inside the **ATOM[i]_CH[x]_STAT** register when ARU is enabled.

The data inside the **CM1** register has to be aligned according to the selected shift direction in the **ACB(0)**/**ACBI(0)** bit. This means that when a right shift is selected, that the data word has to be aligned to bit 0 of the **CM1** register and when a left shift is selected, that the data has to be aligned to bit 23 of the **CM1** register.

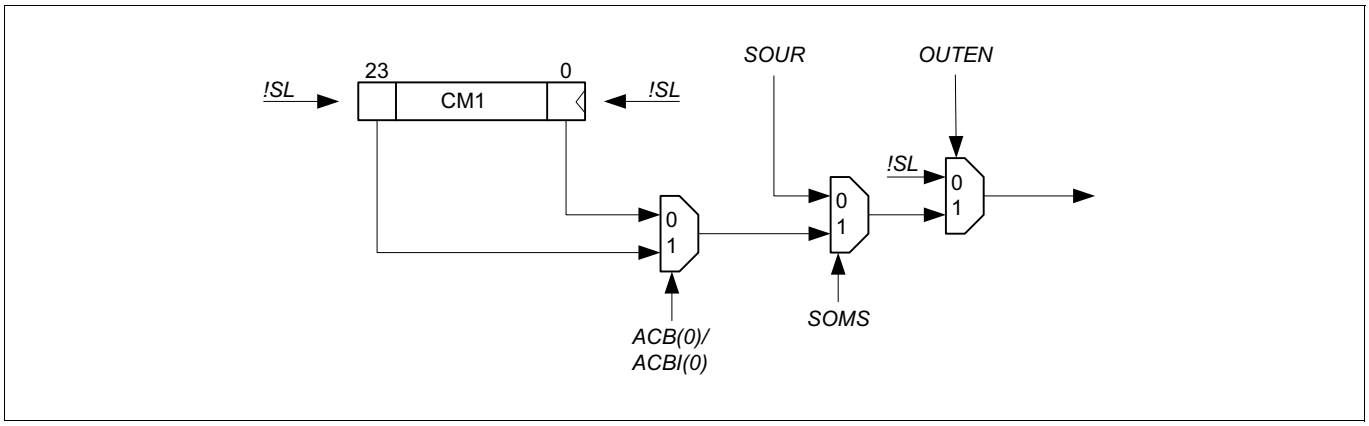


Figure 86 SOMS Mode output generation

Figure 86 shows the output generation in case of SOMS mode is selected.

In SOMS mode CCU0 runs in counter/compare mode and counts the number of bits shifted out so far. The total number of bits that should be shifted is defined as **CM0**. The total number of bits that are visible at *ATOM_OUT* is **CM0**+1.

When the output is disabled the *ATOM_OUT* is set to the inverse **SL** bit definition.

When the content of the **CM1** register is shifted out, the inverse signal level is shifted into the **CM1** register.

When the output is enabled while **UPEN_CTRL[x]** is disabled, the ATOM_OUT signal level is defined by **CM1** bit 0 or 23, dependent on the shift direction defined by **ACB(0)** or **ACBI(0)** register setting. **Figure 87** should clarify the ATOM channel startup behavior in this case for right shift. For left shift the **CM1** bit 0 in **Figure 87** has to be replaced by **CM1** bit 23.

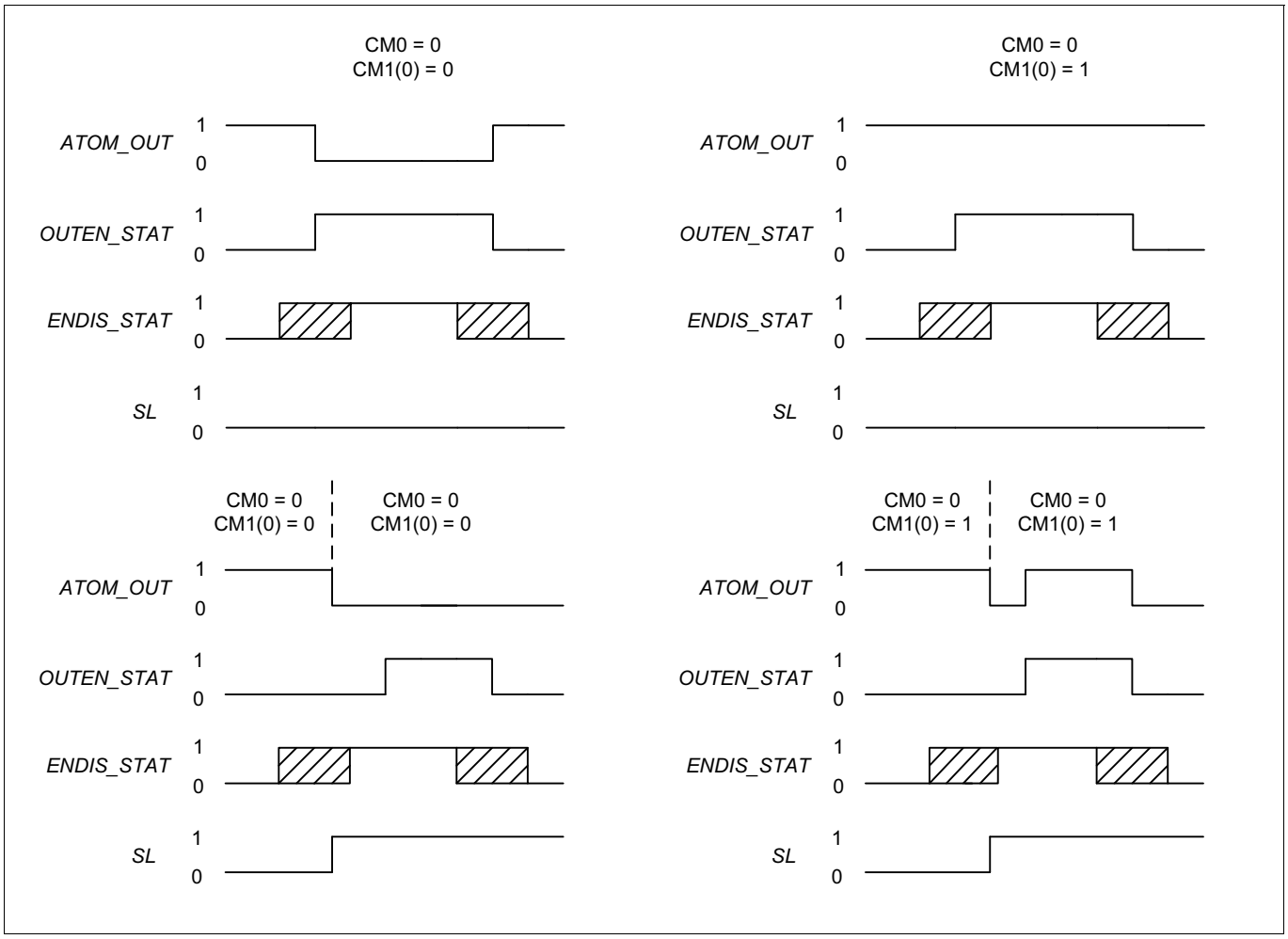


Figure 87 SOMS Output signal level at startup, UPEN_CTRL[x] disabled

If **UPEN_CTRL[x]** is set and the channel is enabled, the output level is defined by bit 0 or 23 of **CM1** register dependent on the shift direction. **Figure 88** shows the output behavior in that case.

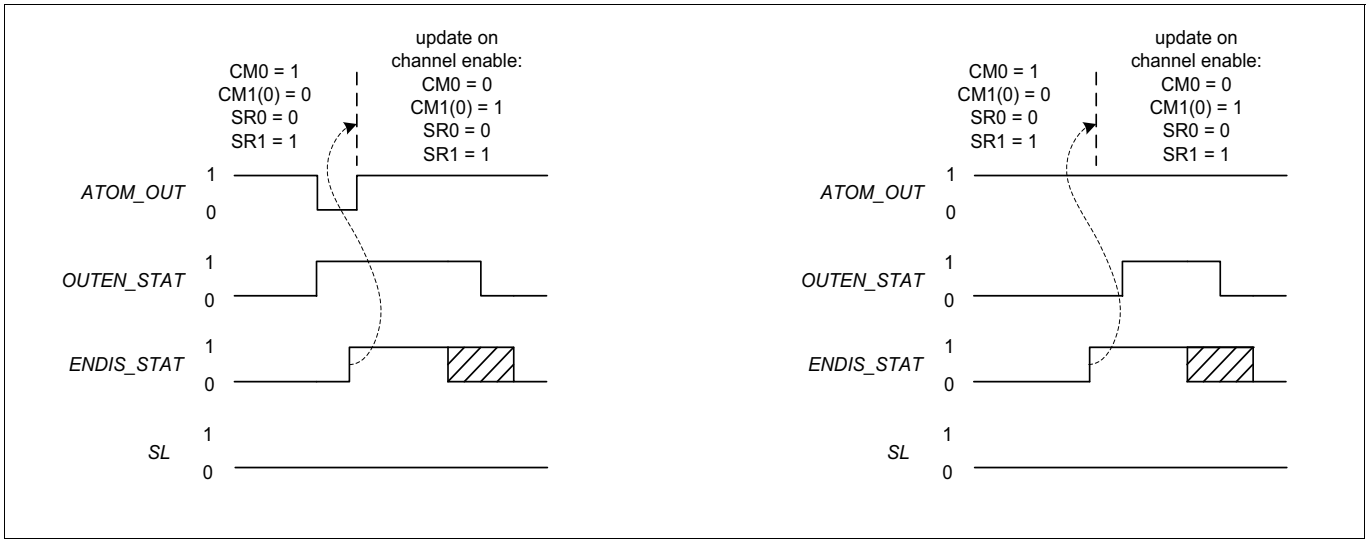


Figure 88 SOMS Output signal level at startup, UPEN_CTRL[x] enabled

When the serial data to be shifted is provided via ARU the number of bits that should be shifted has to be defined in the lower 24 bits of the ARU word (23 to 0) and the data that is to be shifted has to be defined in the ARU bits 47 to 24 aligned according to the shift direction. This shift direction has to be defined in the ARU word bit 48 (**ACB0** bit).

If bit **UPEN_CTRL[x]** of a channel x is set, after update of **CM0**/**CM1** register with the content of the **SR0**/**SR1** register, a new ARU read request is set up.

If bit **UPEN_CTRL[x]** of a channel x is not set, no (further) ARU read request is set up (because the **SR0**/**SR1** register are never used for update) and the ATOM may stop shifting after **CN0** has reached **CM0**. Note, that in this case also no automatic restart of shifting is possible.

If a channel is enabled with the settings SOMS mode and **ARU_EN** = 1, the first received values from ARU are stored in register **SR0** and **SR1**. If **CN0** and **CM0** are 0 (i.e. **CN0** is not counting) and the update of channel x is enabled (**UPEN_CTRL[x]**=1), an immediate update of the register **CM0** and **CM1** is also done. This update of **CM0** and **CM1** triggers the start of shifting.

It is recommended to configure the ATOM channel in One-shot mode when the **ARU_EN** bit is not set, since the ATOM channel would reload new values from the shadow registers when **CN0** reaches **CM0**.

28.15.3.4.1 SOMS mode with ARU_EN = 1 and OSM = 0, UPEN_CTRL[x] = 1

In case of bit **ARU_EN** is set and bit **OSM** is not set, the channel is running in the SOMS continuous mode. Then, if the content of the **CM0** register equals the counter **CN0**, the **CM0** and **CM1** registers are reloaded with the **SR0** and **SR1** content and new values are requested from the ARU. If the update of the shadow registers does not happen before **CN0** reaches **CM0** the old values of **SR0** and **SR1** are used to reload the operation registers.

In contrast to controlling the channel via AEI, the shift direction defined by ARU word bit 48 has only effect after the update of **CMx** operation registers from the **SRx** registers.

28.15.3.4.2 SOMS mode with ARU_EN = 1 and OSM = 1, UPEN_CTRL[x] = 1

In case of bit **ARU_EN** is set and bit **OSM** is set, the channel is running in the SOMS one-shot mode. Then, if the content of the **CM0** register equals the counter **CN0** and if new values are available in **SR0** and **SR1** (bit DV set), the **CM0** and **CM1** registers are reloaded with the **SR0** and **SR1** content and new values are requested from the ARU. If no new values are available in **SR0** and **SR1**, the register **CM0** and **CM1** will not be updated, the counter

CN0 stops and the ATOM channel continues to request new data from ARU. A later reception of new ARU data in **SR0** and **SR1** will immediately force the update of the register **CM0** and **CM1** and restart the counter **CN0**.

28.15.3.4.3 SOMS mode with ARU_EN = 0 and OSM = 0, UPEN_CTRL[x] = 1

In case of bit **ARU_EN** is not set and bit **OSM** is not set, the ATOM channel updates its CM0/CM1 register with the content of the SR0/SR1 register and restarts shifting immediately. The first bit of new CM1 register value will be applied at the output without any gap to the last bit of the previous CM1 register value.

28.15.3.4.4 SOMS mode with ARU_EN = 0 and OSM = 1, UPEN_CTRL[x] = 1:

In case of bit **ARU_EN** is not set and bit **OSM** is set, the ATOM channel stops shifting when **CN0** reaches **CM0** and no update of **CM0** and **CM1** is performed.

Then, the shifting of the channel can be restarted again by writing a zero to the **CN0** register again. Please note, that the **CN0** register should be written with a zero since the **CN0** register counts the number of bits shifted out by the ATOM channel.The writing of a zero to **CN0** causes also an immediate update of **CM0**/**CM1** register with the content of **SR0**/**SR1** register.

28.15.3.4.5 SOMS mode with double output

If in SOMS mode additionally the mode bit DSO is set (in register **ATOM[i]_CH[x]_CTRL**) two 12 bit data streams can be shifted out on the outputs *ATOM_OUT* and *ATOM_OUT_T* in parallel.This is reached by splitting the register **CM1** into two parts. The lower 12 bits are used as a shift register of output *ATOM_OUT* (i.e. bit 0 is assigned to output *ATOM_OUT*), the upper 12 bits are used as a shift register of output *ATOM_OUT_T* (i.e. bit 12 is assigned to output *ATOM_OUT_T*).On bit 23 and 11 of register **CM1** the value !**SL** is shifted in. Note: In this mode only shift right is possible. Bit **ACB0** is ignored.This behavior is depicted in the following figure:

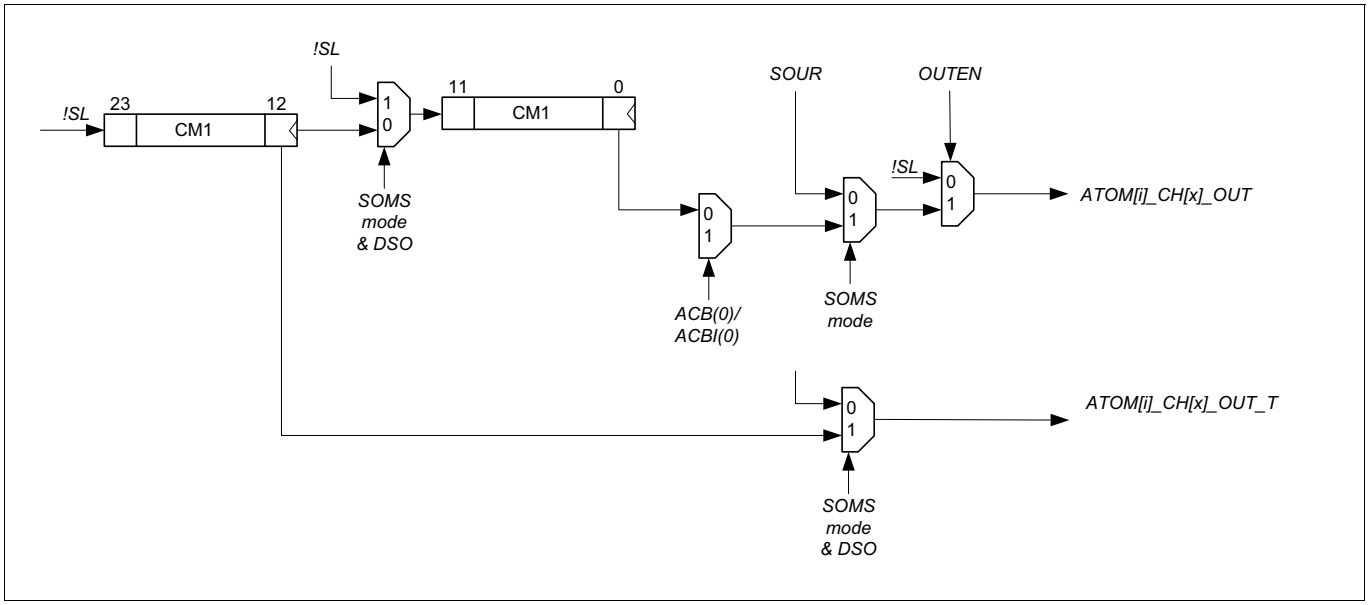


Figure 89 Double Output Shift Mode

28.15.3.4.6 Interrupts in SOMS mode

In ATOM Signal Output Mode Serial only the interrupt CCU0TC (**ATOM[i]_CH[x]_IRQ_NOTIFY**) in case of **CN0** >= **CM0** is generated. The interrupt **CCU1TC** has no meaning and is not generated.

28.15.3.4.7 Register ATOM[i]_CH[x]_CTRL in SOMS mode

Register ATOM[i]_CH[x]_CTRL in SOMS mode

GTM_ATOMi_CHx_SOMS (i=0-11; x=0-7)

ATOMi Channel x Control Register in SOMS Mode ($E8004_H$ + i*800_H+ x*80_H) **Reset Value: 00000800_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREEZE	Not_used	EXT_FUPD	Reserved	Not_used	OSM	Not_used	Not_used	Not_used			Not_used	Not_used		Not_used	Not_used
rw	rw	rw	r	rw	rw	rw	rw	r			rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECLK_SRC	CLK_SRC_SR			SL	Not_used	Not_used	Not_used	DSO	Not_used		ACB0	ARU_EN	Not_used	MODE	
rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	

Field	Bits	Type	Description
MODE	1:0	rw	**ATOM channel mode select** 11_B ATOM Signal Output Mode Serial (SOMS)
Not_used	2	rw	**Not used** Note: Not used in this mode.
ARU_EN	3	rw	**ARU Input stream enable** 0_B ARU Input stream disabled 1_B ARU Input stream enabled
ACB0	4	rw	**Shift direction for CM1 register** 0_B Right shift of data is started from bit 0 of CM1 1_B Left shift of data is started from bit 23 of CM1 Note: The data that has to be shifted out has to be aligned inside the CM1 register according to the defined shift direction. Note: This bit is only applicable if ARU_EN = 0. Note: If the direction (ACB0) is changed the output ATOM_OUT[x] switches immediately to the other 'first' bit of CM1 (bit 0 if ACB0 = 0, bit 23 if ACB0 = 1).
Not_used	6:5	rw	**Not used** Note: Not used in this mode.
DSO	7	rw	**Double Shift Output** 0_B CM1 is used as a 24 bit shift register 1_B CM1 is split into two 12 bit shift register Note: if DSO=1, only shift right is possible
Not_used	8	rw	**Not used** Note: Not used in this mode.
Not_used	9	rw	**Not used** Note: Not used in this mode.
Not_used	10	rw	**Not used** Note: Not used in this mode.

Field	Bits	Type	Description
SL	11	rw	**Defines signal level when channel and output is disabling** 0_B Low signal level 1_B High signal level Note: Reset value depends on the hardware configuration chosen by silicon vendor. Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL. Note: If the output is enabled, the output ATOM_OUT[x] is set to bit 0 or 23 of CM1 register. Note: The inverse value of SL is shifted into the CM1 register. If FREEZE=0, the following note is valid: If the channel is disabled, the output register of SOU unit is set to inverse value of SL. If FREEZE=1, the following note is valid: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.
CLK_SRC_SR	14:12	rw	**Shift frequency select for channel** If ECLK_SRC=0 / ECLK_SRC=1: 000_B *CMU_CLK0* selected / *CMU_CLK0* selected 001_B *CMU_CLK1* selected / *CMU_CLK1* selected 010_B *CMU_CLK2* selected / *CMU_CLK2* selected 011_B *CMU_CLK3* selected / Reserved 100_B *CMU_CLK4* selected / clock stopped 101_B *CMU_CLK5* selected / *TRIG[x-1]* selected 110_B *CMU_CLK6* selected / *TIM_EXT_CAPTURE[x]* selected 111_B *CMU_CLK7* selected / *CMU_CLK7* selected Note: This register is a shadow register for the register CLK_SRC. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done either by an end of a period or a FORCE_UPDATE. Note: After (channel) reset the selected CLK_SRC value is the SYS_CLK (input of Global Clock Divider). To use in SOMP mode one of the CMU_CLKx, it is recommended to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel.
ECLK_SRC	15	rw	**Extend CLK_SRC** 0_B CLK_SRC_SR set 1 selected 1_B CLK_SRC_SR set 2 selected See bit CLK_SRC_SR description for details.
Not_used	16	rw	**Not used** Note: Not used in this mode.
Not_used	17	rw	**Not used** Note: Not used in this mode.
Not_used	19:18	rw	**Not used** Note: Not used in this mode.
Not_used	20	rw	**Not used** Note: Not used in this mode.

Field	Bits	Type	Description
Not_used	23:21	r	**Not used** Note: Not used in this mode.
Not_used	24	rw	**Not used** Note: Not used in this mode.
Not_used	25	rw	**Not used** Note: Not used in this mode.
OSM	26	rw	**One-shot mode** 0_B Continuous shifting is enabled 1_B Channel stops, after number of bits defined in CM0 is shifted out
Not_used	27	rw	**Not used** Note: Not used in this mode.
Reserved	28	r	**Reserved** Read as zero, should be written as zero.
EXT_FUPD	29	rw	**External forced update** 0_B use FUPD(x) signal from AGC to force update 1_B use TIM_EXT_CAPTURE signal to force update Note: This bit is only applicable in SOMP and SOMS mode.
Not_used	30	rw	**Not used** Note: Not used in this mode.
FREEZE	31	rw	**FREEZE** 0_B a channel disable/enable may change internal register and output register 1_B a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

28.15.3.5 ATOM Signal Output Mode Buffered Compare (SOMB)

28.15.3.5.1 Overview

In ATOM Signal Output Mode buffered Compare (SOMB) the output action is performed according to the comparison result of the input values located in **CM0** and/or **CM1** registers and the two (three) time base values *TBU_TS0* or *TBU_TS1* (or *TBU_TS2*) provided by the TBU. For a description of the time base generation please refer to the TBU specification in the chapter “Time Base Unite (TBU)”. It is configurable, which of the two (three) time bases is to be compared with one or both values in **CM0** and **CM1**.

The compare strategy of the two compare units CCU0 and CCU1 is controlled by the value of bit field ACBI of register **ATOM[i]_CH[x]_STAT**. This bit field is only readable by CPU. If ARU is disabled, the bit field ACBI can only be updated with the value of bit field **ACB** of register **ATOM[i]_CH[x]_CTRL**. If ARU is enabled, the **ACBI** bit field can be updated with the value of shadow register **ACB_SR** which contains a value received via ARU or the value of bit field **ACB** of register **ATOM[i]_CH[x]_CTRL**.

The table below lists all valid control configurations for bit field **ACBI** of register **ATOM[i]_CH[x]_STAT**.

Table 57 ATOM SOMB compare strategies

ACBI(4)	ACBI(3)	ACBI(2)	CCUx control
0	0	0	Reserved. Has no effect.
0	0	1	Reserved. Has no effect.
0	1	0	Compare in CCU0 only, use time base TBU_TS0. Output signal level is defined by combination of SL, ACB10/ACBI(1…0) bits.
0	1	1	Compare in CCU1 only, use time base TBU_TS1 or TBU_TS2. Output signal level is defined by combination of SL, ACBI[1:0] bits.
1	0	0	Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS0. Output signal level when CCU0 matches is defined by combination of SL, ACBI[1:0]. On the CCU1 match the output level is toggled.
1	0	1	Serve Last: Compare in CCU0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU0 matches is defined by combination of SL, ACBI[1:0]. On the CCU1 match the output level is toggled.
1	1	0	Serve Last: Compare in CCU0 using TBU_TS0 and then in CCU1 using TBU_TS1 or TBU_TS2. Output signal level when CCU1 matches is defined by combination of SL, ACBI[1:0]
1	1	1	Cancels pending comparison independent on ARU_EN

The CCUx trigger signals *TRIG_CCU0* and *TRIG_CCU1* creates edges depending on the combination of the predefined signal level in **SL** bit and the two control bits **ACBI[1:0]**.

In SOMB mode, if ARU access is enabled, the new compare values received via ARU are always stored in the shadow register **SR0** and **SR1** and the **ACB** bits are stores in an internal register **ACB_SR**.

If the scheduled compare matches in CCU0 and/or CCU1 are occurred and the **SRx** register contain new valid values, the register **CM0** and **CM1** are updated automatically with the content of the corresponding **SRx** register, the **ACBI** bit field is updated with the content of internal **ACB_SR** register and the DV bit of register **ATOM[i]_CH[x]_STAT** is set. If the **SRx** register and the **CMx** register contain no valid value, the compare units are waiting in an idle state.

On a compare match of one of the compare units CCUx units the output ATOM_OUT is set according to combination of **ACBI** bit 1 down to 0 (in register **ATOM[i]_CH[x]_STAT**) and the **SL** bit of register **ATOM[i]_CH[x]_CTRL**.

Table 58 ATOM SOMB output control by ACBI[1:0] and SL

SL	ACBI(1)	ACBI(0)	Output Behavior
0	0	0	No signal level change at output.
0	0	1	Set output signal level to 1.
0	1	0	Set output signal level to 0.
0	1	1	Toggle output signal level.
1	0	0	No signal level change at output.
1	0	1	Set output signal level to 0.
1	1	0	Set output signal level to 1.
1	1	1	Toggle output signal level.

In opposite to SOMC mode no time stamp value of TBU is captured in **SRx** register.

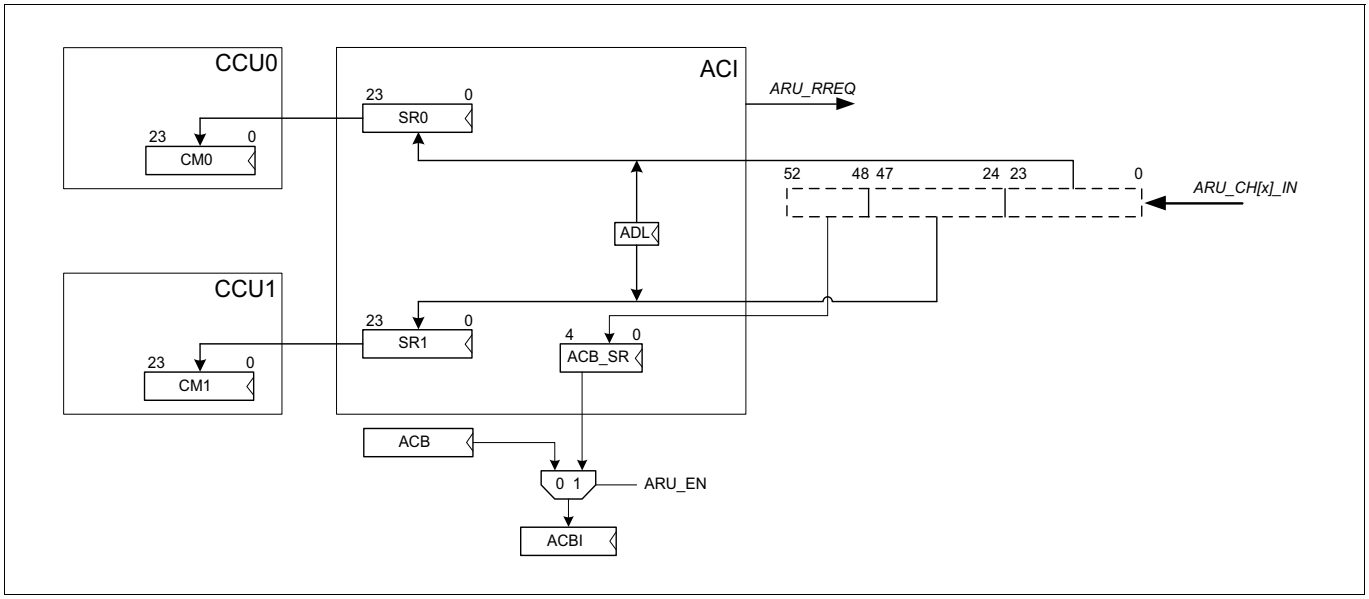


Figure 90 ARU interface behavior in SOMB mode

The flag **DV** of register **ATOM[i]_CH[x]_STAT** indicates that at least one of the **CMx** register contains valid data and a compare event may be pending (if channel is enabled).

The **DV** flag is reset if none of the **CMx** register contains valid data.

28.15.3.5.2 SOMB under CPU control

If bit **ARU_EN** of register **ATOM[i]_CH[x]_CTRL** is not set, the ATOM channel can only be controlled via CPU.

Writing to one of the **CMx** register sets automatically the **DV** bit to validate the new compare value. A comparison depending on value **ACBI** of register **ATOM[i]_CH[x]_STAT** is started immediately.Because only the **ACB** bit of register **ATOM[i]_CH[x]_CTRL** can be written and this bit field serves as a shadow register for the work register **ACBI** (bit field of register **ATOM[i]_CH[x]_STAT)**, it is recommended to first update the **ACB** bit field before updating **CMx**/**SRx** register.

The compare strategy is controlled by the value stored in bit field **ACBI** of register **ATOM[i]_CH[x]_STAT**. If ARU is disabled, this bit field can only be updated with the value of bit field **ACB** of register **ATOM[i]_CH[x]_CTRL**.

The update of bit field **ACBI** can be triggered by a forced update or the normal update mechanism controlled by bit **UPEN_CTRL[x]** in register **ATOM[i]_AGC_GLB_CTRL**.

Writing to one of the **SRx** register and triggering a forced update, updates the **CMx** register with the value of **SRx** register and the **ACBI** bit field with the content of **ACB** bit field of register **ATOM[i]_CH[x]_CTRL**. A new comparison is started.

Writing to one of the **SRx** register while update of **CMx** register is disabled (**UPEN_CTRL[x]** = 0 in **ATOM[i]_AGC_GLB_CTRL**) and enabling update afterwards, triggers the update of **CMx** register and the **ACBI** bit field and starts comparison if previous comparison is finished (**DV** bit was reset).

If ARU access is disabled (**ARU_EN**=0), a force update updates the **CMx** register with the content of **SRx** register and the **ACBI** bit field with the content of **ACB** bit field of register **ATOM[i]_CH[x]_CTRL**.

28.15.3.5.3 SOMB under ARU control

If both compare units CCU0/CCU1 are finished with previous job (depending on compare strategy) and the **SRx** register contain no new value, they are waiting until new data was received via ARU and stored in **SRx** register. Then, an immediately update takes place.

If both compare units are finished with previous job (depending on compare strategy) and there are new data available in **SRx** register, the update the **CMx** register with the value of the **SRx** register and the **ACBI** bit field with the value of internal **ACB_SR** register takes place and a new compare job is started immediately.

After an update of the **CMx** register, a new ARU read request is set.

New compare values received via the ARU are stored in shadow register **SRx**. The **ACB** bits received via ARU are stored in the internal register **ACB_SR**.

If ARU access is enabled (**ARU_EN**=1), a force update updates the **CMx** register with the content of **SRx** register and the **ACBI** bit field with the content of internal **ACB_SR** register.

For compare strategy 'serve last' the CCU0 and CCU1 compare match may occur sequentially. During different phases of compare match the CPU access rights to register **CM0** and **CM1** as well as to **WR_REQ** bit is different. These access rights by CPU to register **CM0** and **CM1** and the **WR_REQ** are depicted in the following figure for the case of **EUPM**=0 (register **ATOM[i]_CH[x]_CTRL**)

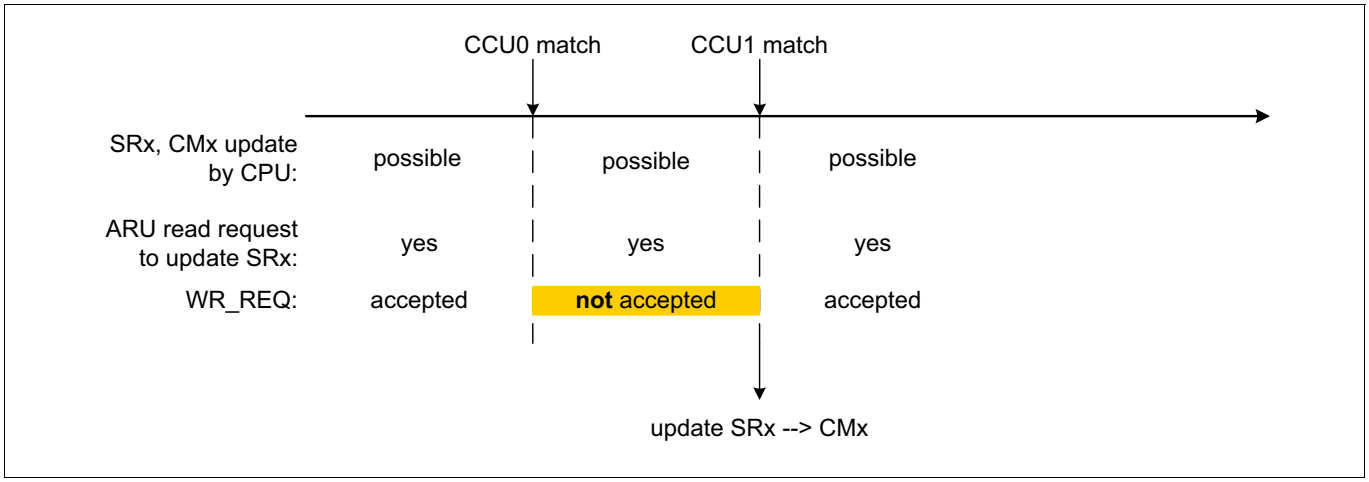


Figure 91 CPU access rights in case of compare strategy 'serve last' and EUPM=0

The access rights by CPU to register **CM0** and **CM1** and the WR_REQ are depicted in the following figure for the case of EUPM=1 (register **ATOM[i]_CH[x]_CTRL**)

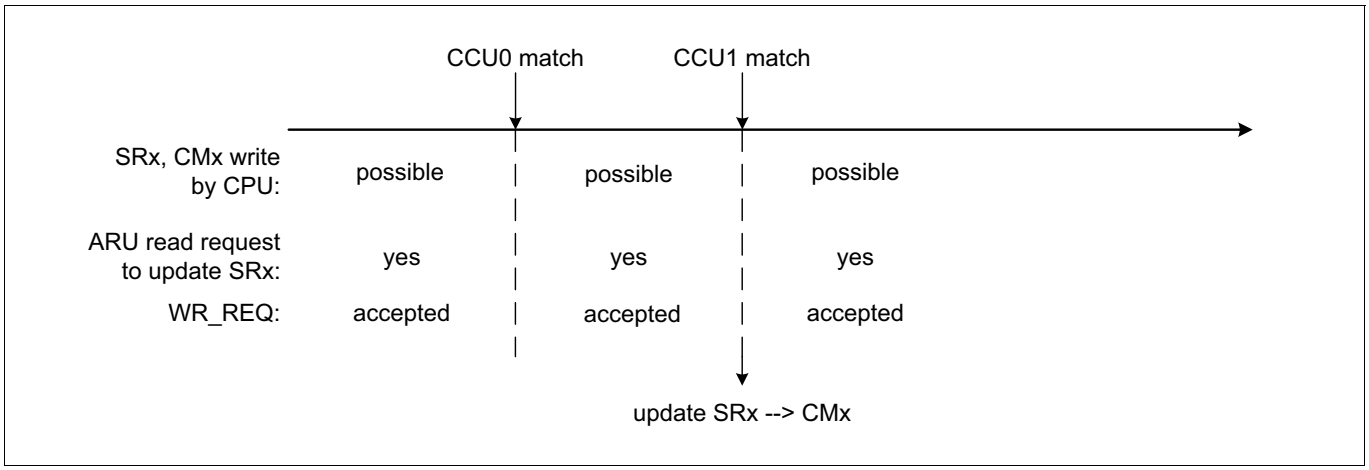


Figure 92 CPU access rights in case of compare strategy 'serve last' in case of EUPM=1

ARU Non-blocking mode

If bit **ABM** in register **ATOM[i]_CH[x]_CTRL** is not set, the ARU blocking mode is disabled. In this case the ATOM channel is continuously reading via ARU and storing new values in the **SRx** register and the ACB shadow register **ACB_SR**.

If **ARU_EN** is not set, the bit **ABM** has no meaning.

ARU Blocking mode

If bit **ABM** in register **ATOM[i]_CH[x]_CTRL** is set, the ARU blocking mode is enabled. In this case the ATOM channel stops requesting new **SRx** values via ARU after reception of a new **SRx** value and restarts requesting a new value via ARU after compare match on both compare units (depending on compare strategy) followed by the immediate update of the **CMx** register with content of **SRx** register and an update of **ACBI** with the content of **ACB_SR**.

If **ARU_EN** is not set, the bit **ABM** has no meaning.

Late Update by CPU

Although, the ATOM channel may be controlled by data received via the ARU, the CPU is able to request at any time a late update of the compare register. This can be initiated by setting the **WR_REQ** bit inside the **ATOM[i]_CH[x]_CTRL** register.

If none of the two compare match event happened, the ATOM channel accepts the setting of **WR_REQ** bit. In this case, the ATOM will request no further data from ARU (if ARU access was enabled) and will disable the update of **CMx** register with the content of **SRx** register on a compare match event.

If at least one of the requested compare match events happened (depending on strategy) the **WR_REQ** bit is not set and the **WRF** flag in register **ATOM[i]_CH[x]_STAT** is set to indicate that the late update was not successful.

The channel will in any case continue to compare against the values stored inside the compare registers (if bit **DV** was set). The CPU can now update the compare values by writing to the shadow registers and force the ATOM channel to update the compare registers by writing to the force update register bits in the AGC register.

With a force update the **WR_REQ** bit is reset automatically and the ARU read request is set up again (if ARU access was enabled).

28.15.3.5.4 Register ATOM[i]_CH[x]_CTRL in SOMB mode

Register ATOM[i]_CH[x]_CTRL in SOMB mode

GTM_ATOMi_CHx_SOMB (i=0-11; x=0-7)

ATOMi Channel x Control Register in SOMB Mode ($E8004_H$ + i*800_H+ x*80_H) **Reset Value: 00000800_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREEZE	SOMB	Not_used	Reserved	ABM	Not_used	Not_used	TRIGOUT	EXTTRIGOUT	Not_used		Not_used	Not_used		Not_used	WR_REQ
rw	rw	rw	r	rw	rw	rw	rw	rw	r		rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Not_used	Not_used			SL	EUPM	CMP_CTRL	ACB[4:2]			ACB[1:0]		ARU_EN	TB12_SEL	MODE	
rw	rw			rw	rw	rw	rw			rw		rw	rw	rw	

Field	Bits	Type	Description
MODE	1:0	rw	**ATOM channel mode select** Not used in ATOM SOMB mode.
TB12_SEL	2	rw	**Select time base value TBU_TS1 or TBU_TS2** 0_B *TBU_TS1* selected for comparison 1_B *TBU_TS2* selected for comparison Note: This bit is only applicable if three time bases are present in the GTM. Otherwise, this bit is reserved.
ARU_EN	3	rw	**ARU Input stream enable** 0_B ARU Input stream disabled 1_B ARU Input stream enabled
ACB[1:0]	5:4	rw	**Signal level control bits** 00_B No signal level change at output. 01_B Set output signal level to 1 when SL bit = 0 else output signal level to 0. 10_B Set output signal level to 0 when SL bit = 0 else output signal level to 1. 11_B Toggle output signal level. Note: These bits are only applicable if ARU_EN = 0.
ACB[4:2]	8:6	rw	**ATOM SOMB compare strategy** 000_B Reserved. Has no effect. 001_B Reserved. Has no effect. 010_B Compare in CCU0 only against *TBU_TS0*. 011_B Compare in CCU1 only against *TBU_TS1* or *TBU_TS2*. 100_B Compare first in CCU0 and then in CCU1. Use *TBU_TS0*. 101_B Compare first in CCU0 and then in CCU1. Use *TBU_TS1* or *TBU_TS2*. 110_B Compare first in CCU0 and then in CCU1. Use *TBU_TS0* in CCU0 and *TBU_TS1* or *TBU_TS2* in CCU1. 111_B Cancel pending comparisons independent on ARU_EN. Note: These bits are only applicable if ARU_EN = 0.

Field	Bits	Type	Description
CMP_CTRL	9	rw	**CCUx compare strategy select** 0_B Greater-equal compare against TBU time base values (TBU_TS1/2 >= CM0/1) 1_B Less-equal compare against TBU time base values (TBU_TS1/2 <= CM0/1) Note: The compare unit CCU0 or CCU1 that compares against TBU_TS0 (depending on CCUx control mode defined by ACB_CM(4:2)) always performs a greater-equal comparison, independent on CMP_CTRL bit.
EUPM	10	rw	**Extended update mode** 0_B No extended update of CM0 and CM1 via CPU or ARU 1_B Extended update mode in case of compare strategy 'serve last': update of CM1 after CCU0 compare match possible via ARU or CPU. Note: If EUPM=1 write access to CM0 or CM1 never causes an AEI write status 0b10. Note: This bit is only applicable in SOMC and SOMB mode.
SL	11	rw	**Initial signal level after channel enable** 0_B Low signal level 1_B High signal level Note: Reset value depends on the hardware configuration chosen by silicon vendor. Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse value of SL. If FREEZE=0, the following notes is valid: If the channel is disabled, the output register of SOU unit is set to value of SL. If FREEZE=1, the following note is valid: If the channel is disabled, the output register of SOU unit is not changed and output ATOM_OUT[x] is not changed.
Not_used	14:12	rw	**Not used** Note: Not used in this mode.
Not_used	15	rw	**Not used** Note: Not used in this mode.
WR_REQ	16	rw	**CPU Write request bit for late compare register update** 0_B No late update requested by CPU 1_B Late update requested by CPU Note: The CPU can disable subsequent ARU read requests by the channel and can update the shadow registers with new compare values, while the compare units operate on old compare values received by former ARU accesses, if occurred. Note: On a compare match event, the WR_REQ bit will be reset by hardware. Note: At the point of the force update only the shadow registers SR0 and SR1 are transferred into the CM0, CM1 registers. The output action is still defined by the ACBI bit field described by the ARU together with the old compare values for CM0/CM1.
Not_used	17	rw	**Not used** Note: Not used in this mode.

Field	Bits	Type	Description
Not_used	19:18	rw	**Not used** Note: Not used in this mode.
Not_used	20	rw	**Not used** Note: Not used in this mode.
Not_used	22:21	r	**Not used** Note: Not used in this mode.
EXTTRIGOUT	23	rw	**TIM_EXT_CAPTURE(x) as potential output signal TRIG_[x]** 0_B signal *TRIG_[x-1]* is selected as output on *TRIG_[x]* (if TRIGOUT=0) 1_B signal *TIM_EXT_CAPTURE(x)* is selected as output on *TRIG_[x]* (if TRIGOUT=0)
TRIGOUT	24	rw	**Trigger output selection (output signal TRIG_CHx) of module ATOM_CHx** 0_B *TRIG_[x]* is *TRIG_[x-1]* or TIM_*EXT_CAPTURE(x)*. 1_B *TRIG_[x]* is *TRIG_CCU0*
Not_used	25	rw	**Not used** Note: Not used in this mode.
Not_used	26	rw	**Not used** Note: Not used in this mode.
ABM	27	rw	**ARU blocking mode** 0_B ARU blocking mode disabled: ATOM reads continuously from ARU and updates SR0, SR1 and ACB bits independent of pending compare match event 1_B ARU blocking mode enabled: after updating SR0,SR1 and ACB bits via ARU, no new data is read via ARU until compare match event occurred.
Reserved	28	r	**Reserved** Read as zero, should be written as zero.
Not_used	29	rw	**Not used** Note: Not used in this mode.
SOMB	30	rw	**SOMB: SOMB mode** 0_B ATOM channel mode defined by bit filed MODE 1_B ATOM SOMB mode enabled
FREEZE	31	rw	**FREEZE** 0_B a channel disable/enable may change internal register and output register 1_B a channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)

28.15.4 ATOM Interrupt Signals

Table 59 ATOM Interrupt Signals

Signal	Description
CCU0TCx_IRQ	CCU0 Trigger condition interrupt for channel x
CCU1TCx_IRQ	CCU1 Trigger condition interrupt for channel x

28.15.5 ATOM Register Overview

Table 60 ATOM Register Overview

Register name	Description	see Page
ATOM[i]_AGC_GLB_CTRL	ATOMi AGC global control register	297
ATOM[i]_AGC_ENDIS_CTRL	ATOMi AGC enable/disable control register	298
ATOM[i]_AGC_ENDIS_STAT	ATOMi AGC enable/disable status register	299
ATOM[i]_AGC_ACT_TB	ATOMi AGC action time base register	300
ATOM[i]_AGC_OUTEN_CTRL	ATOMi AGC output enable control register	301
ATOM[i]_AGC_OUTEN_STAT	ATOMi AGC output enable status register	301
ATOM[i]_AGC_FUPD_CTRL	ATOMi AGC force update control register	302
ATOM[i]_AGC_INT_TRIG	ATOMi AGC internal trigger control register	303
ATOM[i]_CH[x]_CTRL	ATOMi channel x control register	304
	ATOMi channel x control register in SOMI mode	248
	ATOMi channel x control register in SOMC mode	265
	ATOMi channel x control register in SOMP mode	279
	ATOMi channel x control register in SOMS mode	286
	ATOMi channel x control register in SOMB mode	293
ATOM[i]_CH[x]_STAT	ATOMi channel x status register	309
ATOM[i]_CH[x]_RDADDR	ATOMi channel x ARU read address register	311
ATOM[i]_CH[x]_CN0	ATOMi channel x CCU0 counter register	311
ATOM[i]_CH[x]_CM0	ATOMi channel x CCU0 compare register	312
ATOM[i]_CH[x]_SR0	ATOMi channel x CCU0 compare shadow register	312
ATOM[i]_CH[x]_CM1	ATOMi channel x CCU1 compare register	313
ATOM[i]_CH[x]_SR1	ATOMi channel x CCU1 compare shadow register	314
ATOM[i]_CH[x]_IRQ_NOTIFY	ATOMi channel x interrupt notification register	314
ATOM[i]_CH[x]_IRQ_EN	ATOMi channel x interrupt enable register	315
ATOM[i]_CH[x]_IRQ_FORCINT	ATOMi channel x software interrupt generation	316
ATOM[i]_CH[x]_IRQ_MODE	ATOMi channel x interrupt mode configuration register	316

28.15.6 ATOM Register Description

28.15.6.1 Register ATOM[i]_AGC_GLB_CTRL

ATOMi AGC Global Control Register

ATOMi_AGC_GLB_CTRL (i=0-11)
ATOMi AGC Global Control Register **($0E8040_H$+i*800_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPEN_CTRL7		UPEN_CTRL6		UPEN_CTRL5		UPEN_CTRL4		UPEN_CTRL3		UPEN_CTRL2		UPEN_CTRL1		UPEN_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST_CH7	RST_CH6	RST_CH5	RST_CH4	RST_CH3	RST_CH2	RST_CH1	RST_CH0	0							HOST_TRIG
w	w	w	w	w	w	w	w	r							w

Field	Bits	Type	Description
HOST_TRIG	0	w	**Trigger request signal (see AGC) to update the register ENDIS_STAT and OUTEN_STAT** Note: this flag is reset automatically after triggering the update 0_B No trigger request 1_B Set trigger request
RST_CHx (x=0-7)	x+8	w	**Software reset of channel x** *Note: This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately. The output register of SOU unit is reset to inverse reset value of SL bit.* 0_B No action 1_B Reset channel
UPEN_CTRLx (x=0-7)	2*x+17:2*x+16	rw	**ATOM channel x enable update of register CM0, CM1 and CLK_SRC from SR0, SR1 and CLK_SRC_SR** Write / Read: 00_B Don't care, bits 1:0 will not be change / update disabled 01_B Disable update / -- 10_B Enable update / -- 11_B Don't care, bits 1:0 will not be changed / update enabled
0	7:1	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.2 Register ATOM[i]_AGC_ENDIS_CTRL

ATOMi AGC Enable/Disable Control Register

ATOMi_AGC_ENDIS_CTRL (i=0-11)

ATOMi AGC Enable/Disable Control Register($0E8044_H$+i*800_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIS_CTRL7		ENDIS_CTRL6		ENDIS_CTRL5		ENDIS_CTRL4		ENDIS_CTRL3		ENDIS_CTRL2		ENDIS_CTRL1		ENDIS_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
ENDIS_CTRLx (x=0-7)	2*x+1:2*x	rw	**ATOM channel x enable/disable update value** If FREEZE=0 and an ATOM channel is disabled, the counter **CN0** is stopped and the output register of SOU unit is set to the inverse value of control bit SL. On an enable event, the counter **CN0** starts counting from its current value. If FREEZE=1 and an ATOM channel is disabled, the counter CN0 is stopped (SOMP, SOMS mode) and each comparison is stopped (SOMC, SOMB mode). On an enable event, the counter CN0 starts counting from its current value or a comparison is restarted. Write of following double bit values is possible: *Note: If the output is disabled (OUTEN[x]=0), the ATOM channel x output ATOM_OUT[x] is the inverted value of bit SL.* 00_B Don't care, bits 1:0 of register ENDIS_STAT will not be changed on an update trigger 01_B Disable channel on an update trigger 10_B Enable channel on an update trigger 11_B Don't change bits 1:0 of this register
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.3 Register ATOM[i]_AGC_ENDIS_STAT

ATOMi AGC Enable/Disable Status Register

ATOMi_AGC_ENDIS_STAT (i=0-11)

ATOMi AGC Enable/Disable Status Register ($0E8048_H$+i*800_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIS_STAT7		ENDIS_STAT6		ENDIS_STAT5		ENDIS_STAT4		ENDIS_STAT3		ENDIS_STAT2		ENDIS_STAT1		ENDIS_STAT0	
rw		rw		rw		rw		rw		rw		rw		rw	

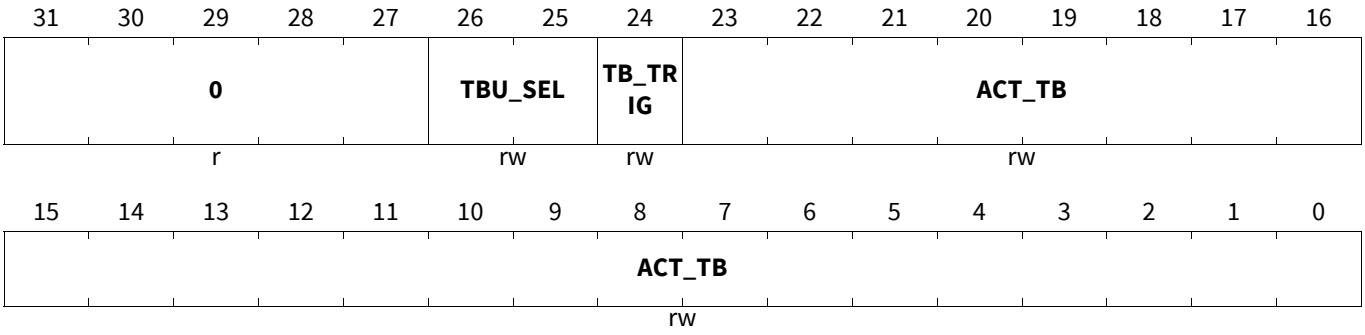
Field	Bits	Type	Description
ENDIS_STATx (x=0-7)	2*x+1:2*x	rw	**ATOM channel x enable/disable** If FREEZE=0 and an ATOM channel is disabled, the counter CN0 is stopped and the output register of SOU unit is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value. If FREEZE=1 and an ATOM channel is disabled, the counter CN0 is stopped (SOMP, SOMS mode) and each comparison is stopped (SOMC, SOMB mode). On an enable event, the counter CN0 starts counting from its current value or a comparison is restarted. 00_B Write: Don't care bits, will not be changed; Read: Output disabled. 01_B Write: Disable channel on an update trigger; Read: Unused 10_B Write: Enable channel on an update trigger; Read: Unused 11_B Write: Don’t change bits 1:0 of this register; Read: Output enabled.
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.4 Register ATOM[i]_AGC_ACT_TB

ATOMi AGC Action Time Base Register

ATOMi_AGC_ACT_TB (i=0-11)

ATOMi AGC Action Time Base Register **($0E804C_H$+i*800_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ACT_TB	23:0	rw	**Time base value** Specifies the signed compare value with selected signal TBU_TS[x], x=0..2. If selected TBU_TS[x] value is in the interval [ACT_TB-007FFFFFh,ACT_TB], the event is in the past, and the trigger is generated immediately. Otherwise, the event is in the future, and the trigger is generated if selected TBU_TS[x] is equal to ACT_TB.
TB_TRIG	24	rw	**Set trigger request** Note: This flag is reset automatically if the selected time base unit (*TBU_TS0*or *TBU_TS1*or *TBU_TS2,*if present) has reached the value **ACT_TB**, and the update of the register was triggered. 0_B No trigger request 1_B Set trigger request
TBU_SEL	26:25	rw	**Selection of time base used for comparison** *Note: The bit combination 0b10 is only applicable if the TBU of the device contains three time base channels. Otherwise, this bit combination is also reserved. Please refer to GTM Architecture block diagram on page 3 to determine the number of channels for TBU of this device.* 00_B TBU_TS0 selected 01_B TBU_TS1 selected 10_B TBU_TS2 selected 11_B Same as 0b00
0	31:27	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.5 Register ATOM[i]_AGC_OUTEN_CTRL

ATOMi AGC Output Enable Control Register

ATOMi_AGC_OUTEN_CTRL (i=0-11)

ATOMi AGC Output Enable Control Register($0E8050_H$+i*800_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTEN_CTRL 7		OUTEN_CTRL 6		OUTEN_CTRL 5		OUTEN_CTRL 4		OUTEN_CTRL 3		OUTEN_CTRL 2		OUTEN_CTRL 1		OUTEN_CTRL 0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
OUTEN_CTRL x (x=0-7)	2*x+1:2*x	rw	**Output ATOM_OUTx enable/disable update value** Write of following double bit values is possible: *Note: If the channel is disabled (ENDIS[0]=0) or the output is disabled (OUTEN[0]=0), the TOM channel 0 output ATOM_OUT[0] is the inverted value of bit SL.* 00_B Don't care, bits 1:0 of register OUTEN_STAT will not be changed on an update trigger 01_B Disable channel output on an update trigger 10_B Enable channel output on an update trigger 11_B Don't change bits 1:0 of this register
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.6 Register ATOM[i]_AGC_OUTEN_STAT

ATOMi AGC Output Enable Status Register

ATOMi_AGC_OUTEN_STAT (i=0-11)

ATOMi AGC Output Enable Status Register ($0E8054_H$+i*800_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTEN_STAT 7		OUTEN_STAT 6		OUTEN_STAT 5		OUTEN_STAT 4		OUTEN_STAT 3		OUTEN_STAT 2		OUTEN_STAT 1		OUTEN_STAT 0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
OUTEN_STATx (x=0-7)	2*x+1:2*x	rw	**Control/status of output ATOM_OUTx** Write / Read : 00_B Don't care, bits 1:0 will not be changed / output disabled 01_B Disable output / -- 10_B Enable output / -- 11_B Don't care, bits 1:0 will not be changed / output enabled
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.7 Register ATOM[i]_AGC_FUPD_CTRL

ATOMi AGC Force Update Control Register

ATOMi_AGC_FUPD_CTRL (i=0-11)

ATOMi AGC Force Update Control Register ($0E8058_H$+i*800_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSTCN0_CH7		RSTCN0_CH6		RSTCN0_CH5		RSTCN0_CH4		RSTCN0_CH3		RSTCN0_CH2		RSTCN0_CH1		RSTCN0_CH0	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FUPD_CTRL7		FUPD_CTRL6		FUPD_CTRL5		FUPD_CTRL4		FUPD_CTRL3		FUPD_CTRL2		FUPD_CTRL1		FUPD_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
FUPD_CTRLx (x=0-7)	2*x+1:2*x	rw	**Force update of ATOM channel x operation registers** If enabled, force update of register CM0, CM1 and CLK_SRC triggered by HOST_TRIG, ACT_TB compare match, or internal trigger. Write / Read : *Note: In SOMP mode, the force update request is stored and executed synchronized to the selected CMU_CLK. In all other modes, the force update request is executed immediately.* *Note: In SOMP mode, in case of ECLK_SRC=1 and CLK_SRC_SR = 0b011/0b100/0b101/0b110 a force update leads to an immediate update of CM0, CM1 and CLK_SRC.* 00_B Don't care, bits 1:0 will not be changed / force update disabled 01_B Disable force update / -- 10_B Enable force update / -- 11_B Don't care, bits 1:0 will not be changed / force update enabled

Field	Bits	Type	Description
RSTCN0_CHx (x=0-7)	2*x+17:2*x +16	rw	**Reset CN0 of channel x on force update event** If enabled, reset CN0 triggered by HOST_TRIG, ACT_TB compare match, or internal trigger. Write / Read : 00_B Don't care, bits 1:0 will not be changed / CN0 is not reset on forced update 01_B Do not reset CN0 on forced update / -- 10_B Reset CN0 on forced update / -- 11_B Don't care, bits 1:0 will not be changed / CN0 is reset on forced update

28.15.6.8 Register ATOM[i]_AGC_INT_TRIG

ATOMi AGC Internal Trigger Control Register

ATOMi_AGC_INT_TRIG (i=0-11)

ATOMi AGC Internal Trigger Control Register(0E805C$_H$+i*800$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_TRIG7		INT_TRIG6		INT_TRIG5		INT_TRIG4		INT_TRIG3		INT_TRIG2		INT_TRIG1		INT_TRIG0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
INT_TRIGx (x=0-7)	2*x+1:2*x	rw	**Select input signal TRIG_x as a trigger source** Write / Read : 00_B Don't care, bits 1:0 will not be changed / internal trigger from channel 0 (TRIG_0) not used 01_B Do not use internal trigger from channel 0 (TRIG_0) / -- 10_B Use internal trigger from channel 0 (TRIG_0) / -- 11_B Don't care, bits 1:0 will not be changed / internal trigger from channel 0 (TRIG_0) used
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.9 Register ATOM[i]_CH[x]_CTRL

ATOMi Channel x Control Register

ATOMi_CHx_CTRL (i=0-11;x=0-7)

ATOMi Channel x Control Register **($0E8004_H$+i*800_H+x*80_H)** **Application Reset Value: 0000 0800_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREEZE	SOMB	EXT_FUPD	0	ABM	OSM	SLA	TRIGOUT	EXTTRIGOUT	EXT_TRIG	OSM_TRIG	RST_CCU0	UDMODE		TRIG_PULSE	WR_REQ
rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECLK_SRC	CLK_SRC_SR			SL	EUPM	CMP_CTRL	ACB					ARU_EN	TB12_SEL	MODE	
rw	rw			rw	rw	rw	rw					rw	rw	rw	

Field	Bits	Type	Description
MODE	1:0	rw	**ATOM channel mode select** 00_B ATOM Signal Output Mode Immediate (SOMI) 01_B ATOM Signal Output Mode Compare (SOMC) 10_B ATOM Signal Output Mode PWM (SOMP) 11_B ATOM Signal Output Mode Serial (SOMS)
TB12_SEL	2	rw	**Select time base value TBU_TS1 or TBU_TS2** *Note: This bit is only applicable in SOMC mode.* 0_B TBU_TS1 selected for comparison 1_B TBU_TS2 selected for comparison
ARU_EN	3	rw	**ARU input stream enable** 0_B ARU Input stream disabled 1_B ARU Input stream enabled
ACB	8:4	rw	**ATOM Mode control bits** These bits have different meaning in the different ATOM channel modes. Please refer to the mode description sections. SOMI : **Section 28.15.3.1** and **Register ATOM[i]_CH[x]_CTRL in SOMI mode** for register description. SOMC : **Section 28.15.3.2** and **Register ATOM[i]_CH[x]_CTRL in SOMC mode** for register description. SOMP : **Section 28.15.3.3** and **Register ATOM[i]_CH[x]_CTRL in SOMP mode** for register description. SOMS : **Section 28.15.3.4** and **Register ATOM[i]_CH[x]_CTRL in SOMS mode** for register description. SOMB : **Section 28.15.3.5** and **Register ATOM[i]_CH[x]_CTRL in SOMB mode** for register description.

Field	Bits	Type	Description
CMP_CTRL	9	rw	**CCUx compare strategy select** *Note: This bit is only applicable in SOMC mode.* 0_B Greater-equal compare against TBU time base values (TBU_TSx greater than or equal to CMx) 1_B Less-equal compare against TBU time base values (TBU_TSx less than or equal to CMx)
EUPM	10	rw	**Extended update mode** *Note: If EUPM=1, a write access to CM0 or CM1 never causes an AEI write status 10_B.* *Note: This bit is only applicable in SOMC and SOMB mode.* 0_B No extended update of CM0 and CM1 via CPU or ARU 1_B Extended update mode in case of compare strategy 'serve last': update of CM1 after CCU0 compare match possible via ARU or CPU.
SL	11	rw	**Initial signal level** *Note: Reset value depends on the hardware configuration chosen by silicon vendor.* *Note: If the output is disabled, the output ATOM_OUT[x] is set to inverse SL, independent of the ATOM channel mode.* If FREEZE=0, following notes are valid: In SOMP, SOMI, SOMS mode, if the channel is disabled, the internal register SOUR inside ATOM sub-unit SOU is set to inverse value of SL. By enabling the channel, the register SOUR is not changed. Thus, if the output is enabled afterwards, the output ATOM_OUT[x] is the inverse value of SL. In SOMC mode, if the channel is disabled, the internal register SOUR inside ATOM sub-unit SOU is set to value of SL. By enabling the channel, the register SOUR is not changed. Thus, if the output is enabled and the channel is disabled, the output ATOM_OUT[x] is the value of SL. If FREEZE=1, the following notes are valid: If the channel is disabled, the output register of SOU unit is not changed, and output ATOM_OUT[x] is not changed. 0_B Low signal level 1_B High signal level

Field	Bits	Type	Description
CLK_SRC_SR	14:12	rw	**Actual CMU clock source (SOMS)/ shadow register for CMU clock source (SOMP)** If ECLK_SRC=0 / ECLK_SRC=1: *Note: This register is a shadow register for the register CLK_SRC. Thus, if the CMU_CLK source for PWM generation should be changed during operation, the old CMU_CLK has to operate until the update of the ATOM channels internal CLK_SRC register by the CLK_SRC_SR content is done, either by an end of a period or a forced update.* *Note: After (channel) reset, the selected CLK_SRC value is the SYS_CLK (input of Global Clock Divider). To use in SOMP mode one of the CMU_CLKx, it is recommended to perform a forced update of CLK_SRC with the value of CLK_SRC_SR value before/with enabling the channel.* *Note: In case of ECLK_SRC=1 and CLK_SRC_SR = 011_B/100_B/101_B/110_B, a force update leads to an immediate update of CM0, CM1 and CLK_SRC.* 000_B CMU_CLK0 selected / CMU_CLK0 selected 001_B CMU_CLK1 selected / CMU_CLK1 selected 010_B CMU_CLK2 selected / CMU_CLK2 selected 011_B CMU_CLK3 selected / reserved 100_B CMU_CLK4 selected / clock stopped 101_B CMU_CLK5 selected / TRIG[x-1] selected 110_B CMU_CLK6 selected / TIM_EXT_CAPTURE[x] selected 111_B CMU_CLK7 selected / CMU_CLK7 selected
ECLK_SRC	15	rw	**Extend CLK_SRC** *Note: This bit is only applicable in SOMP and SOMS mode.* See bit CLK_SRC_SR description for details. 0_B CLK_SRC_SR set 1 selected 1_B CLK_SRC_SR set 2 selected
WR_REQ	16	rw	**CPU Write request bit for late compare register update** *Note: The CPU can disable subsequent ARU read requests by the channel and can update the shadow registers with new compare values, while the compare units operate on old compare values received by former ARU accesses, if occurred.* *Note: On a compare match event, the WR_REQ bit will be reset by hardware.* *Note: At the point of the force update, only the shadow registers SR0 and SR1 are transferred into the CM0, CM1 registers. The output action is still defined by the ACBI bit field described by the ARU together with the old compare values for CM0/CM1.* *Note: This bit is only applicable in SOMC and SOMB mode.* 0_B No late update requested by CPU 1_B Late update requested by CPU

Field	Bits	Type	Description
TRIG_PULSE	17	rw	**Trigger output pulse length of one SYS_CLK period** 0_B Output on TOM[i]_OUT[x]_T is '1' as long as CN0=SR0 (if SR=_TRIG=1) 1_B Output on TOM[i]_OUT[x]_T is '1' for only one SYS_CLK period if CN0=SR0 (if SR=_TRIG=1)
UDMODE	19:18	rw	**Up/down counter mode** *Note: This mode is only applicable in SOMP mode.* 00_B Up/down counter mode disabled: CN0 counts always up 01_B Up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches 0 (i.e. changes from down to up) 10_B Up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches CM0 (i.e. changes from up to down) 11_B Up/down counter mode enabled: CN0 counts up and down, CM0,CM1 are updated if CN0 reaches 0 or CM0 (i.e. changes direction)
RST_CCU0	20	rw	**Reset source of CCU0** *Note: If RST_CCU0=1 and UPEN_CTRLx=1 are set, TRIG_[x-1] or TIM_EXT_CAPTURE(x) triggers also the update of work register (CM0, CM1 and CLK_SRC).* *Note: This bit is only applicable in SOMP mode.* *Note: This bit should only be set if bit OSM=0 (i.e. in continuous mode).* 0_B Reset counter register CN0 to 0 on matching comparison with CM0 1_B Reset counter register CN0 to 0 on trigger TRIG_[x-1] or TIM_EXT_CAPTURE(x)
OSM_TRIG	21	rw	**Enable trigger of one-shot pulse by trigger signal OSM_TRIG** *Note: This bit should only be set if bit OSM=1 and bit RST_CCU0=0.* 0_B Signal OSM_TRIG cannot trigger start of single pulse generation 1_B Signal OSM_TRIG can trigger start of single pulse generation (only if bit OSM = 1)
EXT_TRIG	22	rw	**Select TIM_EXT_CAPTURE(x) as trigger signal** 0_B Signal TRIG_[x-1] is selected as trigger to reset CN0 or to start single pulse generation 1_B Signal TIM_EXT_CAPTURE(x) is selected
EXTTRIGOUT	23	rw	**Select TIM_EXT_CAPTURE(x) as potential output signal TRIG_[x]** 0_B Signal TRIG_[x-1] is selected as output on TRIG_[x] (if TRIGOUT=0) 1_B Signal TIM_EXT_CAPTURE(x) is selected as output on TRIG_[x] (if TRIGOUT=0)

Field	Bits	Type	Description
TRIGOUT	24	rw	**Trigger output selection (output signal TRIG_CHx) of module ATOM_CHx** 0_B TRIG_[x] is TRIG_[x-1] or TIM _EXT_CAPTURE(x) 1_B TRIG_[x] is TRIG_CCU0
SLA	25	rw	**Serve last ARU communication strategy** *Note: This bit is only applicable in SOMC mode.* *Note: Please note, that setting of this bit has only effect, when ACBI(4:2) is configured for serve last compare strategy (100_B, 101_B, or 110_B).* *Note: When this bit is not set, the captured time stamps in the shadow registers SRx are only provided after the CCU1 match occurred. The ACBO(4:3) bits always return 10_B in that case.* *Note: By setting this bit, the ATOM channel also provides the captured time stamps after the CCU0 match event to the ARU. The ACBO(4:3) bits are set to 01_B in that case. After the CCU1 match event, the time stamps are captured again in the SRx registers and provided to the ARU. The ACBO(4:3) bits are set to 10_B. When the data in the shadow registers after the CCU0 match was not consumed by an ARU destination and the CCU1 match occurs, the data in the shadow registers is overwritten by the new captured time stamps. The ATOM channel does not request new data from the ARU when the CCU0 match values are read from an ARU destination.* 0_B Capture SRx time stamps after CCU0 match event not provided to ARU 1_B Capture SRx time stamps after CCU0 match event provided to ARU
OSM	26	rw	**One-shot mode** *Note: This bit is only applicable in SOMP and SOMS modes.* 0_B Continuous PWM generation after channel enable 1_B A single pulse is generated
ABM	27	rw	**ARU blocking mode** *Note: This bit is only applicable in SOMC and SOMB mode.* 0_B ARU blocking mode disabled: ATOM reads continuously from ARU and updates CM0, CM1 and ACB bits in case of SOMC mode, or SR0, SR1 and ACB bits in case of SOMB mode, independent of pending compare match event 1_B ARU blocking mode enabled: After update of CM0, CM1 and ACB bit in case of SOMC mode, or SR0, SR1 and ACB bits in case of SOMB mode via ARU, no new data is read via ARU until compare match event occurred, and in case of SOMC mode, SR0 and/or SR1 are read

Field	Bits	Type	Description
EXT_FUPD	29	rw	**External forced update** *Note: This bit is only applicable in SOMP and SOMS mode.* 0_B Use FUPD(x) signal from AGC to force update 1_B Use TIM_EXT_CAPTURE signal to force update
SOMB	30	rw	**SOMB mode** 0_B ATOM channel mode defined by bit field MODE 1_B ATOM SOMB mode enabled
FREEZE	31	rw	**FREEZE** *Note: if channel is disabled and ouptut is enabled, in SOMP mode with UDMODE!=0b00 the output is dependng directly on SL bit, independent on FREEZE mode.* 0_B A channel disable/enable may change internal register and output register 1_B A channel enable/disable does not change an internal or output register but stops counter CN0 (in SOMP mode), comparison (in SOMC/SOMB mode) and shifting (in SOMS mode)
0	28	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.10Register ATOM[i]_CH[x]_STAT

ATOMi Channel x Status Register

ATOMi_CHx_STAT (i=0-11;x=0-7)
ATOMi Channel x Status Register **($0E801C_H$+i*800_H+x*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**ACBO**					**DR**	**WRF**	**DV**	**ACBI**				
r			r					r	rw	r	r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**OL**
r															r

Field	Bits	Type	Description
OL	0	r	**Actual output signal level of ATOM_CHx_OUT** *Note: Reset value is the inverted value of bit SL, which depends on the hardware configuration chosen by silicon vendor.* 0_B Actual output signal level is low 1_B Actual output signal level is high

Field	Bits	Type	Description
ACBI	20:16	r	**ATOM Mode control bits** *Note: For ATOM SOMI, SOMC, SOMP and SOMS mode, this register serves as a mirror for the five ARU control bits received through the ARU interface. The bits are valid when the DV bit is set.* *Note: For SOMB mode, this bit field serves as the work register of the compare strategy. It can be updated with the value of bit field ACB of register ATOM[i]_CH[x]_CTRL, or the value of internal shadow register ACB_SR.*
DV	21	r	**Valid ARU Data stored in compare registers** *Note: This bit is only applicable in SOMC and SOMB mode. The CPU can determine the status of the ARU transfers with this bit. After the compare event occurred, the bit is reset by hardware.* 0_B No valid data stored in register CM0 and/or CM1, no comparison is activated 1_B Valid data stored in CM0 and/or CM1, comparison activated
WRF	22	rw	**Write request of CPU failed for late update** *Note: This bit is only applicable in SOMC and SOMB mode.* Bit WRF can be reset by writing a 1 to it. 0_B Late update was successful, CCUx units wait for comparison 1_B Late update failed
DR	23	r	**ARU data rejected flag** *Note: The flag is cleared if valid data is received and stored via ARU.* 0_B Received ARU data stored 1_B Received ARU data rejected
ACBO	28:24	r	**ATOM Internal status bits** ACBO[3] = 1: CCU0 Compare match occurred. ACBO[4] = 1: CCU1 Compare match occurred. *Note: These bits are only set in SOMC mode.* *Note: ACBO is reset to 0b00000 on an update of register CM0 or CM1 (via ARU or CPU).* *Note: In SOMC mode, these bits are sent as ARU control bits 52..48.*
0	15:1, 31:29	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.11Register ATOM[i]_CH[x]_RDADDR

ATOMi Channel x ARU read address Register

ATOMi_CHx_RDADDR (i=0-11;x=0-7)

ATOMi Channel x ARU read address Register($0E8000_H$+i*800_H+x*80_H) Application Reset Value: 01FE $01FE_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							RDADDR1								
r							rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							RDADDR0								
r							rw								

Field	Bits	Type	Description
RDADDR0	8:0	rw	**ARU Read address 0** *Note: This read address is used by the ATOM channel to receive data from ARU immediately after the channel and ARU access is enabled (see ATOM[i]_CH[x]_CTRL register for details).* *Note: This bit field is only writable if channel is disabled.*
RDADDR1	24:16	rw	**ARU Read address 1** *Note: The ATOM channel switches to this read address, when requested in the ARU control bits 52 to 48 with the pattern "111--". The channel switches back to the RDADDR0 after one ARU data package was received on RDADDR1 and the compare match event is occurred.* *Note: This read address is only applicable in SOMC mode.* *Note: This bit field is only writable if channel is disabled.*
0	15:9, 31:25	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.12Register ATOM[i]_CH[x]_CN0

ATOMi Channel x CCU0 Counter Register

ATOMi_CHx_CN0 (i=0-11;x=0-7)

ATOMi Channel x CCU0 Counter Register($0E8018_H$+i*800_H+x*80_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								CN0							
r								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CN0															
rw															

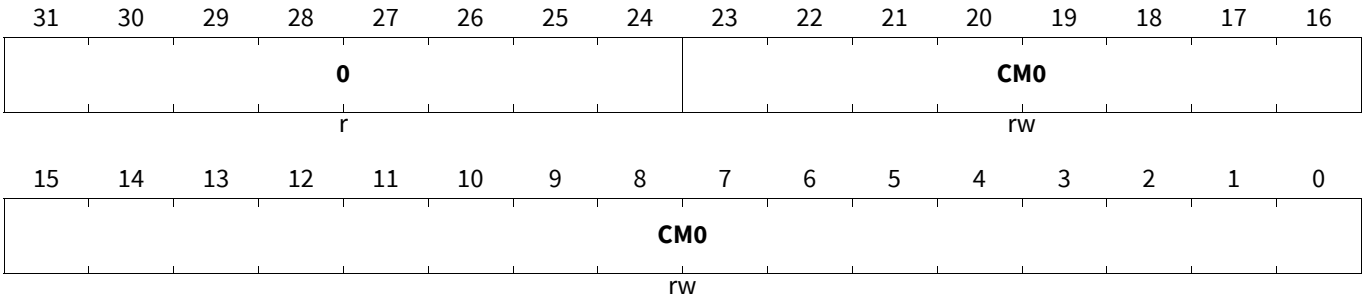
Field	Bits	Type	Description
CN0	23:0	rw	**ATOM CCU0 counter register**
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.13Register ATOM[i]_CH[x]_CM0

ATOMi Channel x CCU0 Compare Register

Note: This register is write protected in SOMC mode and returns AEI_STATUS=0b10 on write access, when in serve last compare strategy the first match of CCU0 occurred.

ATOMi_CHx_CM0 (i=0-11;x=0-7)
ATOMi Channel x CCU0 Compare Register($0E8010_H$+i*800_H+x*80_H) Application Reset Value: 0000 0000_H

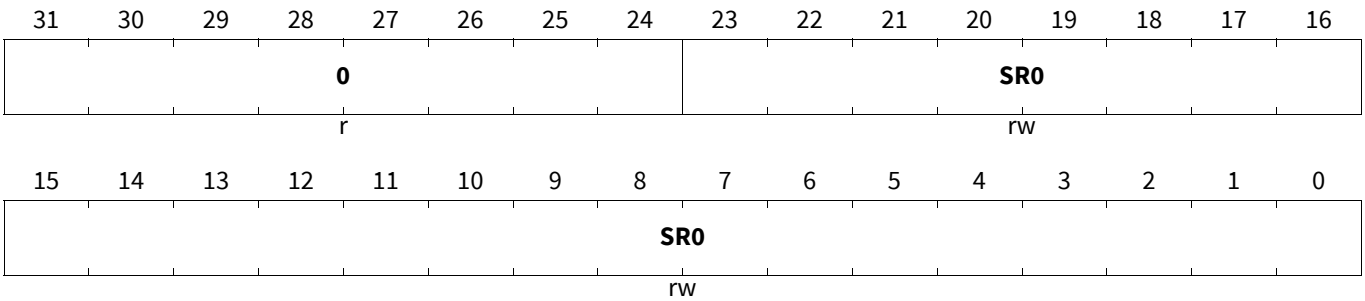


Field	Bits	Type	Description
CM0	23:0	rw	**ATOM CCU0 compare register**
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.14Register ATOM[i]_CH[x]_SR0

ATOMi Channel x CCU0 Compare Shadow Register

ATOMi_CHx_SR0 (i=0-11;x=0-7)
ATOMi Channel x CCU0 Compare Shadow Register($0E8008_H$+i*800_H+x*80_H) Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
SR0	23:0	rw	**ATOM channel x shadow register SR0** *Note: The SR0 register is used as shadow register for CM0 in SOMP and SOMS modes, and is used as capture register for time base TBU_TS0 in SOMC mode.*
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

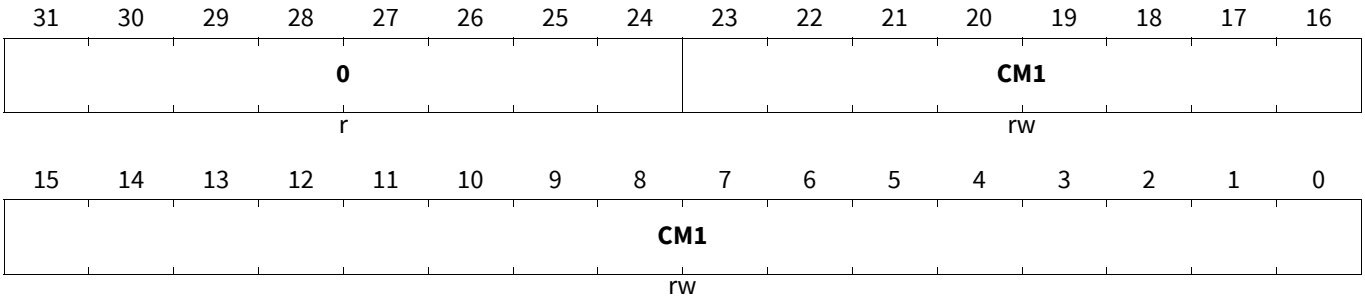
28.15.6.15Register ATOM[i]_CH[x]_CM1

ATOMi Channel x CCU1 Compare Register

Note: This register is write protected in SOMC mode and returns AEI_STATUS=0b10 on write access, when in serve last compare strategy the first match of CCU0 occurred.

ATOMi_CHx_CM1 (i=0-11;x=0-7)

ATOMi Channel x CCU1 Compare Register($0E8014_H$+i*800_H+x*80_H) **Application Reset Value: $0000\ 0000_H$**

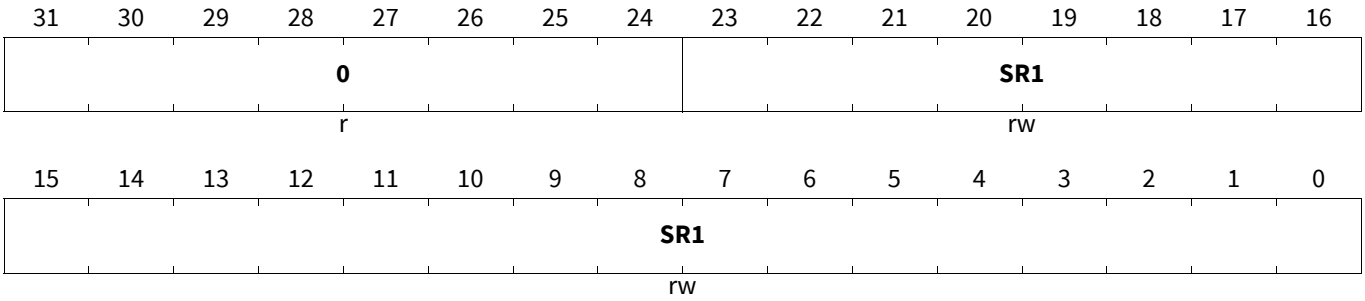


Field	Bits	Type	Description
CM1	23:0	rw	**ATOM CCU1 compare register**
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.16Register ATOM[i]_CH[x]_SR1

ATOMi Channel x CCU1 Compare Shadow Register

ATOMi_CHx_SR1 (i=0-11;x=0-7)
ATOMi Channel x CCU1 Compare Shadow Register($0E800C_H$+i*800_H+x*80_H) Application Reset Value: 0000 0000_H

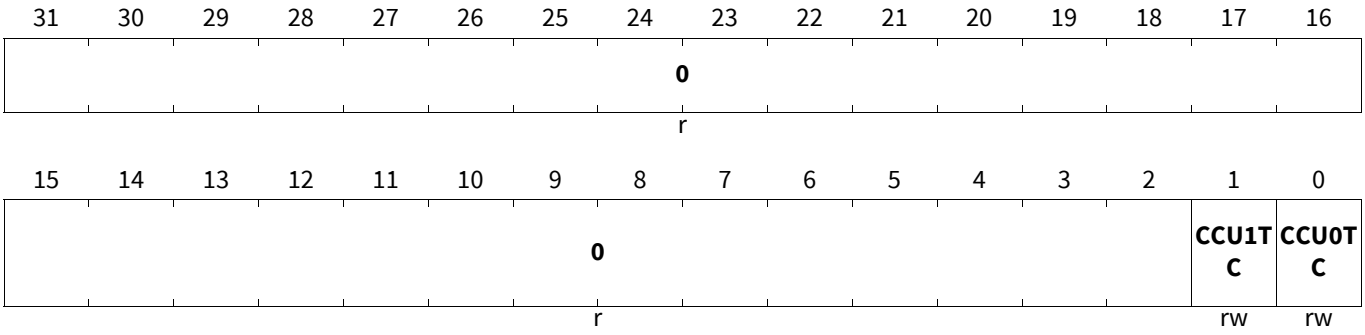


Field	Bits	Type	Description
SR1	23:0	rw	**ATOM channel x shadow register SR1** *Note: The SR1 register is used as shadow register for CM1 in SOMP and SOMS modes, and is used as capture register for time base TBU_TS1 or TBU_TS2 (when selected in ATOM[i]_CH[x]_CTRL register) in SOMC mode.*
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.17Register ATOM[i]_CH[x]_IRQ_NOTIFY

ATOMi Channel x Interrupt Notification Register

ATOMi_CHx_IRQ_NOTIFY (i=0-11;x=0-7)
ATOMi Channel x Interrupt Notification Register($0E8020_H$+i*800_H+x*80_H) Application Reset Value: 0000 0000_H

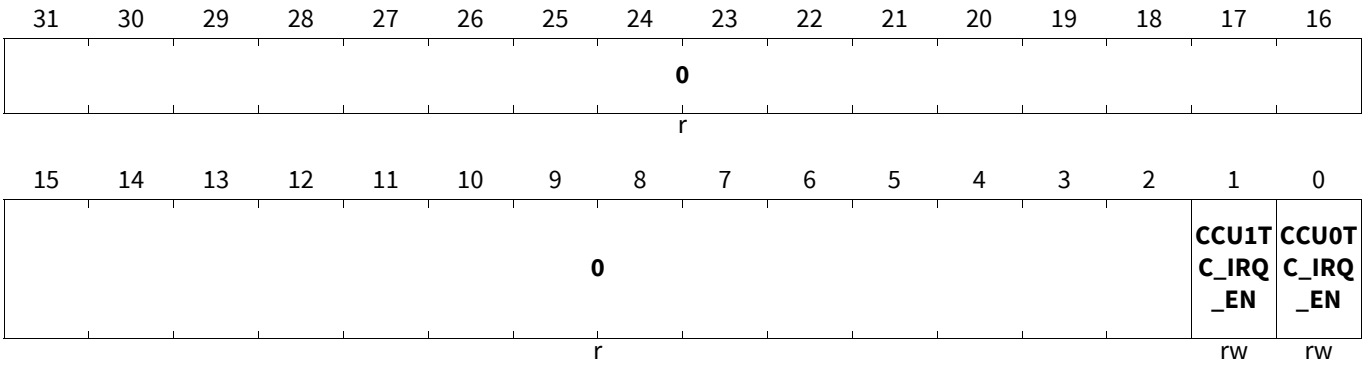


Field	Bits	Type	Description
CCU0TC	0	rw	**CCU0 Trigger condition interrupt for channel x** The notification of the interrupt is only triggered one time after reaching the condition CN0 >= CM0. To enable re-trigger of the notification, first the condition CN0 < CM1 has to be reached. *Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.* 0_B No interrupt occurred 1_B CCU0 Trigger condition interrupt was raised by ATOM channel x
CCU1TC	1	rw	**CCU1TC: CCU1 Trigger condition interrupt for channel x** The notification of the interrupt is only triggered one time after reaching the condition CN0 >= CM1. To enable re-trigger of the notification, first the condition CN0 < CM1 has to be reached. *Note: This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.* *Note: If bit SR0_TRIG is set to 1 (only valid in SOMP mode), this interrupt notify flag is set in case of SR0 is equal to CN0 and not set in case of CM1 >=/<= CN0.*
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.18Register ATOM[i]_CH[x]_IRQ_EN

ATOMi Channel x Interrupt Enable Register

ATOMi_CHx_IRQ_EN (i=0-11;x=0-7)
ATOMi Channel x Interrupt Enable Register($0E8024_H$+i*800_H+x*80_H) Application Reset Value: 0000 0000_H

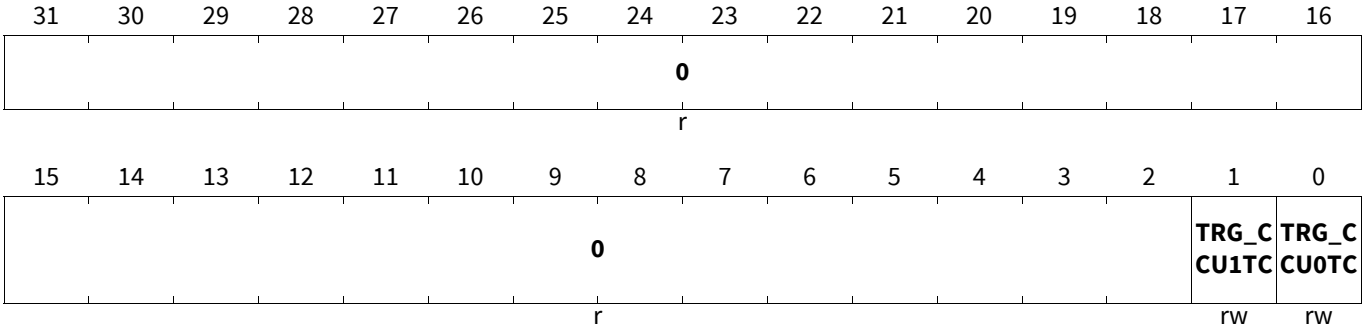


Field	Bits	Type	Description
CCU0TC_IRQ_EN	0	rw	**ATOM_CCU0TC_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
CCU1TC_IRQ_EN	1	rw	**ATOM_CCU1TC_IRQ interrupt enable** See bit 0.
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.19Register ATOM[i]_CH[x]_IRQ_FORCINT

ATOMi Channel x Software Interrupt Generation Register

ATOMi_CHx_IRQ_FORCINT (i=0-11;x=0-7)
ATOMi Channel x Software Interrupt Generation Register($0E8028_H$+i*800_H+x*80_H) Application Reset Value: $0000\ 0000_H$

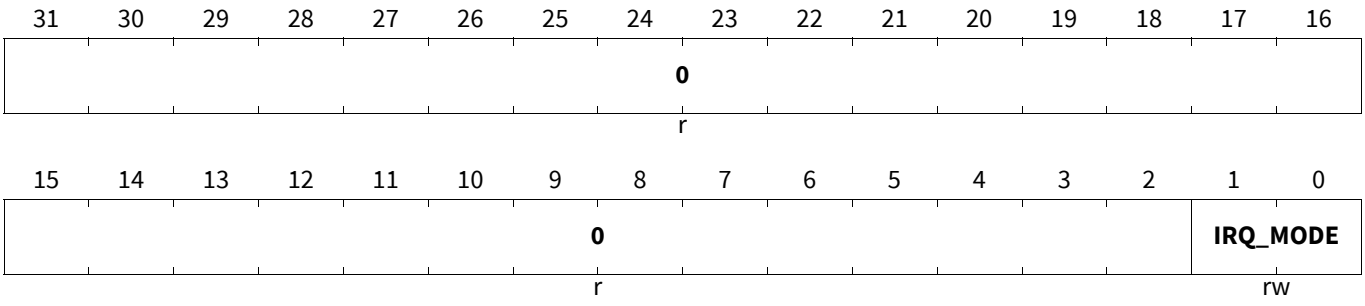


Field	Bits	Type	Description
TRG_CCU0TC	0	rw	**Trigger ATOM_CCU0TC_IRQ interrupt by software** *Note: This bit is cleared automatically after write.* *Note: This bit is write protected by bit RF_PROT of register GTM_CTRL* 0_B No interrupt triggering 1_B Assert CCU0TC_IRQ interrupt for one clock cycle
TRG_CCU1TC	1	rw	**Trigger ATOM_CCU1TC_IRQ interrupt by software** *Note: This bit is cleared automatically after write.* *Note: This bit is write protected by bit RF_PROT of register GTM_CTRL* 0_B No interrupt triggering 1_B Assert CCU1TC_IRQ interrupt for one clock cycle
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.15.6.20Register ATOM[i]_CH[x]_IRQ_MODE

ATOMi Channel x Interrupt Mode Configuration Register

ATOMi_CHx_IRQ_MODE (i=0-11;x=0-7)
ATOMi Channel x Interrupt Mode Configuration Register($0E802C_H$+i*800_H+x*80_H) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** *Note: The interrupt modes are described in **Section 28.4.5**.* 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero

28.16 Dead Time Module (DTM)

28.16.1 Overview

Figure 93 gives an overview of the structure of the Dead Time Module (DTM).

Note: In this paragraph, the following is used. Variable n is used for CDTMs, x for channels and i is used for DTMs. (An exception for TIM names.)

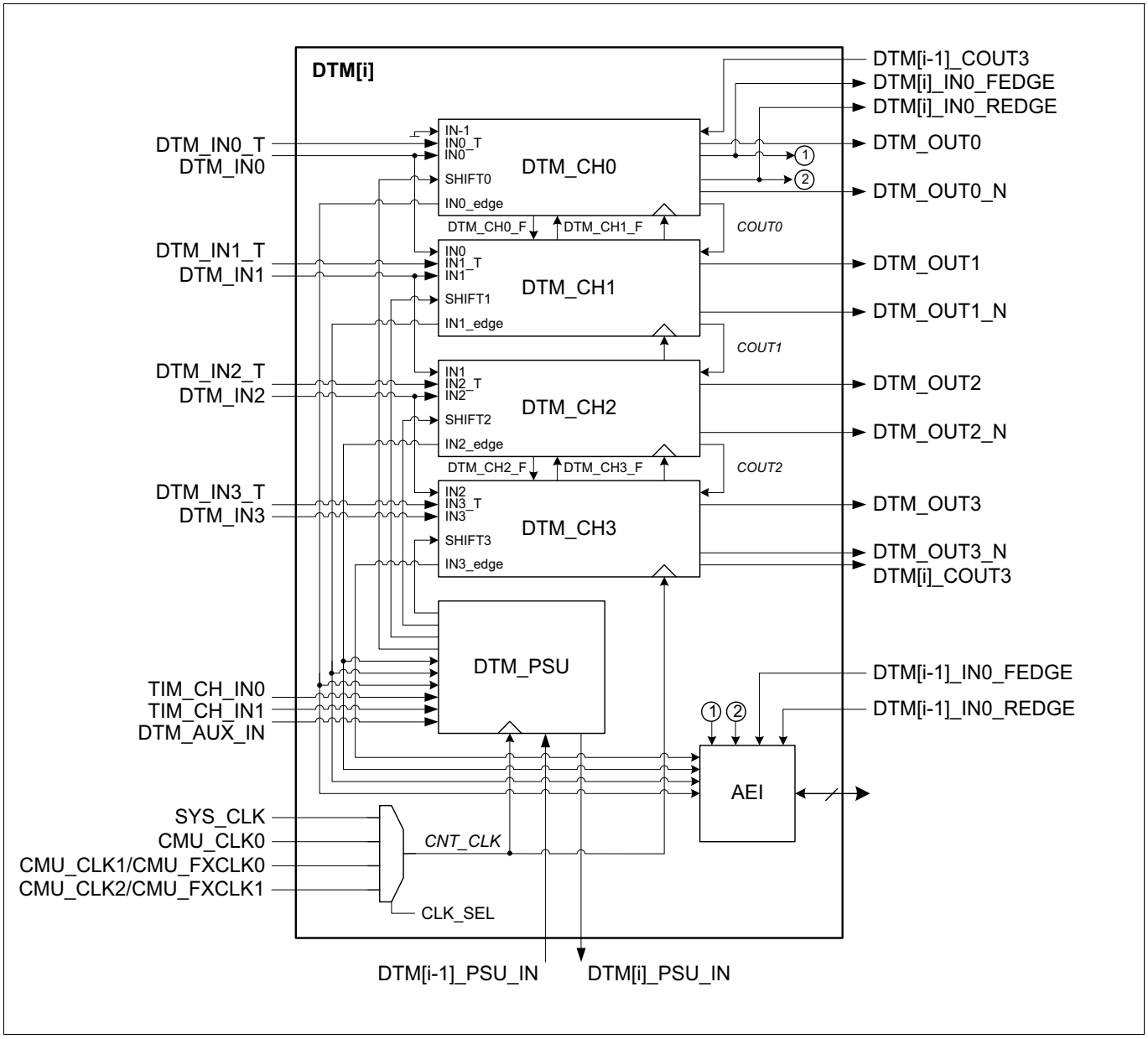


Figure 93 DTM overview

The main function of the DTM is to derive for each input *DTM_IN0* to *DTM_IN3* the individual inverse signal (*DTM[i]_OUT[x]_N*) and to apply an edge specific delay between the edge of the original signal and the edge of the derived inverted signal (i.e., the dead time). This function is mainly used for controlling of half bridges.

A second function provided by DTM is to set the outputs of one channel to the value of the preceding channel if requested by a trigger on input *TIM_CH_IN*0, *TIM_CH_IN1* or *DTM_AUX_IN*. This feature allows a phase shift on one PWM signal to the phase of the preceding PWM signal up to the next edge on this channel.

The third function provided by DTM is to (N)AND/(N)OR/X(N)OR combine the input *DTM_IN[x]* signal of one DTM channel with the signal on input *TIM_CH_IN0, TIM_CH_IN1* or *DTM_AUX_IN* (selected inside DTM_PSU and assigned to one of the signals *SHIFT[x]*) or with the combinational output (signal *COUT[x]*) of preceding channel.As a result *COUT2* may be the combined signal of *DTM_IN0* and *TIM_CH_IN0, TIM_CH_IN1* or *DTM_AUX_IN* and the signal *DTM_IN1*. For *COUT3* this chain can be combined again with signal *DTM_IN3*.

The outputs of each channel may be swapped individually to provide the function of combining signals on each output of a channel.

In general, the DTM instances are placed behind the TOM and the ATOM instances, i.e., the outputs *TOM_OUT[x] and TOM_OUT[x]_T* or *ATOM_OUT[x] and ATOM_OUT[x]_T* are each routed to the DTM instance inputs *DTM_IN[y]* and *DTM_IN[y]_T*.Four DTM instances behind a TOM instance i and two DTM instances behind an ATOM instance i are grouped together in a Cluster DTM hierarchy called CDTM[i]. The connections between DTM and the modules TOM and ATOM are depicted in **Figure 94**.

Note, depending on device configuration, not every DTM instance is available. E.g. a device may only have one DTM connected to the first four channels of ATOM. In this case, the other four channels (4 to 7) are connected directly to GTM outputs.For detailed information, which DTM instance is available, refer to corresponding device specific device specific appendix of this specification.

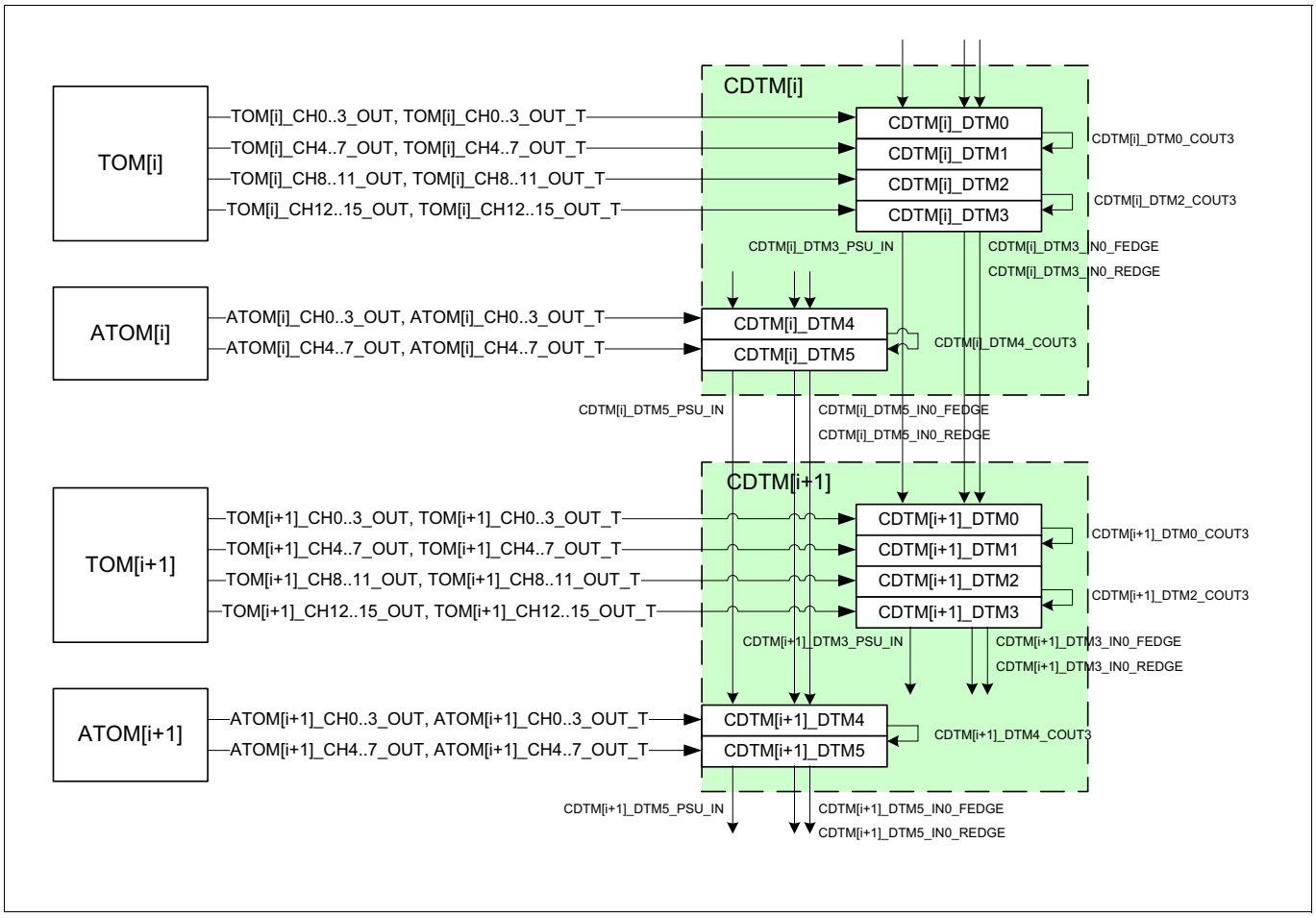


Figure 94 Connections of TOM and ATOM to DTM inputs DTM_IN[y]/DTM_IN[y]_T

Additionally, the DTM instances have inputs *TIM_CH_IN0*/*TIM_CH_IN1* which are driven by TIM output signals *TIM[i]_CH[x]_F_OUT*.

There are two configurations of TIM to DTM connections possible depending on the DTM channel specific configuration bit TIM_SEL.

In case of TIM_SEL=0 the connected TIM input may not be of the same cluster as the DTM.

In case of TIM_SEL=1 the TIM input is of the same cluster as the DTM.

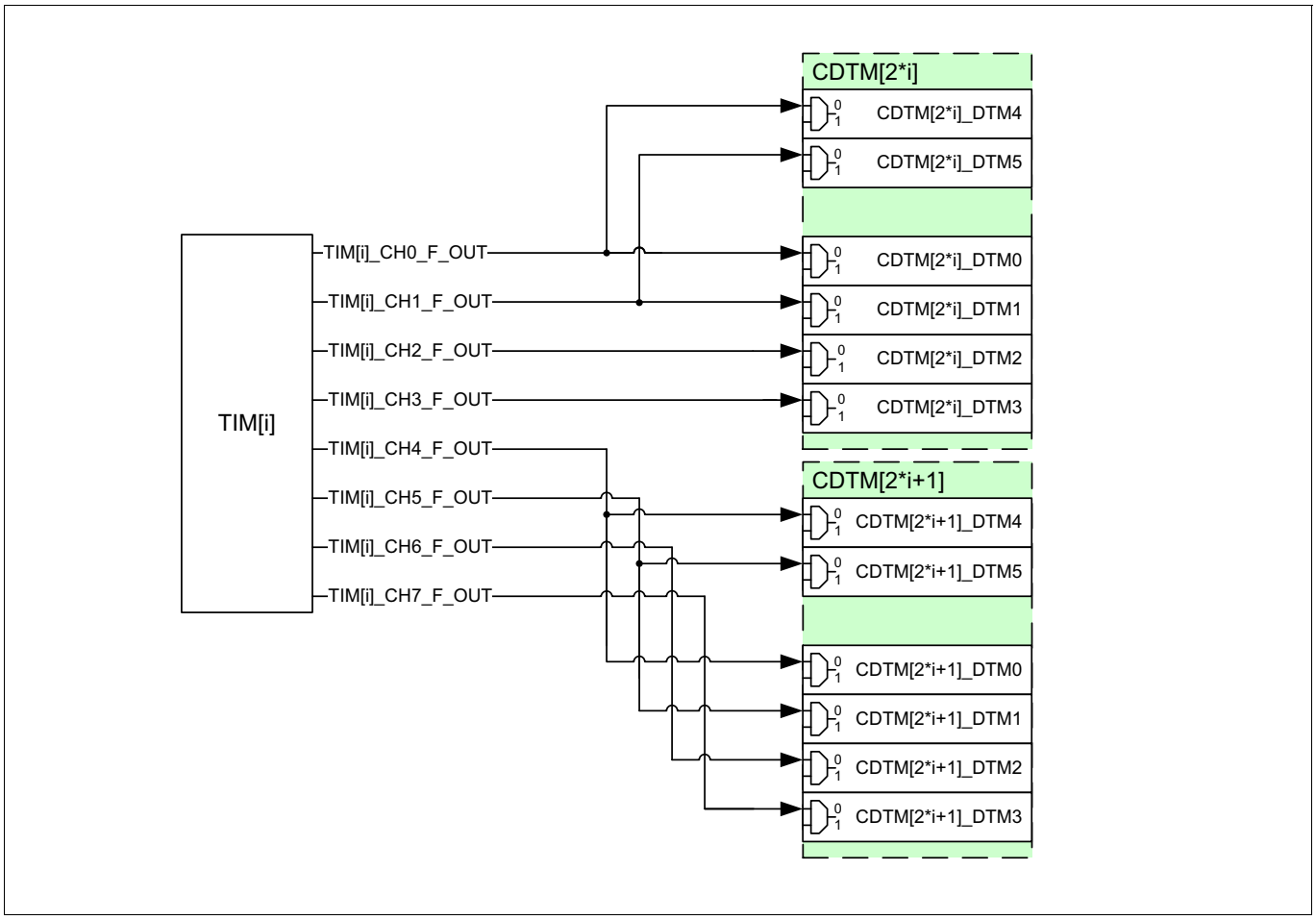


Figure 95 Connections of TIM to DTM inputs TIM_CH_IN0/TIM_CH_IN1 for TIM_SEL=0

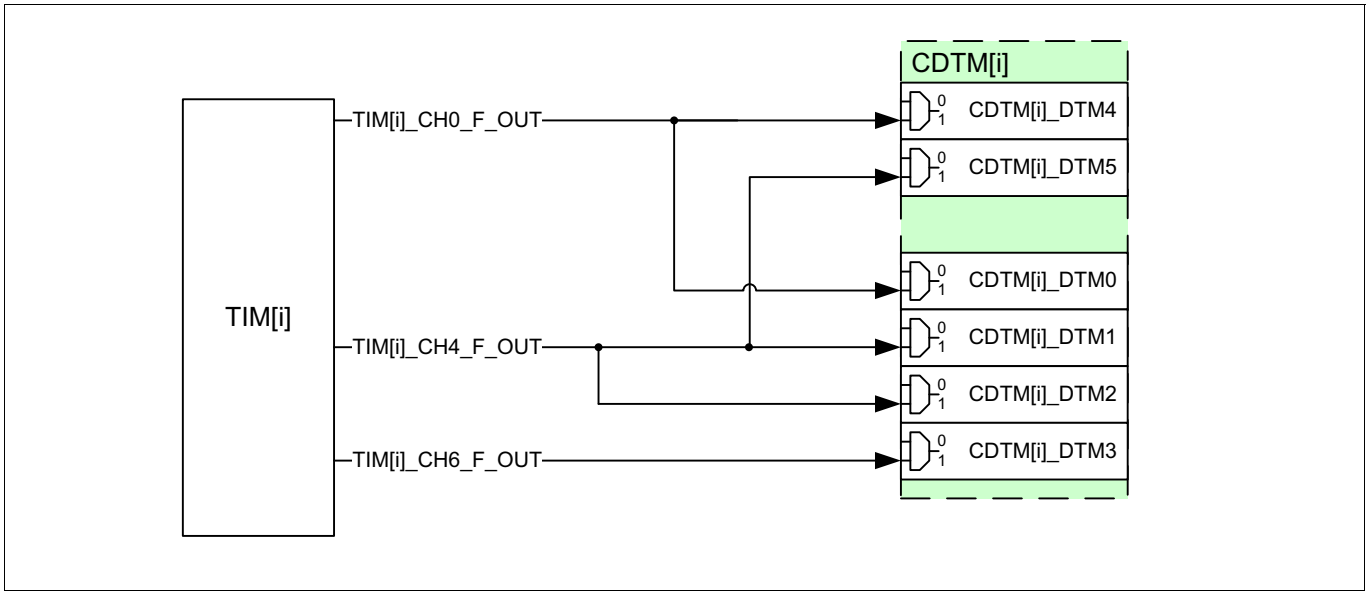


Figure 96 Connections of TIM to DTM inputs TIM_CH_IN0/TIM_CH_IN1 for TIM_SEL=1

There are also connections between DTM instances of same CDTM instance.For each pair of DTM instance 2i and 2i+1 the combinatorial output *COUT(3)* of DTM[2i] channel 3 is connected to DTM[2i+1] channel 0 *COUT(0-1)*.

With this a combinatorial chain over two neighbored DTM instances can be configured.

If one of the two neighbored DTM instances is not available (i.e. it is an empty instance) the inputs used for connections between two neighbored DTM instances are left open.

Note: For channel x=0 of DTM instance 2i input signals COUT[x-1] is unused and I1SEL[x] is defined as 0.

An additional link between DTM instances behind an ATOM is a forwarding of *DTM[i]_PSU_IN* signal to next available instance of DTM behind an ATOM (e.g. *DTM[i+1]_PSU_IN*).

The same link is available between all available DTM behind a TOM.

Note, for unavailable DTM[i] instances (i.e. the instance DTM[i] is called empty) the signal *DTM[i-1]_PSU_IN* is passed through empty instance DTM[i] to *DTM[i+1]_PSU_IN*, *DTM[i-1]_IN0_FEDGE* and *DTM[i-1]_IN0_REDGE* are passed through DTM[i] to *DTM[i+1]_IN0_FEDGE* and *DTM[i+1]_IN0_REDGE*.

Further connections between neighbored DTM instances are depicted in **Figure 97**:

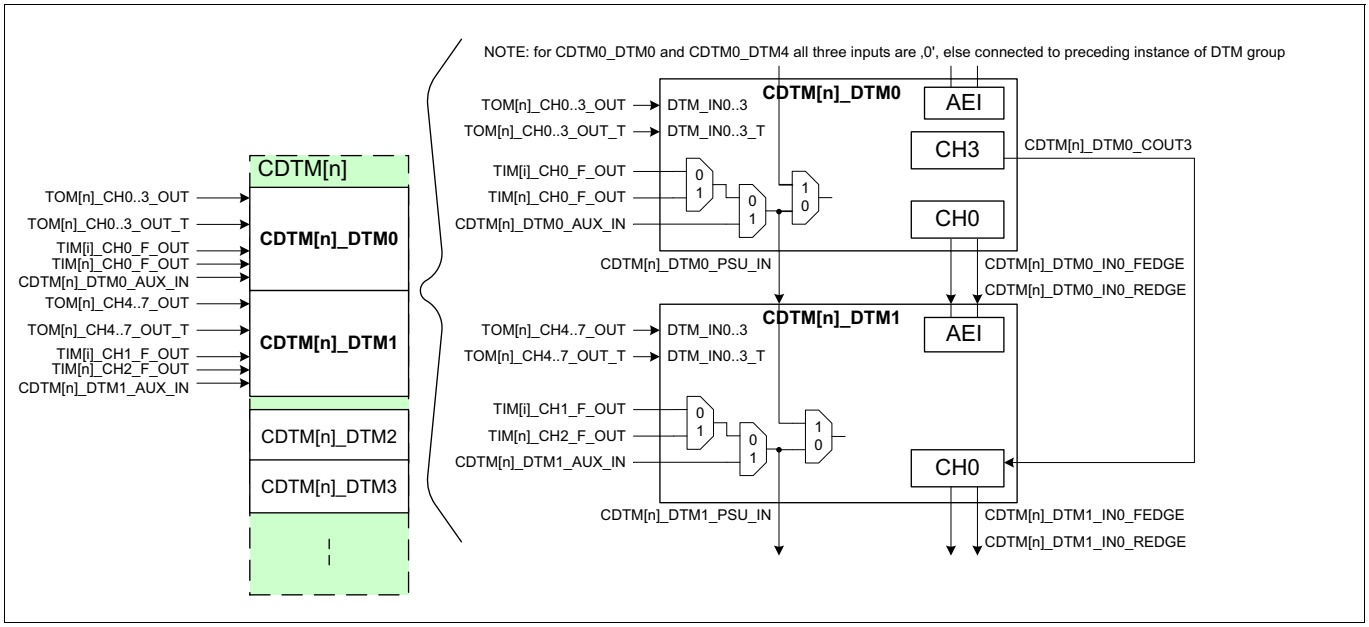


Figure 97 Connections between DTM instances

28.16.2 DTM Channel

Figure 98 depicts the functions of a DTM channel.

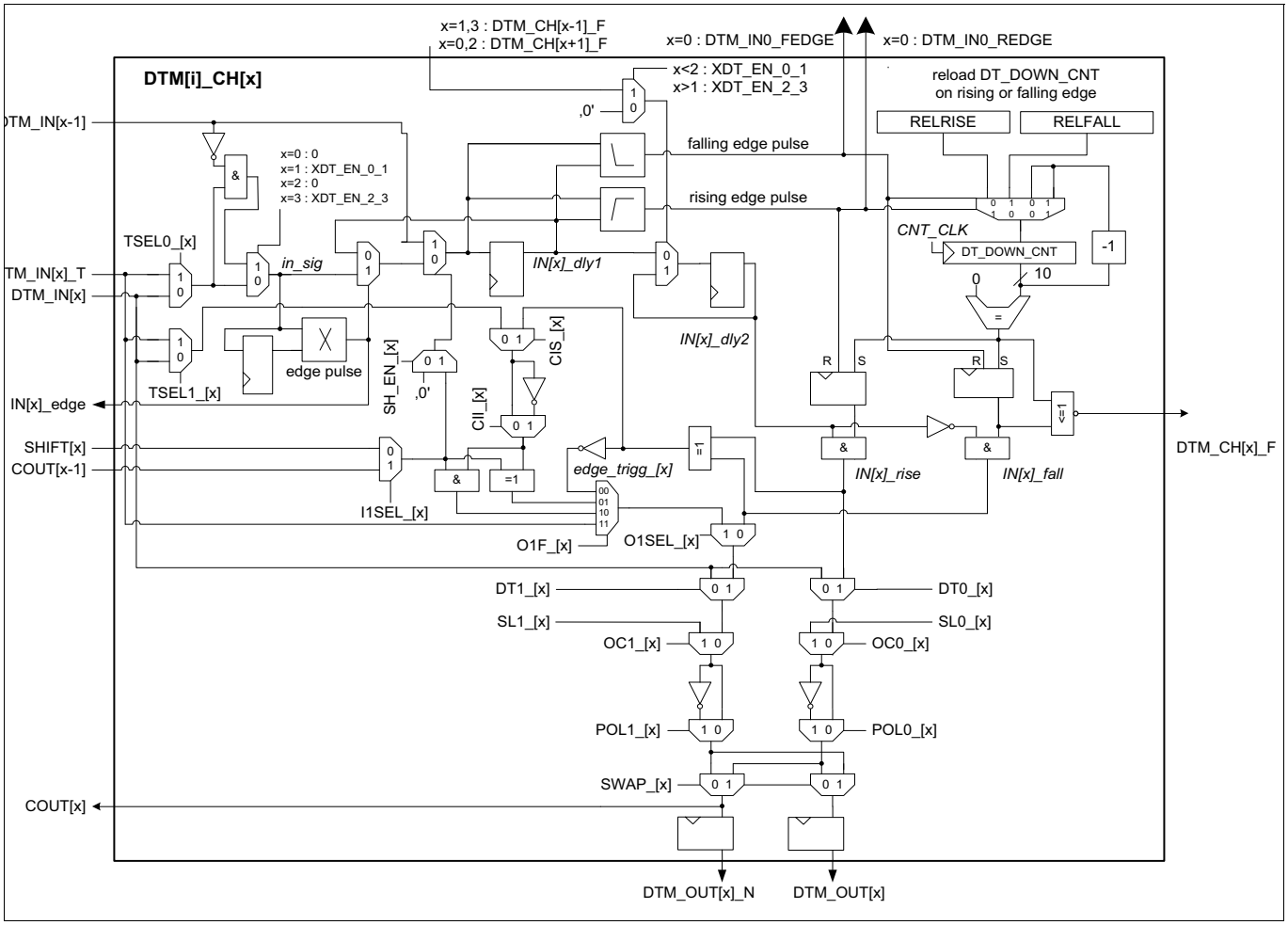


Figure 98 DTM channel overview

The main feature of each channel is to derive the inverse signal out of the input signal *DTM_IN[x]*, apply an edge dependent delay on the two resulting signal paths and provide these signals at the outputs *DTM[i]_OUT[x]* and *DTM[i]_OUT[x]_N*.

There are two possibilities to apply dead time on GTM output signals.One is to use one DTM channel per TOM/ATOM channel and generate inside the DTM the second inverse signal. This is called the standard dead time generation.The second way is to generate two signals out of two TOM/ATOM channel and to apply inside the DTM only the dead time by using two cross linked DTM channel. This is called the cross dead time generation.

28.16.2.1 Standard dead time generation

The dead time can be configured for each edge individually. The bit field **RELRISE** in register **DTM[i]_CH[x]_DTV** contains the reload value for the counter and defines the delay for rising edges in multiples of selected clock ticks.The bit field **RELFALL** in register **DTM[i]_CH[x]_DTV** contains the reload value for the counter and defines the delay for falling edges in multiples of selected clock ticks.

The counter is reloaded with the value of **RELRISE** on a rising edge and reloaded with the value of **RELFALL** on a falling edge on input *DTM_IN[x]* (or *DTM_IN[x-1]* in case of shift enable **SH_EN_[x]**).

On a reload of the counter the flip-flop following the counter output comparator is reset and stays reset until the counter has reached 0.After reload, the counter **DT_DOWN_CNT** counts down until it reaches 0 and stops at 0.

The signal flow for function of standard dead time signal generation is depicted in **Figure 99**.

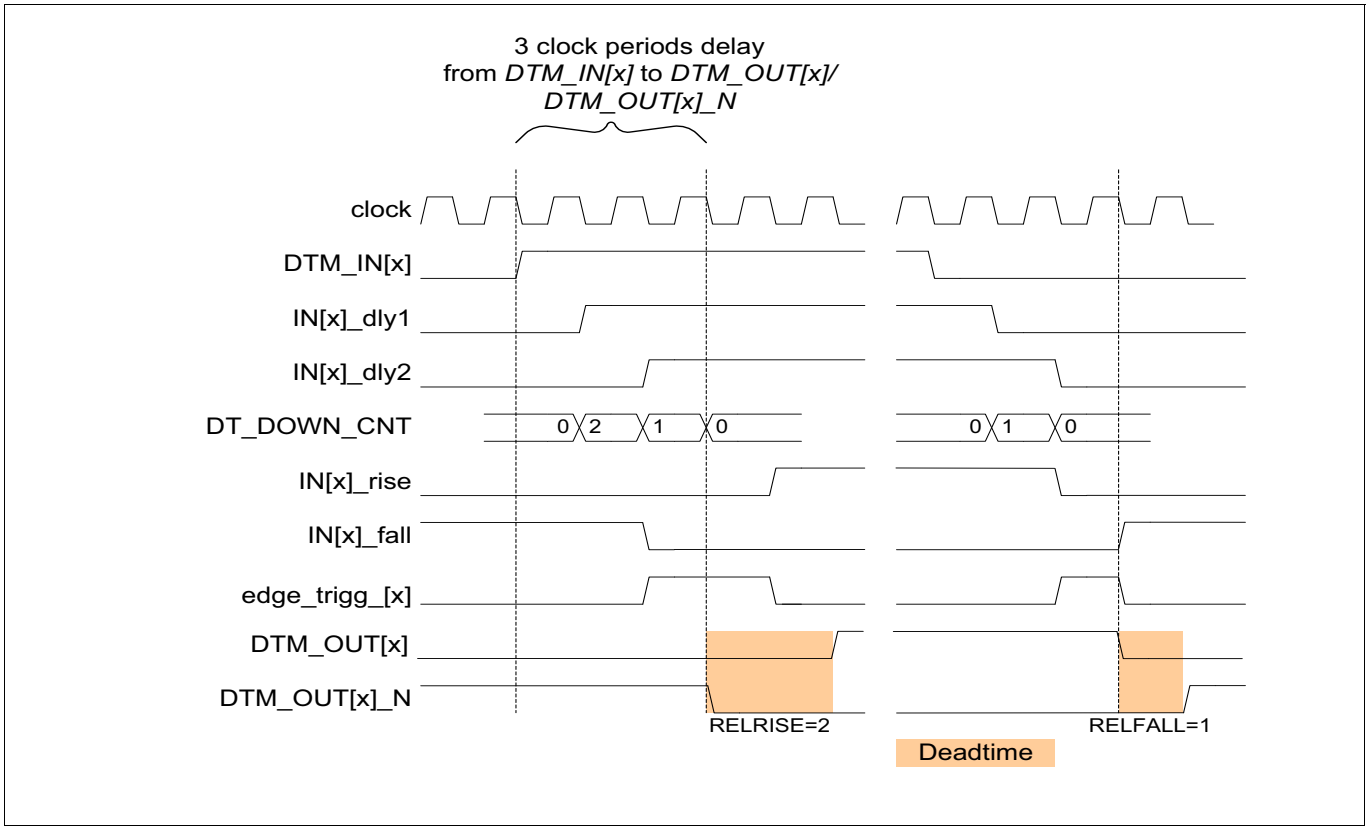


Figure 99 Wave signals for function of dead time generation

*Note: The delay from the input signal DTM_IN[x] to the output signals DTM[i]_OUT[x] and DTM[i]_OUT[x]_N is three system clock periods by disabled feed through (see **DT0/1_[x]** in **DTM[i]_CH_CTRL2**).*

*Note: The delay from the input signal DTM_IN[x] to the output signals DTM[i]_OUT[x] and DTM[i]_OUT[x]_N is one system clock periods by enabled feed through (see **DT0/1_[x]** in **DTM[i]_CH_CTRL2**).*

*Note: The delay from the input signal DTM_IN[x]_T to the output signals DTM[i]_OUT[x] and DTM[i]_OUT[x]_N is three system clock periods in case of disabled feed through (see **O1F_[x]** and **O1SEL_[x]** in **DTM[i]_CH_CTRL2**).*

*Note: The delay from the input signal DTM_IN[x]_T to the output signals DTM[i]_OUT[x] and DTM[i]_OUT[x]_N is one system clock periods in case of enabled feed through (see **O1F_[x]** and **O1SEL_[x]** in **DTM[i]_CH_CTRL2**).*

Note: The reset level of the output signals DTM[i]_OUT[x] connected from ATOM module depends on the hardware configuration value atom_out_reset_level_c chosen by silicon vendor.

Note: The reset level of the output signals DTM[i]_OUT[x]_N connected from ATOM module is defined by the inverse hardware configuration value atom_out_reset_level_c chosen by silicon vendor.

Note: The reset level of the output signals DTM[i]_OUT[x] connected from TOM module is defined by the hardware configuration value tom_out_reset_level_c chosen by silicon vendor.

Note: The reset level of the output signals DTM[i]_OUT[x]_N connected from TOM module is defined by the inverse hardware configuration value tom_out_reset_level_c chosen by silicon vendor.

28.16.2.2 Cross channel dead time

A second way to apply a dead time value on two output signals is the cross channel dead time.

In opposite to the dead time described in **Section 28.16.2.1** the cross channel dead time mode does not generate out of one signal the corresponding inverse signal but tries to apply the dead time on the input signals of two neighbored DTM channel.

To do this, two neighbored DTM input signals (on DTM channel (2k) and (2k+1) for k=0,1) are cross linked together in the way that a falling edge on one channel leads to a hold phase of current signal value on the cross linked channel.

This behavior is reached by the following:

A falling edge on e.g. channel (2k) reloads the **DT_DOWN_CNT** with the value of **RELFALL**. While this counter is counting down, the output signal of the cross linked channel (2k+1) keeps its value. If the counter **DT_DOWN_CNT** has reached 0 again, the channel (2k+1) output is released and can follow the value on its input.The timing of the cross channel dead time is depicted in the following figure:

Figure 100 shows the behavior in case of input edges at *DTM_IN[2k]* and *DTM_IN[2k+1]* occur at the same point in time.Then the falling edge is forwarded immediately (with only two clock cycles delay) and the rising edge is delayed additionally by the number of clock ticks specified by the **RELFALL** parameter of the cross linked channel.

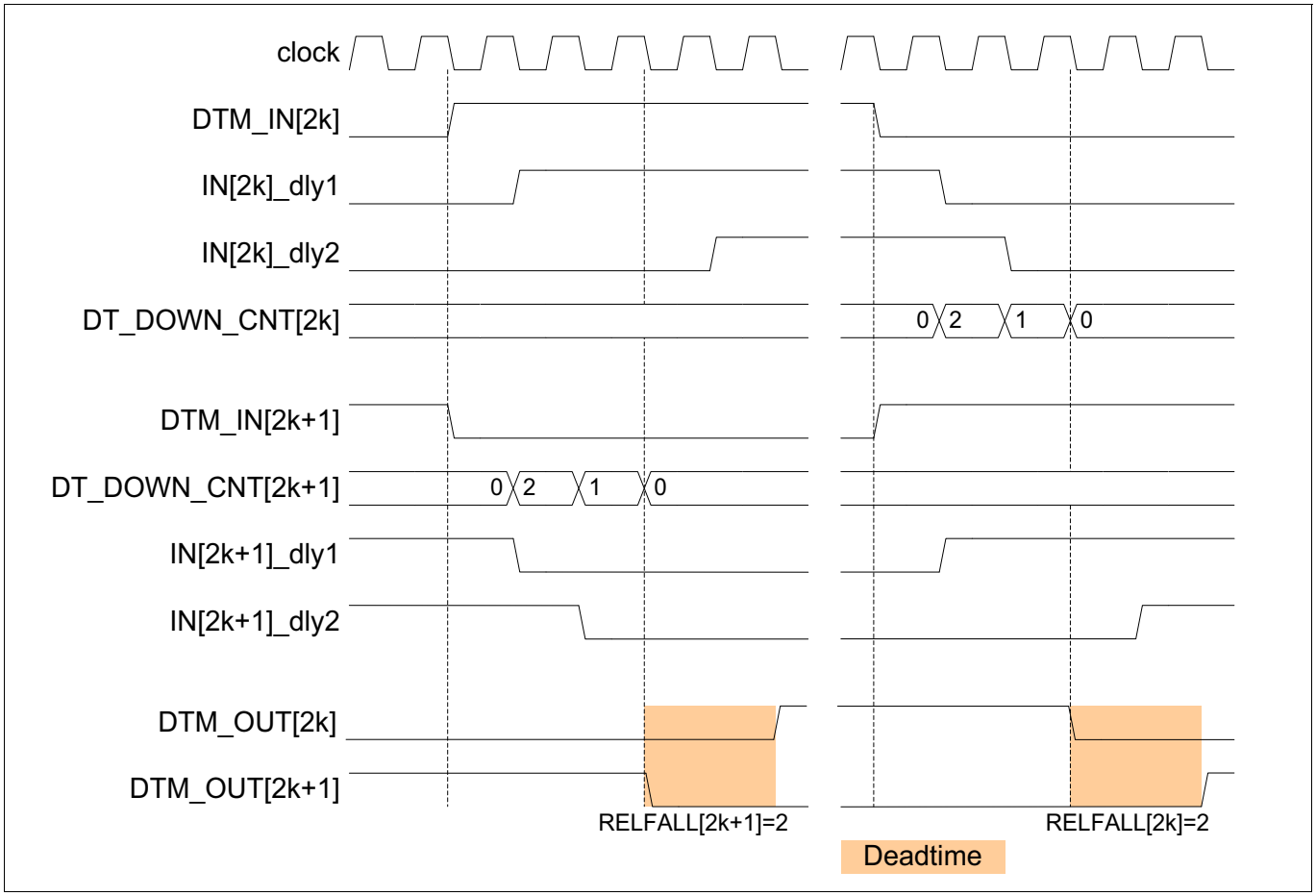


Figure 100 Cross channel dead time timing diagram

In case of high level (i.e. 1) at the DTM inputs *DTM_IN[2k]* and *DTM_IN[2k+1]* at the same point in time, the channel of (2k) has higher priority than the corresponding channel (2k+1). This means that in this case the input *DTM_IN[2k+1]* if forced immediately at channel input to low level (i.e. 0).

As a result the DTM output of channel *DTM_OUT[2k+1]* can never be high if the cross linked channel *DTM_OUT[2k]* is high.

28.16.3 Phase Shift Control Unit

The phase shift unit (DTM_PSU) is depicted in the following figure. It supports the second major function of the DTM module to allow phase shifting of PWM signal on one of the channels.

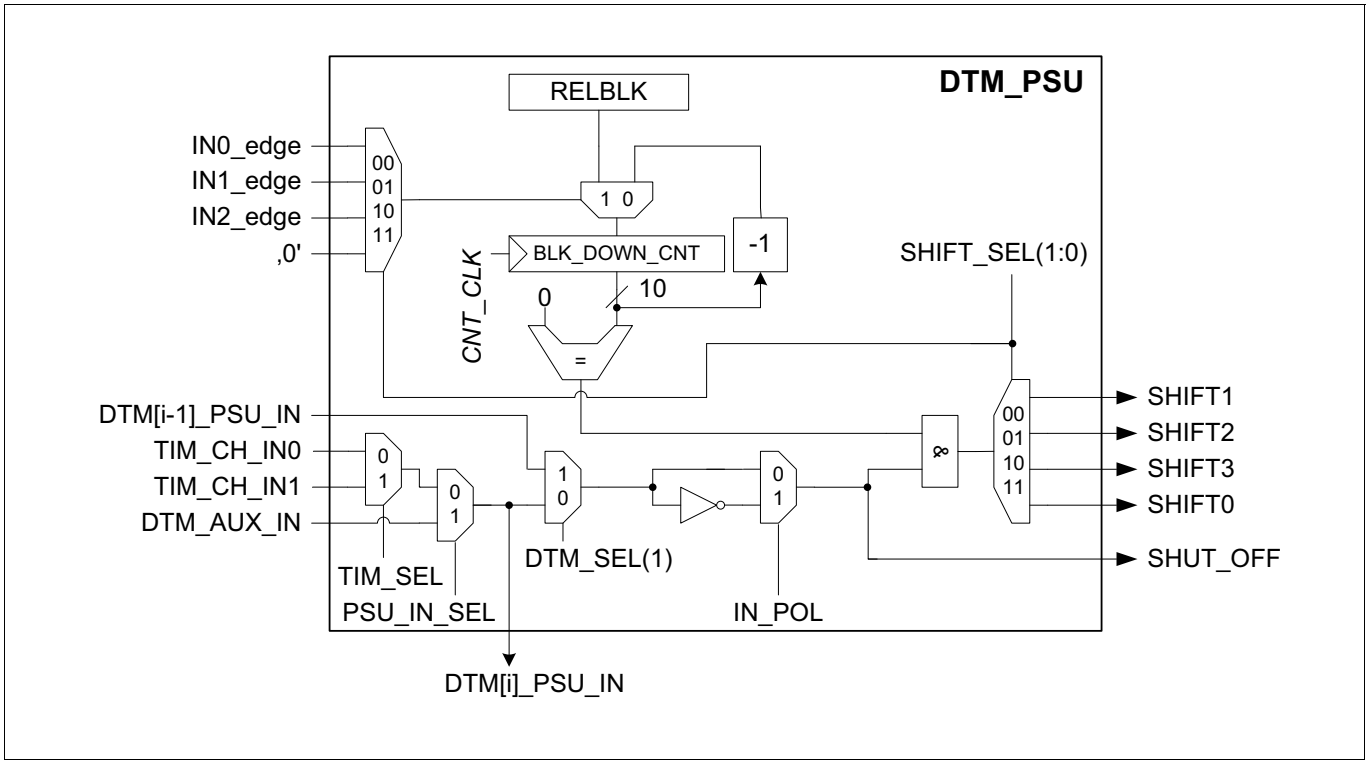


Figure 101 Phase Shift Unit overview

This sub-module provides an additional counter **BLK_DOWN_CNT** and reload register **RELBLK** (bit field of register **DTM[i]_PS_CTRL**). The counter is reloaded on an edge detected on one of the selected signals *IN0_edge* to *IN2_edge* (selected by bit field **SHIFT_SEL** in register **DTM[i]_PS_CTRL**). Then, the counter counts down until it reaches 0. While the counter is counting down, it blocks the trigger (i.e. the selected one of the signals *SHIFT[x]*) of one of the channels by one of the input signals *TIM_CH_IN0, TIM_CH_IN1* or *DTM_AUX_IN*.

If the counter **BLK_DOWN_CNT** is not counting, a pulse on the input *TIM_CH_IN*0, *TIM_CH_IN*1 or *DTM_AUX_IN* is forwarded to one of the selected DTM_PSU outputs *SHIFT[x]*. This signal triggers in the selected channel (if **SH_EN_x**=1) the update of the first flip-flop on channel x (i.e. representing *IN[x]_DLY*) to the input value *DTM_IN[x-1]* of the preceding channel. If this update leads to an edge, the succeeding part of DTM channel derives the inverse signal and applies the corresponding dead time (i.e. the edge delay) to the output signals of the channel.

*Note: For channel x=0 input signals DTM_IN[x-1] is unused and **SH_EN_[x]** is defined as 0.*

Figure 102 shows an example of phase shifting on channel 1.

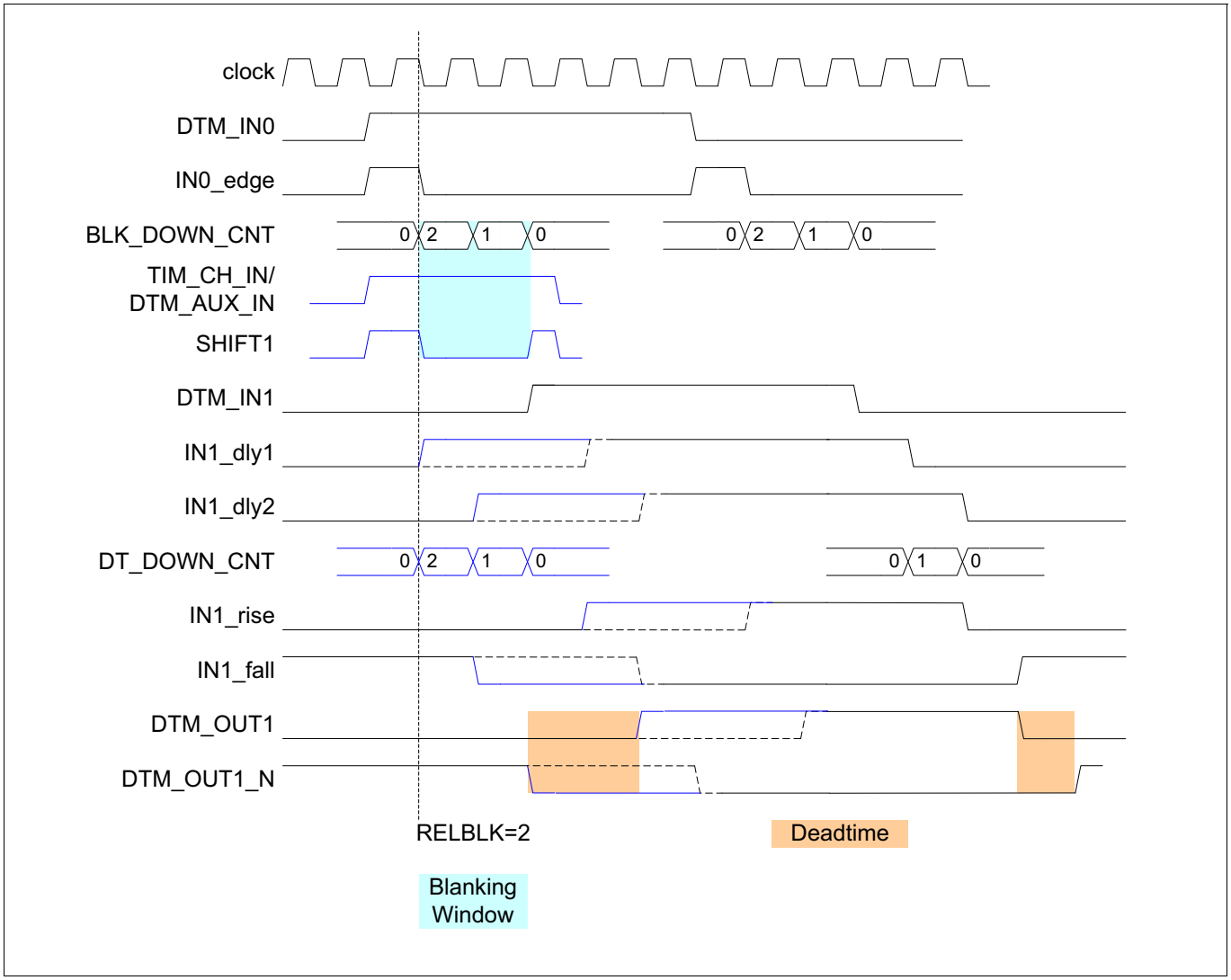


Figure 102 Example wave of phase shift on channel 1

28.16.4 Multiple output signal combination

Each channel provides additionally the possibility to combine the channel inputs *DTM_IN[x]* and *SHIFT[x]* or *COUT[x-1]* (selected by **I1SEL_[x]**) by an AND or an XOR gate (selected by **O1F_[x]**).

It is recommended to use the combination of signals only if bit field **RELBLK** of register **DTM[i]_PS_CTRL** is 0. Otherwise, the signal *TIM_CH_IN0, TIM_CH_IN1 or DTM_AUX_IN* may be disturbed by the blanking window counter.

Together with the inverter inside sub-module DTM_PSU (selected by **IN_POL**), the inverter on each output of a channel (selected by **POL0_[x]**/**POL1_[x]**) and the possibility to change polarity of *DTM_IN[x]* inside connected TOM/ATOM channel, a (N)AND, (N)OR or X(N)OR combination of the signals is possible.

28.16.4.1 Combination of input signal TIM_CH_IN/AUX_IN with TOM/ATOM signal

If the input selection **I1SEL_[x]** of a channel x is set to 0, the output selection **O1SEL_[x]** is set to 1 and **SWAP_[x]** is set to 0, depending on **PSU_IN_SEL** either *TIM_CH_IN0, TIM_CH_IN1* or *DTM_AUX_IN* can be combined with signal *DTM_IN[x]*.

The function of combination on DTM output *DTM[i]_OUT[x]_N* (and also *COUT[x]*) is defined by **O1F_[x]** in the following way:

Table 61 Function of combination on DTM channel x=0 output DTM[i]_OUT[x]_N (and also COUT[x])

	O1F_x	POL1_x	IN_POL	(A)TOM output inverted
XOR	01	0	0	no
AND	10	0	0	no
XNOR	01	1	0	no
NAND	10	1	0	no
XNOR	01	1	1	yes
OR	10	1	1	yes
XOR	01	0	1	yes
NOR	10	0	1	yes

*Note: The inversion of the (A)TOM output can be reached by switching the **SL** bit (for TOM and ATOM SOMP/SOMC mode).*

28.16.4.2 Combination of multiple TOM/ATOM output signals

If the input selection **I1SEL_[x]** of a channel x (with x=1…3) is set to 1, the output selection **O1SEL_[x]** is set to 1 and **SWAP_[x]** is set to 0, the output of the preceding DTM channel *COUT[x-1]* can be combined with signal *DTM_IN[x]*.

The function of combination on DTM output *DTM[i]_OUT[x]_N* (and also *COUT[x]*) is defined by **O1F_[x]** in the following way:

Table 62 Function of combination on DTM on channel x=1…3 output DTM[i]_OUT[x]_N (and also COUT[x])

	O1F_x	POL1_x	POL1_x-1	(A)TOM output inverted
XOR	01	0	0	no
AND	10	0	0	no
XNOR	01	1	0	no
NAND	10	1	0	no
XNOR	01	1	1	yes
OR	10	1	1	yes
XOR	01	0	1	yes
NOR	10	0	1	yes

By setting **I1SEL_[x]** to 1 on all four channel, a combination of all four signals *DTM_IN0* to *DTM_IN3* can be achieved (combinatorial chain).

To allow also combination of signals generated for output *DTM[i]_OUT[x]*, the outputs 0 and 1 can be swapped by setting bit **SWAP_ [x]** for channel x.

28.16.4.3 Pulse generation on edge

Another feature of the DTM is to generate on the second output *DTM[i]_OUT[x]_N* a pulse on every edge of corresponding input signal *DTM[i]_IN[x]*.

This can be reached by configuring **O1SEL_[x]** to 1, i.e. selecting signal *edge_trigg_[x]* as the output signal (**O1F_[x]** has to be 0b00). The signal *edge_trigg_[x]* is depicted in **Figure 99**.

The pulse length can be adjusted individually for each edge type by the configuration value **REL_RISE** and **REL_FALL** of register **DTM[i]_CH[x]_DV**.

The parameter **REL_RISE** defines the pulse length in case of a rising edge on input *DTM[i]_IN[x]*, the parameter **REL_FALL** define the pulse length in case of a falling edge on input *DTM[i]_IN[x]*.

The generated edge signal *edge_trigg_[x]* can be combined with the output signal of the preceding DTM channel x-1 at channel input *COUT[x-1]* (see **Figure 98**).

With the configuration of **CIS[x]**=1 and **I1SEL_[x]**=1, **CII[x]**=0 and **POL1_[x]**=1, the signal *edge_trigg_[x]* of channel x is ORed with the inverse signal at channel input *COUT[x-1]*. The signal at **COUT[x-1]** can be inverted by changing **POL1_[x-1]** of channel x-1.

As a result of this configuration one can generate at each edge on DTM input *DTM_IN[x]* a pulse signal and OR-combine these generated pulse signals with the generated signal of preceding DTM channel. If the combinatorial chain is configured over all four DTM channel the final signal is available at last DTM output *DTM_OUT3_N*.

28.16.5 Synchronous update of channel control register 2

It is possible to use the shadow register **DTM[i]_CH_CTRL2_SR** and a selected edge of one of the channel 0 to 3 to update the work register **DTM[i]_CH_CTRL2**.

The update mechanism and its configuration are depicted in **Figure 103**.

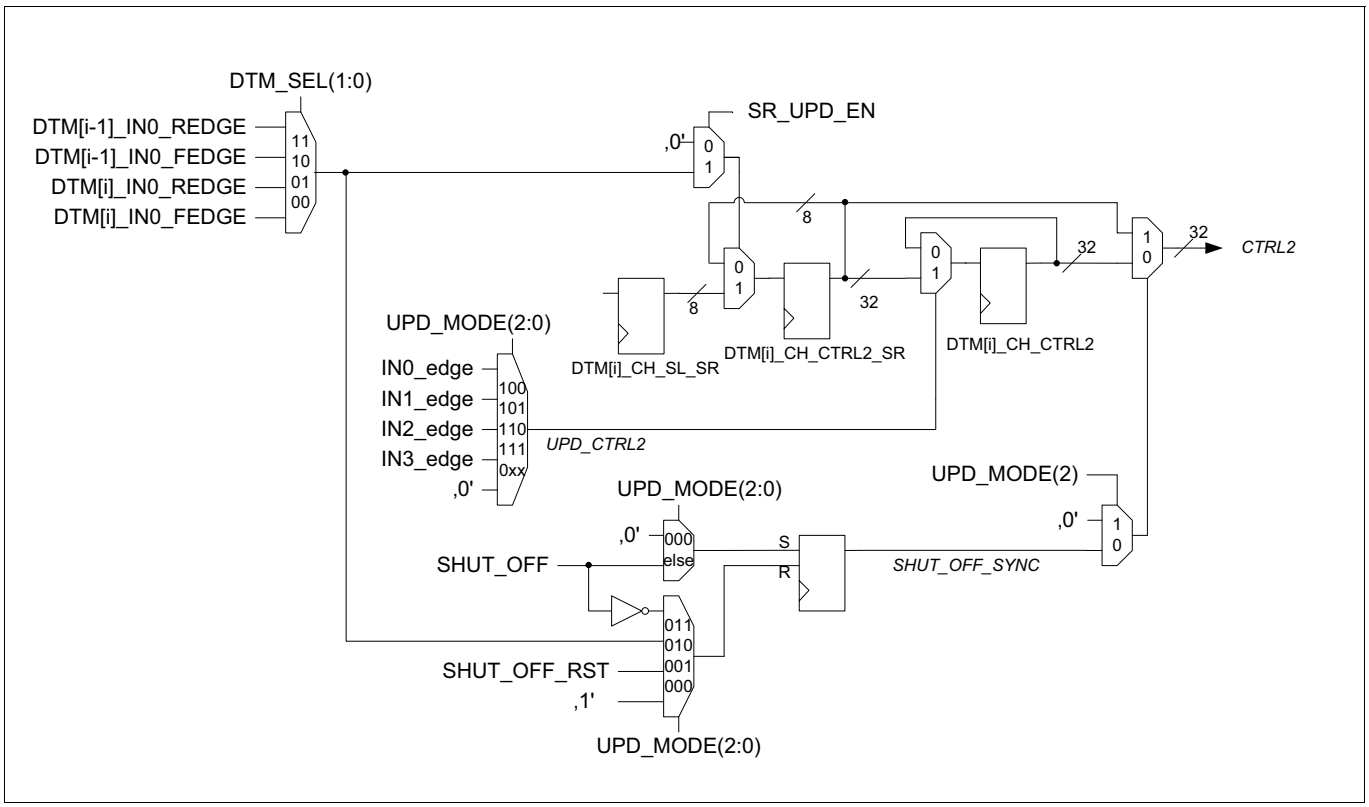


Figure 103 Synchronous update mechanism of register DTM[i]_CH_CTRL2

If enabled by the bit field **UPD_MODE** of register **DTM[i]_CTRL** (i.e. **UPD_MODE**=1xx), the register **DTM[i]_CH_CTRL2_SR** serves as a shadow register of register **DTM[i]_CH_CTRL2**. The update is then triggered by an edge on one of the selected inputs *DTM_IN0* to *DTM_IN3*.

The synchronous update allows the user to change output polarity, the selection of constant signal level, the constant signal level itself and the switch to/from feed through path on all four channels in parallel synchronized to one of the input edges on *DTM_IN0* to *DTM_IN3*.

28.16.6 DTM output shut off

A fast shut off for the eight outputs of DTM instance i can be triggered by one of the two assigned inputs TIM[n]_CH_IN or DTM[i]_AUX_IN or the two inputs TIM[m]_CH_IN or DTM[i-1]_AUX_IN of the previous DTM instance i-1. The selection of the trigger signal source is done by the bits **PSU_IN_SEL** and **DTM_SEL(1)** (see **Figure 101**). The selected trigger signal is named *SHUT_OFF*.

Enabling of the shut off feature is done by setting **UPD_MODE(2:0)** to one of the values 0b001, 0b010 or 0b011.

The shut off behavior of the DTM outputs is defined by the value of register **DTM[i]_CH_CTRL2_SR**.

If the shut off feature is enabled by **UPD_MODE**, as long as the signal *SHUT_OFF_SYNC* is 0, the register **DTM[i]_CH_CTRL2** defines the output signal behavior.

If the signal *SHUT_OFF_SYNC* is 1, the register **DTM[i]_CH_CTRL2_SR** defines the output signal behavior.

The signal *SHUT_OFF_SYNC* is set to 1 if signal *SHUT_OFF* switches to 1.The reset depends on value of **UPD_MODE(2:0)**.

There are three different ways to reset the signal *SHUT_OFF_SYNC* to 0:

- the CPU writes a 1 to bit **SHUT_OFF_RST** of register **DTM_CH_CTRL1**
- synchronous to an edge on DTM channel 0 input of this DTM instance i or on an edge on DTM channel 0 input of preceding DTM instance i-1.
- asynchronous if signal *SHUT_OFF* switches back to 0

Additionally, setting UPD_MODE(2:0) to value 0b000 or 0b1xx resets also the signal SHUT_OFF_SYNC.

Figure 103 depicts the shut off feature and the different shut off release possibilities.

Note: The reset of SHUT_OFF_SYNC has lower priority than the set of this signal.

A second shadow register **DTM[i]_CH_SR** exist for the eight **SL** bits (SLx_y_SR) of the shadow register **DTM[i]_CH_CTRL2_SR**.

If enabled by configuration bit **SR_UPD_EN** of register **DTM[i]_CTRL**, the update of **SL** bits of register **DTM[i]_CH_CTRL2_SR** can be triggered by one of the signals selected by bit field **DTM_SEL** of register **DTM[i]_CTRL**. This trigger signal is either the rising or the falling edge detected on *DTM[i]_IN0* of instance i or the rising or the falling edge on *DTM[i-1]_IN0* of preceding instance i-1.

As depicted in **Figure 95** the DTM input signal *TIM_CH_IN0, TIM_CH_IN1 or DTM_AUX_IN* can be forwarded to the succeeding instance. Thus, it can be used to trigger shut off in two consecutive DTM instances.

28.16.7 DTM connections on GTM top level

The DTM, if present, is placed behind the outputs of a TOM or ATOM. The outputs of the DTM are routed directly to the top level ports of GTM.If there is a DTM placed behind a TOM or ATOM depends on the GTM device configuration.

In case of a DTM behind a TOM or ATOM, the outputs (A)TOM_OUT and (A)TOM_OUT_T are connected to DTM inputs DTM_IN and DTM_IN_T. The outputs of the DTM are routed directly to the top level of GTM.The behavior of DTM after reset is shown in **Figure 104**.

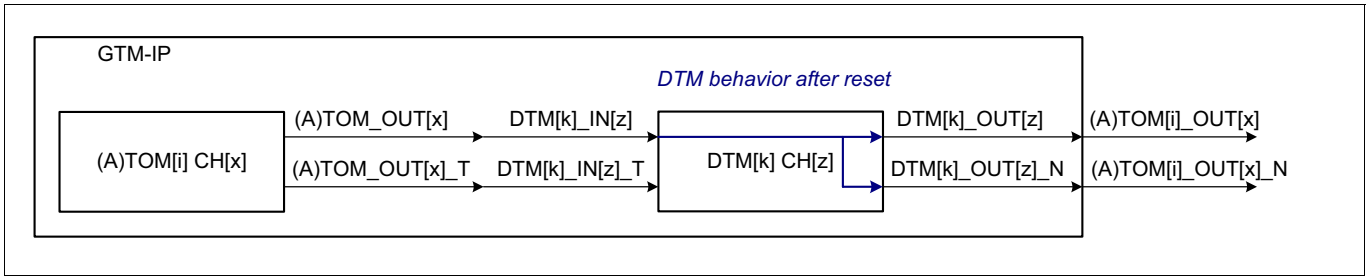


Figure 104 DTM behavior after reset

To route the signal *DTM[k]_IN[z]_T* to the DTM output DTM_[k]_OUT[z]_N, the following DTM channel configuration has to be chosen:

O1F_[x] = 11, **O1SEL_[x]**=1 and **DT1_[x]**=1.

The signals names and the signal routing in the case of no DTM instance is placed behind a TOM or ATOM is shown in **Figure 105**.

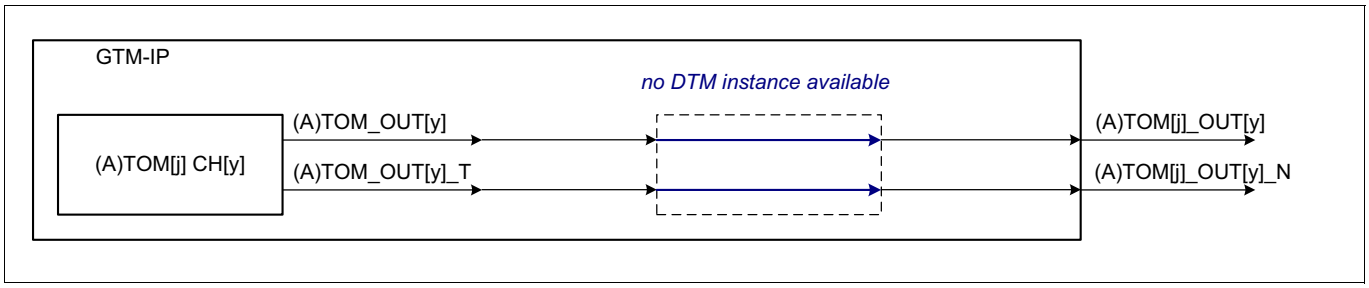


Figure 105 (A)TOM output signal routing in case of no DTM instance available

28.16.8 CDTM Configuration Register Overview

Table 63 Configuration Register Overview

Register name	Description	see Page
CDTM[i]_DTM[j]_CTRL	CDTMi DTMj global configuration and control register	332
CDTM[i]_DTM[j]_CH_CTRL1	CDTMi DTMj channel control register 1	334
CDTM[i]_DTM[j]_CH_CTRL2	CDTMi DTMj channel control register 2	337
CDTM[i]_DTM[j]_CH_CTRL2_SR	CDTMi DTMj channel control register 2 shadow	340
CDTM[i]_DTM[j]_CH_CTRL3	CDTMi DTMj channel control register 3	343
CDTM[i]_DTM[j]_PS_CTRL	CDTMi DTMj phase shift unit configuration and control register	345
CDTM[i]_DTM[j]_CH[z]_DTV	CDTMi DTMj dead time reload values	346
CDTM[i]_DTM[j]_CH_SR	CDTM DTMj channel shadow register	347

28.16.9 Configuration Register Description

28.16.9.1 Register CDTM[i]_DTM[j]_CTRL

CDTM0 DTM0 Global Configuration and Control Register

CDTMi_DTMj_CTRL (i=0-4;j=0-1,4-5)
CDTMi DTMj Global Configuration and Control Register($0E4000_H$+i*400_H+j*40_H) Application Reset Value: $0000\ 0000_H$
CDTMi_DTMj_CTRL (i=5-6;j=0-3)
CDTMi DTMj Global Configuration and Control Register($0E4000_H$+i*400_H+j*40_H) Application Reset Value: $0000\ 0000_H$
CDTMi_DTMj_CTRL (i=0-4;j=2-3)
CDTMi DTMj Global Configuration and Control Register($0E4000_H$+i*400_H+j*40_H) Application Reset Value: $0000\ 0000_H$
CDTMi_DTMj_CTRL (i=5-6;j=4-5)
CDTMi DTMj Global Configuration and Control Register($0E4000_H$+i*400_H+j*40_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															SHUT_OFF_RST
r															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							SR_UPD_EN	0	UPD_MODE			DTM_SEL		CLK_SEL	
r							rw	r	rw			rw		rw	

Field	Bits	Type	Description
CLK_SEL	1:0	rw	**Clock source select** 00_B SYS_CLK selected 01_B CMU_CLK0 selected 10_B CMU_CLK1 selected (if DTM is connected to an ATOM) / CMU_FXCLK0 selected (if DTM is connected to a TOM) 11_B CMU_CLK2 selected (if DTM is connected to an ATOM) / CMU_FXCLK1 selected (if DTM is connected to a TOM)
DTM_SEL	3:2	rw	**Select DTM update and SHUT_OFF reset signal** $0X_B$ Shut off by signal *TIM_CH_IN0*, *TIM_CH_IN1* or *DTM_AUX_IN* $1X_B$ Shut off by signal *DTM[i-1]_PSU_IN* 00_B Select falling edge on DTM[i] channel 0 input 01_B Select rising edge on DTM[i] channel 0 input 10_B Select falling edge on DTM[i-1] channel 0 input 11_B Select rising edge on DTM[i-1] channel 0 input 0b1 = shut off by signal DTM[i-1]_PSU_IN

Field	Bits	Type	Description
UPD_MODE	6:4	rw	**Update mode** *Note: If an INx_edge is not implemented, the value is unused. A write with this unused value returns 0b10 on status.* 000_B Asynchronous update - DTM[i]_CH_CTRL2_SR not used for update of DTM[i]_CH_CTRL2 001_B Shut off release by writing 1 to bit SHUT_OFF_RST of register DTM[i]_CTRL 010_B Shut off release by an edge on DTM[i]_IN0 or DTM[i-1]_IN0 (defined by bit field DTM_SEL of register DTM[i]_CTRL) 011_B Shut off release by shut off signal SHUT_OFF (defined by bits PSU_IN_SEL and IN_POL of register DTM[i]_PS_CTRL and DTM_SEL(2) of register DTM[i]_CTRL) 100_B Signal IN0_edge used to trigger update of DTM[i]_CH_CTRL2 with content of DTM[i]_CH_CTRL2_SR 101_B Signal IN1_edge used to trigger update of DTM[i]_CH_CTRL2 with content of DTM[i]_CH_CTRL2_SR 110_B Signal IN2_edge used to trigger update of DTM[i]_CH_CTRL2 with content of DTM[i]_CH_CTRL2_SR 111_B Signal IN3_edge used to trigger update of DTM[i]_CH_CTRL2 with content of DTM[i]_CH_CTRL2_SR
SR_UPD_EN	8	rw	**Shadow register update enable** 0_B No update of SLx_y_SR register bits in register DTM[i]_CH_CTRL2_SR 1_B Update of SLx_y_SR register bits in register DTM[i]_CH_CTRL2_SR on trigger
SHUT_OFF_RST	16	rw	**Shut off reset** Writing a 1 releases shut off (resets signal *SHUT_OFF_SYNC* if selected by UPD_MODE(2:0)=0b001)
0	7, 15:9, 31:17	r	**Reserved** Read as zero, shall be written as zero.

28.16.9.2 Register CDTM[i]_DTM[j]_CH_CTRL1

CDTM0 DTM0 Channel Control Register 1

CDTMi_DTMj_CH_CTRL1 (i=0-4;j=0-1,4-5)
CDTMi DTMj Channel Control Register 1 ($0E4004_H$+i*400_H+j*40_H) **Application Reset Value: 0000 0000_H**
CDTMi_DTMj_CH_CTRL1 (i=5-6;j=0-3)
CDTMi DTMj Channel Control Register 1 ($0E4004_H$+i*400_H+j*40_H) **Application Reset Value: 0000 0000_H**
CDTMi_DTMj_CH_CTRL1 (i=0-4;j=2-3)
CDTMi DTMj Channel Control Register 1 ($0E4004_H$+i*400_H+j*40_H) **Application Reset Value: 0000 0000_H**
CDTMi_DTMj_CH_CTRL1 (i=5-6;j=4-5)
CDTMi DTMj Channel Control Register 1 ($0E4004_H$+i*400_H+j*40_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		O1F_3		SWAP_3	SH_EN_3	I1SEL_3	O1SEL_3	0	XDT_EN_2_3	O1F_2		SWAP_2	SH_EN_2	I1SEL_2	O1SEL_2
r		rw		rw	rw	rw	rw	r	rw	rw		rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		O1F_1		SWAP_1	SH_EN_1	I1SEL_1	O1SEL_1	0	XDT_EN_0_1	O1F_0		SWAP_0	0	I1SEL_0	O1SEL_0
r		rw		rw	rw	rw	rw	r	rw	rw		rw	r	rw	rw

Field	Bits	Type	Description
O1SEL_0	0	rw	**Output 1 select channel 0** 0_B Inverse dead time signal selected 1_B Special function on output 1 selected (defined by O1F_0)
I1SEL_0	1	rw	**Input 1 select channel 0** *Note: If i is even, I1SEL_0 is not implemented. Then the bit is read as zero and shall be written as zero.* 0_B Signal (PSU_)SHIFT0 selected 1_B Signal COUT3 from DTM[i-1] selected
SWAP_0	3	rw	**Swap outputs DTM[i]_CH[0]_OUT0 and DTM[i]_CH[0]_OUT1 (before final output register)** 0_B Outputs not swapped 1_B Swap outputs DTM[i]_OUT0 and DTM[i]_OUT0_N
O1F_0	5:4	rw	**Output 1 function channel 0** 00_B Signal edge_trigg is selected 01_B XOR of DTM[i]_IN0 and signal SHIFT0 10_B AND of DTM[i]_IN0 and signal SHIFT0 11_B DTM[i]_IN0_T selected

Field	Bits	Type	Description
XDT_EN_0_1	6	rw	**Cross dead time enable on channels 0 and 1** *Note: TSEL0_[x] and SH_EN_1 must be '0' for using cross dead time to avoid wrong input signals. (x:0,1)* 0_B Cross dead time disabled on channels 0 and 1 1_B Cross dead time enabled on channels 0 and 1 *Note: When a '1' is written to bit XDT_EN_0_1, the internal register IN[x]_dly1, IN[x]_dly2 and DT_DOWN_CNT is reset to '0' (x:0,1)*
O1SEL_1	8	rw	**Output 1 select channel 1** 0_B Inverse dead time signal selected 1_B Special function on output 1 selected (defined by O1F_1)
I1SEL_1	9	rw	**Input 1 select channel 1** 0_B Signal (PSU_)SHIFT1 selected 1_B Signal COUT1 selected
SH_EN_1	10	rw	**Shift enable channel 1** 0_B DTM[i]_IN0 is not used; no input signal shift 1_B Signal selected by I1SEL_1 triggers update of DTM[i]_IN1 with input of DTM[i]_IN0 -> input signal shift
SWAP_1	11	rw	**Swap outputs DTM[i]_CH[1]_OUT0 and DTM[i]_CH[1]_OUT1 (before final output register)** 0_B Outputs not swapped 1_B Swap outputs DTM[i]_OUT1 and DTM[i]_OUT1_N
O1F_1	13:12	rw	**Output 1 function channel 1** 00_B Signal edge_trigg is selected 01_B XOR of DTM[i]_IN1 and signal SHIFT1/OUT0 10_B AND of DTM[i]_IN1 and signal SHIFT1/OUT0 11_B DTM[i]_IN1_T selected
O1SEL_2	16	rw	**Output 1 select channel 2** 0_B Inverse dead time signal selected 1_B Special function on output 1 selected (defined by O1F_2)
I1SEL_2	17	rw	**Input 1 select channel 2** 0_B Signal (PSU_)SHIFT2 selected 1_B Signal COUT1 selected
SH_EN_2	18	rw	**Shift enable channel 2** 0_B DTM[i]_IN1 is not used; no input signal shift 1_B Signal selected by I1SEL_2 triggers update of DTM[i]_IN2 with input of DTM[i]_IN1 -> input signal shift
SWAP_2	19	rw	**Swap outputs DTM[i]_CH[2]_OUT0 and DTM[i]_CH[2]_OUT1 (before final output register)** 0_B Outputs not swapped 1_B Swap outputs DTM[i]_OUT2 and DTM[i]_OUT2_N
O1F_2	21:20	rw	**Output 1 function channel 2** 00_B Signal edge_trigg is selected 01_B XOR of DTM[i]_IN2 and signal SHIFT2/OUT1 10_B AND of DTM[i]_IN2 and signal SHIFT2/OUT1 11_B DTM[i]_IN2_T selected

Field	Bits	Type	Description
XDT_EN_2_3	22	rw	**Cross dead time enable on channels 0 and 1** *Note: TSEL0_[x] and SH_EN_[x] must be '0' for using cross dead time to avoid wrong input signals. (x:2,3)* 0_B Cross dead time disabled on channels 2 and 3 1_B Cross dead time enabled on channels 2 and 3 *Note: When a '1' is written to bit XDT_EN_2_3, the internal register IN[x]_dly1, IN[x]_dly2 and DT_DOWN_CNT is reset to '0' (x:2,3)*
O1SEL_3	24	rw	**Output 1 select channel 3** 0_B Inverse dead time signal selected 1_B Special function on output 1 selected (defined by O1F_3)
I1SEL_3	25	rw	**Input 1 select channel 3** 0_B Signal (PSU_)SHIFT3 selected 1_B Signal COUT2 selected
SH_EN_3	26	rw	**Shift enable channel 3** 0_B DTM[i]_IN2 is not used; no input signal shift 1_B Signal selected by I1SEL_3 triggers update of DTM[i]_IN3 with input of TM[i]_IN2-> input signal shift
SWAP_3	27	rw	**Swap outputs DTM[i]_CH[3]_OUT0 and DTM[i]_CH[3]_OUT1 (before final output register)** 0_B Outputs not swapped 1_B Swap outputs DTM[i]_OUT3 and DTM[i]_OUT3_N
O1F_3	29:28	rw	**Output 1 function channel 3** 00_B Signal edge_trigg is selected 01_B XOR of DTM[i]_IN3 and signal SHIFT3 / OUT2 10_B AND of DTM[i]_IN3 and signal SHIFT3 / OUT2 11_B DTM[i]_IN3_T selected
0	2, 7, 15:14, 23, 31:30	r	**Reserved** Read as zero, shall be written as zero.

28.16.9.3 Register CDTM[i]_DTM[j]_CH_CTRL2

CDTM0 DTM0 Channel Control Register 2

CDTMi_DTMj_CH_CTRL2 (i=0-4;j=0-1,4-5)
CDTMi DTMj Channel Control Register 2 ($0E4008_H$+i*400_H+j*40_H) **Application Reset Value: 0000 0000_H**
CDTMi_DTMj_CH_CTRL2 (i=5-6;j=0-3)
CDTMi DTMj Channel Control Register 2 ($0E4008_H$+i*400_H+j*40_H) **Application Reset Value: 0000 0000_H**
CDTMi_DTMj_CH_CTRL2 (i=0-4;j=2-3)
CDTMi DTMj Channel Control Register 2 ($0E4008_H$+i*400_H+j*40_H) **Application Reset Value: 0000 0000_H**
CDTMi_DTMj_CH_CTRL2 (i=5-6;j=4-5)
CDTMi DTMj Channel Control Register 2 ($0E4008_H$+i*400_H+j*40_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DT1_3	**SL1_3**	**OC1_3**	**POL1_3**	**DT0_3**	**SL0_3**	**OC0_3**	**POL0_3**	**DT1_2**	**SL1_2**	**OC1_2**	**POL1_2**	**DT0_2**	**SL0_2**	**OC0_2**	**POL0_2**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT1_1	**SL1_1**	**OC1_1**	**POL1_1**	**DT0_1**	**SL0_1**	**OC0_1**	**POL0_1**	**DT1_0**	**SL1_0**	**OC1_0**	**POL1_0**	**DT0_0**	**SL0_0**	**OC0_0**	**POL0_0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
POL0_0	0	rw	**Polarity on output 0 channel 0** 0_B Output signal not inverted 1_B Output signal inverted
OC0_0	1	rw	**Output 0 control channel 0** 0_B Functional output 1_B Constant output defined by SL0_0
SL0_0	2	rw	**Signal level on output 0 channel 0** 0_B Signal Level is 0 on output if OC0_0=1 1_B Signal Level is 1 on output if OC0_0=1
DT0_0	3	rw	**Dead time path enable on output 0 channel 0** 0_B Feed through from DTM_IN0 to DTM[i]_OUT0 enabled 1_B Dead time path enabled
POL1_0	4	rw	**Polarity on output 1 channel 0** 0_B Output signal not inverted 1_B Output signal inverted
OC1_0	5	rw	**Output 1 control channel 0** 0_B Functional output 1_B Constant output defined by SL1_0
SL1_0	6	rw	**Signal level on output 1 channel 0** 0_B Signal Level is 0 on output if OC1_0=1 1_B Signal Level is 1 on output if OC1_0=1

Field	Bits	Type	Description
DT1_0	7	rw	**Dead time path enable on output 1 channel 0** 0_B Feed through from DTM_IN0 to DTM[i]_OUT0_N enabled 1_B Dead time path enabled
POL0_1	8	rw	**Polarity on output 0 channel 1** 0_B Output signal not inverted 1_B Output signal inverted
OC0_1	9	rw	**Output 0 control channel 1** 0_B Functional output 1_B Constant output defined by SL0_1
SL0_1	10	rw	**Signal level on output 0 channel 1** 0_B Signal Level is 0 on output if OC0_1=1 1_B Signal Level is 1 on output if OC0_1=1
DT0_1	11	rw	**Dead time path enable on output 0 channel 1** 0_B Feed through from DTM_IN1 to DTM[i]_OUT1 enabled 1_B Dead time path enabled
POL1_1	12	rw	**Polarity on output 1 channel 1** 0_B Output signal not inverted 1_B Output signal inverted
OC1_1	13	rw	**Output 1 control channel 1** 0_B Functional output 1_B Constant output defined by SL1_1
SL1_1	14	rw	**Signal level on output 1 channel 1** 0_B Signal Level is 0 on output if OC1_1=1 1_B Signal Level is 1 on output if OC1_1=1
DT1_1	15	rw	**Dead time path enable on output 1 channel 1** 0_B Feed through from DTM_IN1 to DTM[i]_OUT1_N enabled 1_B Dead time path enabled
POL0_2	16	rw	**Polarity on output 0 channel 2** 0_B Output signal not inverted 1_B Output signal inverted
OC0_2	17	rw	**Output 0 control channel 2** 0_B Functional output 1_B Constant output defined by SL0_2
SL0_2	18	rw	**Signal level on output 0 channel 2** 0_B Signal Level is 0 on output if OC0_2=1 1_B Signal Level is 1 on output if OC0_2=1
DT0_2	19	rw	**Dead time path enable on output 0 channel 2** 0_B Feed through from DTM_IN2 to DTM[i]_OUT2 enabled 1_B Dead time path enabled
POL1_2	20	rw	**Polarity on output 1 channel 2** 0_B Output signal not inverted 1_B Output signal inverted
OC1_2	21	rw	**Output 1 control channel 2** 0_B Functional output 1_B Constant output defined by SL1_2

Field	Bits	Type	Description
SL1_2	22	rw	**Signal level on output 1 channel 2** 0_B Signal Level is 0 on output if OC1_2=1 1_B Signal Level is 1 on output if OC1_2=1
DT1_2	23	rw	**Dead time path enable on output 1 channel 2** 0_B Feed through from DTM_IN2 to DTM[i]_OUT2_N enabled 1_B Dead time path enabled
POL0_3	24	rw	**Polarity on output 0 channel 3** 0_B Output signal not inverted 1_B Output signal inverted
OC0_3	25	rw	**Output 0 control channel 3** 0_B Functional output 1_B Constant output defined by SL0_3
SL0_3	26	rw	**Signal level on output 0 channel 3** 0_B Signal Level is 0 on output if OC0_3=1 1_B Signal Level is 1 on output if OC0_3=1
DT0_3	27	rw	**Dead time path enable on output 0 channel 3** 0_B Feed through from DTM_IN3 to DTM[i]_OUT3 enabled 1_B Dead time path enabled
POL1_3	28	rw	**Polarity on output 1 channel 3** 0_B Output signal not inverted 1_B Output signal inverted
OC1_3	29	rw	**Output 1 control channel 3** 0_B Functional output 1_B Constant output defined by SL1_3
SL1_3	30	rw	**Signal level on output 1 channel 3** 0_B Signal Level is 0 on output if OC1_3=1 1_B Signal Level is 1 on output if OC1_3=1
DT1_3	31	rw	**Dead time path enable on output 1 channel 3** 0_B Feed through from DTM_IN3 to DTM[i]_OUT3_N enabled 1_B Dead time path enabled

28.16.9.4 Register CDTM[i]_DTM[j]_CH_CTRL2_SR

CDTM0 DTM0 Channel Control Register 2 Shadow

CDTMi_DTMj_CH_CTRL2_SR (i=0-4;j=0-1,4-5)
CDTMi DTMj Channel Control Register 2 Shadow($0E400C_H$+i*400_H+j*40_H) Application Reset Value: $0000\ 0000_H$
CDTMi_DTMj_CH_CTRL2_SR (i=5-6;j=0-3)
CDTMi DTMj Channel Control Register 2 Shadow($0E400C_H$+i*400_H+j*40_H) Application Reset Value: $0000\ 0000_H$
CDTMi_DTMj_CH_CTRL2_SR (i=0-4;j=2-3)
CDTMi DTMj Channel Control Register 2 Shadow($0E400C_H$+i*400_H+j*40_H) Application Reset Value: $0000\ 0000_H$
CDTMi_DTMj_CH_CTRL2_SR (i=5-6;j=4-5)
CDTMi DTMj Channel Control Register 2 Shadow($0E400C_H$+i*400_H+j*40_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DT1_3_SR	SL1_3_SR	OC1_3_SR	POL1_3_SR	DT0_3_SR	SL0_3_SR	OC0_3_SR	POL0_3_SR	DT1_2_SR	SL1_2_SR	OC1_2_SR	POL1_2_SR	DT0_2_SR	SL0_2_SR	OC0_2_SR	POL0_2_SR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT1_1_SR	SL1_1_SR	OC1_1_SR	POL1_1_SR	DT0_1_SR	SL0_1_SR	OC0_1_SR	POL0_1_SR	DT1_0_SR	SL1_0_SR	OC1_0_SR	POL1_0_SR	DT0_0_SR	SL0_0_SR	OC0_0_SR	POL0_0_SR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
POL0_0_SR	0	rw	**Polarity on output 0 channel 0 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC0_0_SR	1	rw	**Output 0 control channel 0 shadow register** 0_B Functional output 1_B Constant output defined by SL0_0
SL0_0_SR	2	rw	**Signal level on output 0 channel 0 shadow register** 0_B Signal Level is 0 on output if OC0_0=1 1_B Signal Level is 1 on output if OC0_0=1
DT0_0_SR	3	rw	**Dead time path enable on output 0 channel 0 shadow register** 0_B Feed through from DTM_IN0 to DTM[i]_OUT0 enabled 1_B Dead time path enabled
POL1_0_SR	4	rw	**Polarity on output 1 channel 0 shadow register** 0_B Output signal not inverted 1_B Output signal inverted

Field	Bits	Type	Description
OC1_0_SR	5	rw	**Output 1 control channel 0 shadow register** 0_B Functional output 1_B Constant output defined by SL1_0
SL1_0_SR	6	rw	**Signal level on output 1 channel 0 shadow register** 0_B Signal Level is 0 on output if OC1_0=1 1_B Signal Level is 1 on output if OC1_0=1
DT1_0_SR	7	rw	**Dead time path enable on output 1 channel 0 shadow register** 0_B Feed through from DTM_IN0 to DTM[i]_OUT0_N enabled 1_B Dead time path enabled
POL0_1_SR	8	rw	**Polarity on output 0 channel 1 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC0_1_SR	9	rw	**Output 0 control channel 1 shadow register** 0_B Functional output 1_B Constant output defined by SL0_1
SL0_1_SR	10	rw	**Signal level on output 0 channel 1 shadow register** 0_B Signal Level is 0 on output if OC0_1=1 1_B Signal Level is 1 on output if OC0_1=1
DT0_1_SR	11	rw	**Dead time path enable on output 0 channel 1 shadow register** 0_B Feed through from DTM_IN1 to DTM[i]_OUT1 enabled 1_B Dead time path enabled
POL1_1_SR	12	rw	**Polarity on output 1 channel 1 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC1_1_SR	13	rw	**Output 1 control channel 1 shadow register** 0_B Functional output 1_B Constant output defined by SL1_1
SL1_1_SR	14	rw	**Signal level on output 1 channel 1 shadow register** 0_B Signal Level is 0 on output if OC1_1=1 1_B Signal Level is 1 on output if OC1_1=1
DT1_1_SR	15	rw	**Dead time path enable on output 1 channel 1 shadow register** 0_B Feed through from DTM_IN1 to DTM[i]_OUT1_N 1_B Dead time path enabled

Field	Bits	Type	Description
POL0_2_SR	16	rw	**Polarity on output 0 channel 2 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC0_2_SR	17	rw	**Output 0 control channel 2 shadow register** 0_B Functional output 1_B Constant output defined by SL0_2
SL0_2_SR	18	rw	**Signal level on output 0 channel 2 shadow register** 0_B Signal Level is 0 on output if OC0_2=1 1_B Signal Level is 1 on output if OC0_2=1
DT0_2_SR	19	rw	**Dead time path enable on output 0 channel 2 shadow register** 0_B Feed through from DTM_IN2 to DTM[i]_OUT2 1_B Dead time path enabled
POL1_2_SR	20	rw	**Polarity on output 1 channel 2 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC1_2_SR	21	rw	**Output 1 control channel 2 shadow register** 0_B Functional output 1_B Constant output defined by SL1_2
SL1_2_SR	22	rw	**Signal level on output 1 channel 2 shadow register** 0_B Signal Level is 0 on output if OC1_2=1 1_B Signal Level is 1 on output if OC1_2=1
DT1_2_SR	23	rw	**Dead time path enable on output 1 channel 2 shadow register** 0_B Feed through from DTM_IN2 to DTM[i]_OUT2_N 1_B Dead time path enabled
POL0_3_SR	24	rw	**Polarity on output 0 channel 3 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC0_3_SR	25	rw	**Output 0 control channel 3 shadow register** 0_B Functional output 1_B Constant output defined by SL0_3
SL0_3_SR	26	rw	**Signal level on output 0 channel 3 shadow register** 0_B Signal Level is 0 on output if OC0_3=1 1_B Signal Level is 1 on output if OC0_3=1

Field	Bits	Type	Description
DT0_3_SR	27	rw	**Dead time path enable on output 0 channel 3 shadow register** 0_B Feed through from DTM_IN3 to DTM[i]_OUT3 1_B Dead time path enabled
POL1_3_SR	28	rw	**Polarity on output 1 channel 3 shadow register** 0_B Output signal not inverted 1_B Output signal inverted
OC1_3_SR	29	rw	**Output 1 control channel 3 shadow register** 0_B Functional output 1_B Constant output defined by SL1_3
SL1_3_SR	30	rw	**Signal level on output 1 channel 3 shadow register** 0_B Signal Level is 0 on output if OC1_3=1 1_B Signal Level is 1 on output if OC1_3=1
DT1_3_SR	31	rw	**Dead time path enable on output 1 channel 3 shadow register** 0_B Feed through from DTM_IN3 to DTM[i]_OUT3_N 1_B Dead time path enabled

28.16.9.5 Register CDTM[i]_DTM[j]_CH_CTRL3

CDTM0 DTM0 Channel Control Register 3

CDTMi_DTMj_CH_CTRL3 (i=0-4;j=0-1,4-5)
CDTMi DTMj Channel Control Register 3 ($0E4028_H+i*400_H+j*40_H$) **Application Reset Value: 0000 0000_H**
CDTMi_DTMj_CH_CTRL3 (i=5-6;j=0-3)
CDTMi DTMj Channel Control Register 3 ($0E4028_H+i*400_H+j*40_H$) **Application Reset Value: 0000 0000_H**
CDTMi_DTMj_CH_CTRL3 (i=0-4;j=2-3)
CDTMi DTMj Channel Control Register 3 ($0E4028_H+i*400_H+j*40_H$) **Application Reset Value: 0000 0000_H**
CDTMi_DTMj_CH_CTRL3 (i=5-6;j=4-5)
CDTMi DTMj Channel Control Register 3 ($0E4028_H+i*400_H+j*40_H$) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				TSEL1_3	TSEL0_3	CIS3	CII3	0				TSEL1_2	TSEL0_2	CIS2	CII2
r				rw	rw	rw	rw	r				rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				TSEL1_1	TSEL0_1	CIS1	CII1	0				TSEL1_0	TSEL0_0	CIS0	CII0
r				rw	rw	rw	rw	r				rw	rw	rw	rw

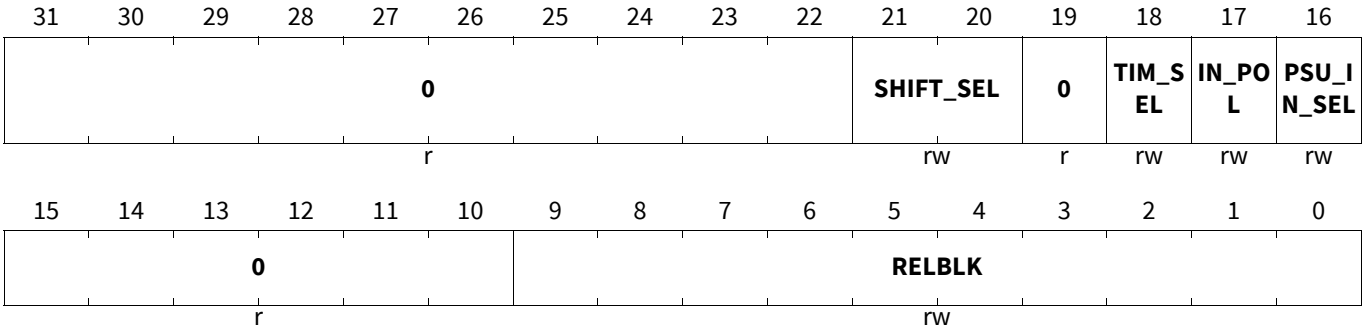
Field	Bits	Type	Description
CII0	0	rw	**Combinational input invert channel 0** 0_B Do not invert input 1_B Invert input
CIS0	1	rw	**Combinational input select channel 0** 0_B Select input DTM[i]_IN0 1_B Select internal signal edge_trigg_0
TSEL0_0	2	rw	**Input selection for dead time / edge trigger generation** 0_B Use DTM[i]_IN0 as input for dead time / edge trigger generation 1_B Use DTM[i]_IN0_T as input for dead time / edge trigger generation
TSEL1_0	3	rw	**Input selection combinational logic path** 0_B Use DTM[i]_IN0 as input for combinational logic path 1_B Use DTM[i]_IN0_T as input for combinational logic path
CII1	8	rw	**Combinational input invert channel 1** 0_B Do not invert input 1_B Invert input
CIS1	9	rw	**Combinational input select channel 1** 0_B Select input DTM[i]_IN1 1_B Select internal signal edge_trigg_1
TSEL0_1	10	rw	**Input selection for dead time / edge trigger generation** 0_B Use DTM[i]_IN1 as input for dead time / edge trigger generation 1_B Use DTM[i]_IN1_T as input for dead time / edge trigger generation
TSEL1_1	11	rw	**Input selection combinational logic path** 0_B Use DTM[i]_IN1 as input for combinational logic path 1_B Use DTM[i]_IN1_T as input for combinational logic path
CII2	16	rw	**Combinational input invert channel 2** 0_B Do not invert input 1_B Invert input
CIS2	17	rw	**Combinational input select channel 2** 0_B Select input DTM[i]_IN2 1_B Select internal signal edge_trigg_2
TSEL0_2	18	rw	**Input selection for dead time / edge trigger generation** 0_B Use DTM[i]_IN2 as input for dead time / edge trigger generation 1_B Use DTM[i]_IN2_T as input for dead time / edge trigger generation
TSEL1_2	19	rw	**Input selection combinational logic path** 0_B Use DTM[i]_IN2_T as input for dead time / edge trigger generation 1_B Use DTM[i]_IN2_T as input for combinational logic path
CII3	24	rw	**Combinational input invert channel 3** 0_B Do not invert input 1_B Invert input
CIS3	25	rw	**Combinational input select channel 3** 0_B Select input DTM[i]_IN3 1_B Select internal signal edge_trigg_3

Field	Bits	Type	Description
TSEL0_3	26	rw	**Input selection for dead time / edge trigger generation** 0_B Use DTM[i]_IN3 as input for dead time / edge trigger generation 1_B Use DTM[i]_IN3_T as input for dead time / edge trigger generation
TSEL1_3	27	rw	**Input selection combinational logic path** 0_B Use DTM[i]_IN3 as input for combinational logic path 1_B Use DTM[i]_IN3_T as input for combinational logic path
0	7:4, 15:12, 23:20, 31:28	r	**Reserved** Read as zero, shall be written as zero.

28.16.9.6 Register CDTM[i]_DTM[j]_PS_CTRL

CDTM0 DTM0 Phase Shift Unit Configuration and Control Register

CDTMi_DTMj_PS_CTRL (i=0-4;j=0-1,4-5)
CDTMi DTMj Phase Shift Unit Configuration and Control Register($0E4010_H$+i*400_H+j*40_H) Application Reset Value: 0000 0000_H
CDTMi_DTMj_PS_CTRL (i=5-6;j=0-3)
CDTMi DTMj Phase Shift Unit Configuration and Control Register($0E4010_H$+i*400_H+j*40_H) Application Reset Value: 0000 0000_H
CDTMi_DTMj_PS_CTRL (i=0-4;j=2-3)
CDTMi DTMj Phase Shift Unit Configuration and Control Register($0E4010_H$+i*400_H+j*40_H) Application Reset Value: 0000 0000_H
CDTMi_DTMj_PS_CTRL (i=5-6;j=4-5)
CDTMi DTMj Phase Shift Unit Configuration and Control Register($0E4010_H$+i*400_H+j*40_H) Application Reset Value: 0000 0000_H



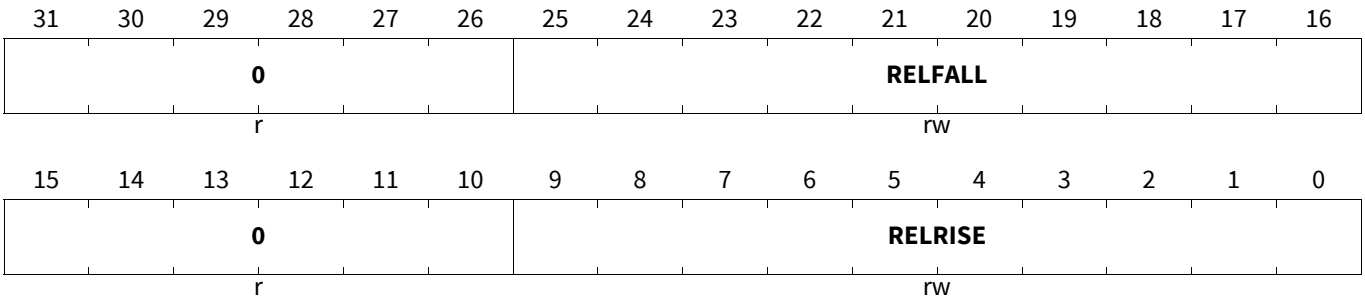
Field	Bits	Type	Description
RELBLK	9:0	rw	**Reload value blanking window** A value of 0x000 resets counter BLK_DOWN_CNT.
PSU_IN_SEL	16	rw	**PSU input selection** 0_B TIM_CH_IN0 or TIM_CH_IN1 selected 1_B DTM_AUX_IN selected
IN_POL	17	rw	**Input polarity** 0_B Input signal is not inverted 1_B Input signal is inverted

Field	Bits	Type	Description
TIM_SEL	18	rw	**TIM input selection** 0_B Select TIM_IN0 1_B Select TIM_IN1
SHIFT_SEL	21:20	rw	**Shift select** *Note: If a channel is not implemented, the value is unused. A write with this unused value returns 0b10 on status.* 00_B DTM channel 1 is connected via signal SHIFT1 with TIM_CH_IN0, TIM_CH_IN1, or DTM_AUX_IN 01_B DTM channel 2 is connected via signal SHIFT2 with TIM_CH_IN0, TIM_CH_IN1, or DTM_AUX_IN 10_B DTM channel 3 is connected via signal SHIFT3 with TIM_CH_IN0, TIM_CH_IN1, or DTM_AUX_IN 11_B DTM channel 0 is connected via signal SHIFT0 with TIM_CH_IN0, TIM_CH_IN1, or DTM_AUX_IN
0	15:10, 19, 31:22	r	**Reserved** Read as zero, shall be written as zero.

28.16.9.7 Register CDTM[i]_DTM[j]_CH[z]_DTV

CDTM0 DTM0 Channel z Dead Time Reload Values

CDTMi_DTMj_CHz_DTV (i=0-4;j=0-1,4-5;z=0-3)
CDTMi DTMj Channel z Dead Time Reload Values($0E4014_H$+i*400_H+j*40_H+z*4) Application Reset Value: $0000\ 0000_H$
CDTMi_DTMj_CHz_DTV (i=5-6;j=0-3;z=0-3)
CDTMi DTMj Channel z Dead Time Reload Values($0E4014_H$+i*400_H+j*40_H+z*4) Application Reset Value: $0000\ 0000_H$
CDTMi_DTMj_CHz_DTV (i=0-4;j=2-3;z=0-3)
CDTMi DTMj Channel z Dead Time Reload Values($0E4014_H$+i*400_H+j*40_H+z*4) Application Reset Value: $0000\ 0000_H$
CDTMi_DTMj_CHz_DTV (i=5-6;j=4-5;z=0-3)
CDTMi DTMj Channel z Dead Time Reload Values($0E4014_H$+i*400_H+j*40_H+z*4) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RELRISE	9:0	rw	**Reload value for rising edge dead time**
RELFALL	25:16	rw	**Reload value for falling edge dead time**

Field	Bits	Type	Description
0	15:10, 31:26	r	**Reserved** Read as zero, shall be written as zero.

28.16.9.8 Register CDTM[i]_DTM[j]_CH_SR

CDTM0 DTM0 Channel Shadow Register

CDTMi_DTMj_CH_SR (i=0-4;j=0-1,4-5)
CDTMi DTMj Channel Shadow Register (0E4024$_H$+i*400$_H$+j*40$_H$) Application Reset Value: 0000 0000$_H$
CDTMi_DTMj_CH_SR (i=5-6;j=0-3)
CDTMi DTMj Channel Shadow Register (0E4024$_H$+i*400$_H$+j*40$_H$) Application Reset Value: 0000 0000$_H$
CDTMi_DTMj_CH_SR (i=0-4;j=2-3)
CDTMi DTMj Channel Shadow Register (0E4024$_H$+i*400$_H$+j*40$_H$) Application Reset Value: 0000 0000$_H$
CDTMi_DTMj_CH_SR (i=5-6;j=4-5)
CDTMi DTMj Channel Shadow Register (0E4024$_H$+i*400$_H$+j*40$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								SL1_3_SR_SR	SL0_3_SR_SR	SL1_2_SR_SR	SL0_2_SR_SR	SL1_1_SR_SR	SL0_1_SR_SR	SL1_0_SR_SR	SL0_0_SR_SR
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SL0_0_SR_SR	0	rw	**Shadow register for bit SL0_0_SR of register DTM[i]_CH_CTRL2_SR**
SL1_0_SR_SR	1	rw	**Shadow register for bit SL1_0_SR of register DTM[i]_CH_CTRL2_SR**
SL0_1_SR_SR	2	rw	**Shadow register for bit SL0_1_SR of register DTM[i]_CH_CTRL2_SR**
SL1_1_SR_SR	3	rw	**Shadow register for bit SL1_1_SR of register DTM[i]_CH_CTRL2_SR**
SL0_2_SR_SR	4	rw	**Shadow register for bit SL0_2_SR of register DTM[i]_CH_CTRL2_SR**
SL1_2_SR_SR	5	rw	**Shadow register for bit SL1_2_SR of register DTM[i]_CH_CTRL2_SR**
SL0_3_SR_SR	6	rw	**Shadow register for bit SL0_3_SR of register DTM[i]_CH_CTRL2_SR**
SL1_3_SR_SR	7	rw	**Shadow register for bit SL1_3_SR of register DTM[i]_CH_CTRL2_SR**
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.17 Multi Channel Sequencer (MCS)

28.17.1 Overview

The Multi Channel Sequencer (MCS) sub module is a generic data processing module that is connected to the ARU. One of its major applications is to calculate complex output sequences that may depend on the time base values of the TBU and are processed in combination with the ATOM sub module. Other applications can use the MCS sub module to perform extended data processing of input data resulting from the TIM sub module. Moreover, some applications may process data provided by the CPU within the MCS sub module, and the calculated results are sent to the outputs using the ATOM sub modules.

*** 'Generic Design Parameters' on page 348 *** summarizes all available generic design parameters of the MCS hardware structure.

Table 64 Generic Design Parameters

Design Parameter	Description
W	Word width of the data path
T	Number of available MCS channels
RDW	RAM data width of connected RAM
RAW	RAM address width used by the MCS for addressing memory
USR	Use second RAM port (0 - one RAM port available, 1 - two RAM ports available)
BAW	Bus Master Address Width
BDW	Bus Master Data Width
URIP	Use RAM input pipeline registers (0 - no register, 1 - use register)
UROP	Use RAM output pipeline registers (0 - no register, 1 - use register)
UDP	Use Decoder Pipeline register (0 - no register, 1 - use register)
UAP	Use ALU Pipeline register (0 - no register, 1 - use register)
NPS	Total number of pipeline stages (with NPS = 3 + URIP + UROP + UDP + UAP)

All MCS instances in the GTM use the values T=8, W=24, RDW=32, RAW=12, USR=1, BAW = 14, BDW = 32, URIP = 1, UROP = 1, UDP = 1, UAP = 1, and NPS = 7.

28.17.2 Architecture

Figure 106 gives an overview of the MCS architecture assuming that all pipeline registers are implemented.

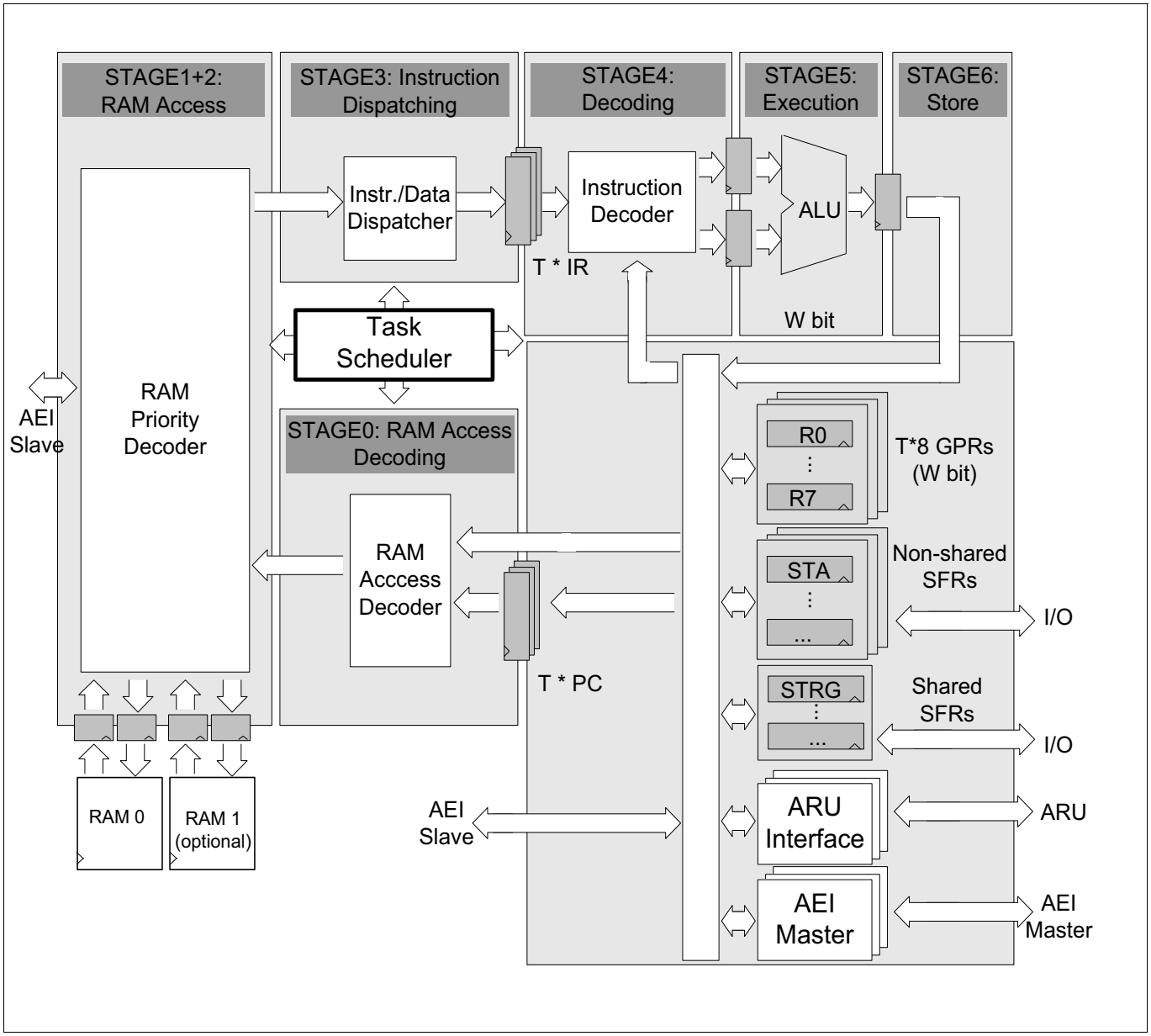


Figure 106 MCS Overview

The data path of the MCS is shared by T so called MCS channels, whereas each MCS channel executes a dedicated micro-program that is stored inside the RAM connected to the MCS module.

The connected RAM may contain arbitrary sized code and data sections that are accessible by all MCS channels and an externally connected master (e.g. a CPU) via AEI Slave interface. More details about the RAM can be found in **Section 28.17.4**

An MCS channel can also be considered as an individual task of a processor that is scheduled to the commonly used data path at a specific point in time. The execution of the different MCS channels on the different pipeline stages is controlled by a central hardware related task scheduler, which enables immediate task switches in parallel to the program execution. Details about the task scheduler and the available scheduling algorithms can be found in **Section 28.17.3**.

Typically, if data has to be exchanged between different MCS channels and/or the CPU, the connected RAM, which is accessible by all MCS channels and the CPU, can be used.

Besides the commonly used data path, each MCS channel has:

- a set of eight General Purpose Registers (GPRs), each W bit wide,
- a set of non-shared Special Function Registers (SFRs) that are only accessible within a dedicated MCS channels.
- a set shared SFRs that are accessible by all MCS channels,
- a channel specific instruction register (IR),
- a channel specific program counter register (PC),
- a dedicated ARU interface for communication with other ARU connected modules,
- and an AEI Bus Master Interface for controlling and configuration of other GTM sub modules.

Generally, the GPRs of an MCS channel x are only accessible by its corresponding MCS channel x. However, the MCS provides a configuration that allows an MCS channel x to access the GPRs of its successor MCS channel x+1. This feature can be used to enlarge the number of registers for a specific MCS channel x and/or to exchange data between neighboring channels.

For safety reasons, the register **MCS[i]_REG_PROT** can be used to define write protections for the neighboring registers of the individual MCS channels.

In order to enable synchronization between different MCS channels and/or the CPU, the MCS provides a common 24 bit wide trigger register that can be accessed as a shared SFR by all MCS channels located in the same module. Writing to **STRG** sets bits and writing to **CTRG** clears bits in the common trigger register. To enable triggering of MCS channels by CPU, the CPU can set bits in the common trigger register by writing to **MCS[i]_STRG** and clear bits by writing to **MCS[i]_CTRG**.

Considering the architecture in the figure above and assuming that all available pipeline stages are implemented (the generic parameters URIP, UROP, UDP, and UAP are set to 1), the main actions of the different pipeline stages are as follows:

- Pipeline stage 0 performs a setup of address, input data, and control signals for the next RAM access of a specific MCS channel.
- The actual RAM access of a specific MCS channel is executed in pipeline stage 1 and 2, assuming an external connection of a synchronous RAM with a latency of one clock cycle.
- Pipeline stage 3 performs pre-decoding and dispatching of instructions and data resulting from the RAM.
- In pipeline stage 4 the instructions are decoded and data from the registers are loaded.
- After that, in pipeline stage 5 the instruction is executed meaning that arithmetic operations are applied.
- Finally, in pipeline stage 6 the calculated results are stored in the registers.

If any of the pipeline registers is not implemented, the adjacent pipeline stages are merged and thus processed within the same clock cycle.

The RAM priority decoder arbitrates RAM accesses that are requested by the CPU via AEI and by the active MCS channel. If both, CPU and an MCS channel request a memory access to the same memory module the MCS channel is prioritized.

Since the internal registers of the MCS can be updated by different sources (MCS write access by various instructions, CPU write access via AEI slave, MCS write access by neighboring channel) a write conflict occurs if more than one source wants to write to the same register. In this case the result of the register is unpredictable. However, the software should setup its application in a way that such conflicts do not occur.

One exception is the common trigger register, which may be written by multiple sources (different MCS channels and CPU) in order to enable triggering of different MCS channels. Typically, the software should setup its application in a manner that different sources should not write the same bits in the trigger register.

28.17.3 Scheduling

The MCS provides a hardware related task scheduler, which globally controls the execution of the tasks in the different pipeline stages. The task scheduler implements four different scheduling modes, that can be selected by the **SCD_MODE** bit field in the **MCS[i]_CTRL_STAT** register. Depending on the selected scheduling mode, the task scheduler is selecting a dedicated MCS channel that will be executed in pipeline stage 0 in the next clock cycle. Additionally, MCS channels that are already present in the pipeline are shifted to its successor pipeline stage, with each clock cycle. This means, that the execution time of an MCS channel in a specific pipeline stage is always one clock cycle.

The MCS task scheduler may also schedule an empty cycle to pipeline stage 0, in order to grant a time slice to the CPU for accessing the connected RAM.

It should be noted, if the task scheduler assigns an MCS channel to pipeline stage 0, but this channel does not access the RAM, the CPU can access the corresponding RAM, even if the scheduler did not reserve an empty clock cycle.

In the following, the available scheduling modes are described.

28.17.3.1 Round Robin Scheduling

The Round Robin Scheduling Mode implements the simplest scheduling algorithm. This algorithm schedules a predefined set of MCS channels in the range [0; **SCD_CH**] in ascending order. After the last channel **SCD_CH** has been assigned to the pipeline, an empty cycle is scheduled in order to enable RAM access for the CPU. The parameter **SCD_CH** can be controlled by the register **MCS[i]_CTRL_STAT**. If the value of **SCD_CH** is greater than T-1, the scheduler assumes a value of T-1 for bit field **SCD_CH**.

Figure 107 shows a timing example of the Round Robin Scheduling with T=8 MCS channels (marked as C_0 to C_7) that are scheduled together with a CPU access to a pipeline with and NPS=7 stages. It is assumed that bit field **SCD_CH** is set to 7.

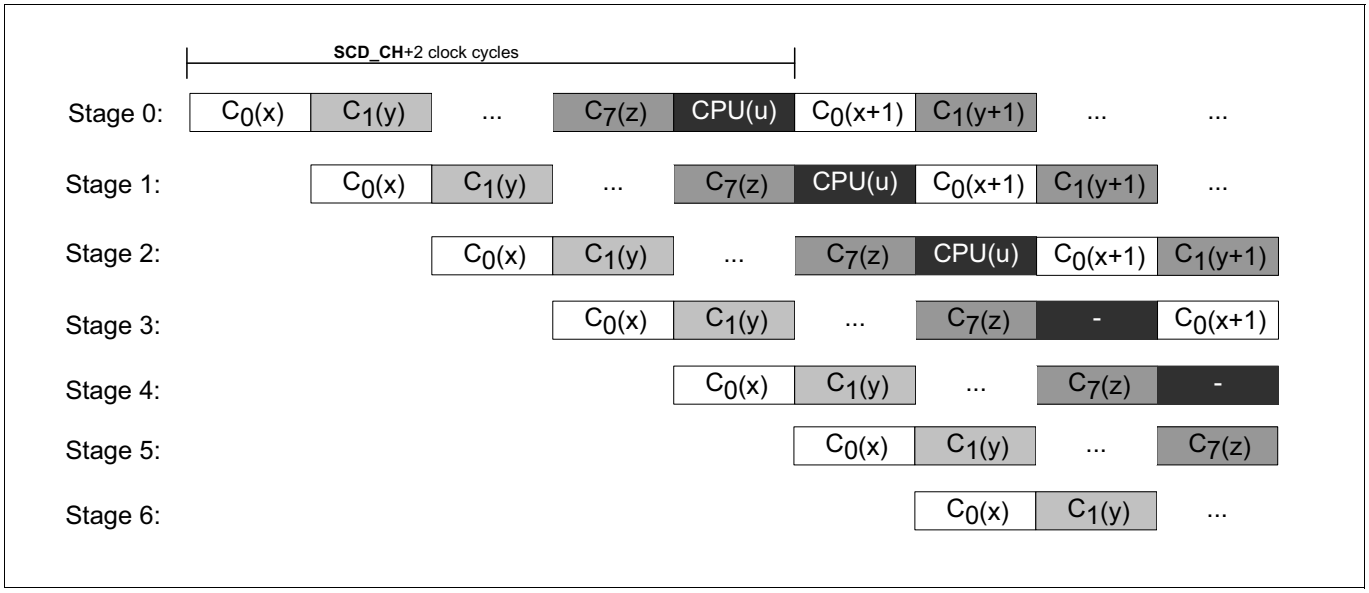


Figure 107 Timing of Round Robin Scheduling

The identifier $C_i(x)$ denotes that MCS channel i is currently executing the instruction or data located in the memory at position x in the corresponding pipeline stage. The figure shows, which MCS channel is activated in specific pipeline stage at a specific point in time.

Moreover, the figure shows that the Round Robin scheduling is always repeated after **SCD_CH**+2 clock cycles, which means that the time duration of an instruction cycle is **SCD_CH**+2 clock cycles. However, if the value

SCD_CH + 2 is less than NPS, the duration of an instruction cycle is limited by the depth of the pipeline to NPS clock cycles. Thus the effective execution time of a single cycle instruction is always MIN(**SCD_CH**+2, NPS) clock cycles, ignoring the latency of the pipeline.

If NPS is greater than T+1, NPS-T-1 additional empty cycles are inserted at the end of a round trip cycle. In this case the round trip time for the scheduler is determined by NPS, and thus the time duration for an instruction cycle is always NPS clock cycles.

The Round Robin scheduling algorithm has the characteristic that it fairly distributes all time slices to all MCS channels and the CPU. This means, that the program execution time of a specific task is independent from the activity of any neighboring task or the CPU RAM access, and thus a correct estimation of the actual program execution time is very easy. However, the round-robin scheduling may waste clock cycles by scheduling MCS channels that are not ready to execute an instruction (e.g. MCS channel is disabled by CPU). The following scheduling modes overcome this issue.

28.17.3.2 Accelerated Scheduling

In order to improve the computational performance, the accelerated scheduling mode provides two key features. Firstly, the scheduler only selects MCS channels that are not suspended and thus can actually execute an instruction. Secondly, the scheduler applies instruction prefetching to minimize empty cycles in the pipeline. An MCS channel is entering suspended state due to one of the reasons:

- An MCS channel is executing a read or write request to an ARU connected sub module (instruction ARD, AWR, ARDI, AWRI, NARD, NARDI).
- An MCS channel is executing a read or write request at its bus master interface (instruction BRD, BWR, BRDI, BWRI).
- An MCS channel waits on a register match event (e.g. instruction WURM), in order to wait on a desired register value (e.g. trigger event from another MCS channel).
- An MCS channel is disabled.

In the case of instruction prefetching, the scheduler will assign an MCS channel C_p to pipeline stage 0, which is already present in another pipeline stage. This means, that the execution of the last instruction of C_p located in the memory MEM(PC/4) is not yet finished completely, whereas PC is the current value of the program counter of MCS channel C_p. Thus, the newly scheduled MCS channel C_p will prefetch a successor instruction MEM(PC/4+PFO) under the assumption that there will be no branch and no memory access in the program between the instructions MEM(PC/4) and MEM(PC/4+PFO). The prefetch offset value PFO is determined by counting the number of already scheduled MCS channels Cp in the pipeline. However, if the assumption fails, the pipeline will be flushed by replacing all MCS channel C_p of the pipeline with an empty cycle, as soon as the instruction decoder detects a branch or a memory access. All other MCS channels unequal to MCS channel C_p within are not affected by the flushing action. The flushing action is always synchronized to the last pipeline stage NPS-1.

Besides the flushing conditions mentioned above, there exist also other conditions that cause a flush of the pipeline for a specific MCS channel. In the following all possible flushing events are summarized:

- An MCS channel is enabled.
- An MCS channel is entering a suspended state.
- An MCS channel is taking a conditional or unconditional branch (instruction JMP, JBS, JBC, CALL, RET, JMPI, JBSI, JBCI, CALLI).
- An MCS channel accessing memory for data transfer (instruction MRD, MWR, MRDI, MWRI, MRDIO, MWRIO, MWRL, MWRIL, PUSH, POP).
- An MCS channel is executing a read or write request at its bus master interface (instruction BRD, BWR, BRDI, BWRI).

- An MCS channel is modifying the trigger register (write access to **CTRG** or **STRG**) and the same channel is reading back this register (read access to **CTRG** or **STRG**) while the delay between both accesses is less than UAP+UDP+1 clock cycles.

In general, each MCS channel can accept instruction prefetching. However, there are some cases in which an upcoming flushing of the pipeline can be easily detected by the MCS hardware due to evaluation of internal states. Therefore, it is defined that an MCS channel accepts instruction prefetching only under the following conditions:

- An MCS channel is currently not in the second cycle of a two-cycle control flow instruction (instruction CALL, RET).
- An MCS channel is currently not in the second cycle of a three-cycle memory access instruction (instruction MWRL, MWRIL).

The accelerated scheduling mode guarantees, that the time duration of an instruction cycle varies between 1 and T+1 cycles. Hence, a single cycle instructions has an effective execution time between 1 to T+1 clock cycles, depending on the number of suspended MCS channels and the actual instruction sequence. The worst case execution time occurs if all channels are active and the CPU also accesses the RAM. The best case occurs e.g. if only one MCS channel is enabled and the executed program sequence has only linear code without branches and memory access.

The algorithm of the accelerated scheduling mode first, evaluates the state of all available MCS channels as well as a CPU request to the RAMs and then it decides if a specific MCS channel or an empty cycle is assigned to pipeline stage 0 in the next clock cycle. It should be noted that the accelerated scheduling mode treats RAM access requests from the CPU in a similar manner as MCS channels, which means that empty cycles for RAM requests are only inserted into the pipeline if there is an active RAM request from the CPU or no other task can be scheduled.

In order to fairly trade all available MCS channels as well as CPU RAM requests and to guarantee a worst case execution time of T+1 clock cycles, an additional task prioritization scheme is applied used that dynamically prioritizes all MCS channels and a CPU memory access depending on the history of the scheduler's decisions. The algorithm of the accelerated scheduler mode is executed every clock cycle and it works in the following manner:

1. Try to find an MCS channel C_r with highest priority that is not suspended and not already scheduled to the pipeline stages 0 to NPS-2. If C_r is found assign C_r to pipeline stage 0 and finish scheduling for current clock cycle.
2. Otherwise, try to find an MCS channel C_p with highest priority that is not suspended and accepts instruction prefetching. If C_p is found assign C_p to pipeline stage 0 and finish scheduling for current clock cycle.
3. Otherwise, try to find an MCS channel C_s with highest priority that is suspended and accepts instruction prefetching. If C_s is found assign C_s to pipeline stage 0 and finish scheduling for current clock cycle.
4. Otherwise, assign an empty cycle to pipeline stage 0 and finish scheduling for current clock cycle.

The underlying task prioritization scheme tracks the history of the scheduled MCS channels in a list consisting of T+1 items. The list is initialized with all MCS channels followed by a reserved time slot for the CPU RAM access. The position of an MCS channel within this list implicitly defines the priority, while the back of this list holds the MCS channel with highest priority. Whenever the scheduling algorithm described above has found an MCS channel C_r or C_p to be scheduled in the next clock cycle, it removes this item from the list and put it to the front of the list. In order to fairly prioritize all MCS channels, the algorithm also removes the item at the back of the list to the second position in the list, after the inserted scheduled front item. Since the list always contains all possible MCS channels and with each clock cycles each nonscheduled item is moved at least one position towards the end of list, it is obvious that each MCS channel will have the highest priority not later than T+1 clock cycles.

Figure 108 shows a timing example of the accelerated scheduling with NPS=7 pipeline stages.

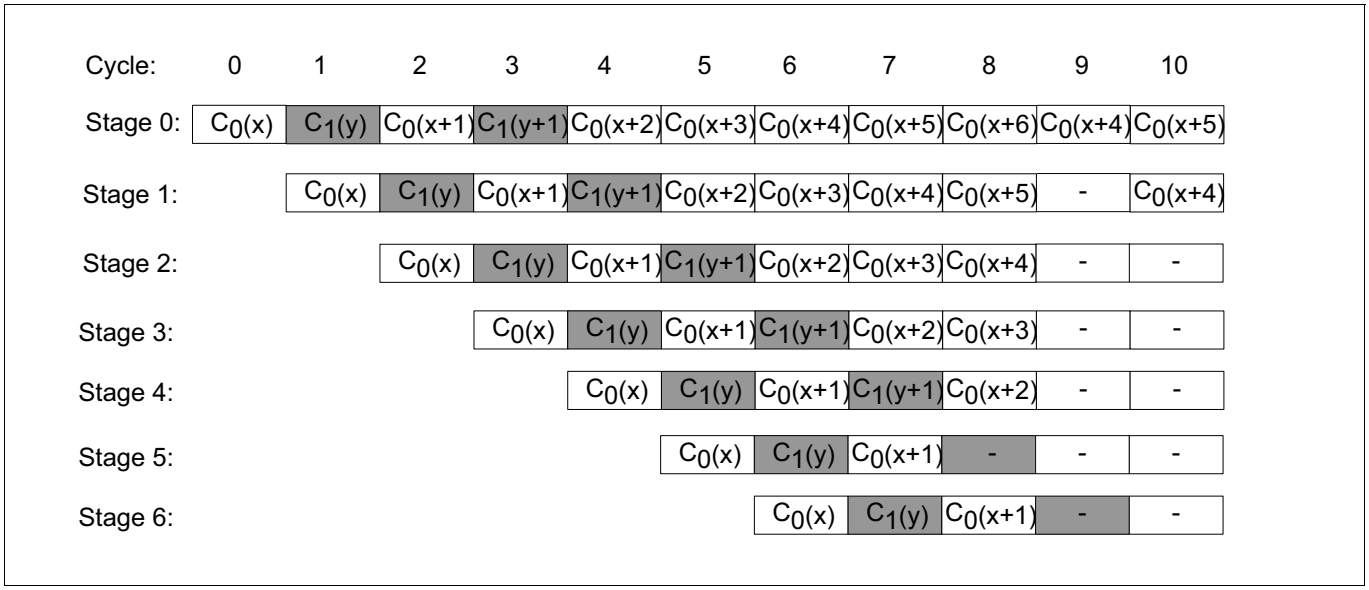


Figure 108 Timing of Accelerated Scheduling

The example assumes that initially MCS channels 0 and 1 are enabled and the program for each MCS channel is located in the RAM as shown in *** 'MCS Code example for Accelerated Scheduling' on page 354 ***.

MCS-Channel 0

Memory Location	Instruction
x+0	ADDL R0, 7
x+1	JBC STA, Z, 4*(x+4)
x+2	MOVL R2, 5

MCS-Channel 1

Memory Location	Instruction
y+0	WURM STRG, R1, 0
y+1	ADDL R2, 5
y+2	ADD R3, R2

Figure 109 MCS Code example for Accelerated Scheduling

Since both channels are ready to run, the scheduler fairly selects the channels in an alternating order, as it can be obtained in stage 0 at the clock cycles before cycle 5. Since MCS channel 1 is entering suspended state (to wait on a trigger bit) at cycle 7 in stage 6 with the instruction of memory location y, the scheduler will only select MCS channel 0 in the following by applying instruction prefetching. Moreover, entering the suspended state in channel 1 also flushes the remaining channels 1 out of the pipeline. But it should be noted, the scheduler applies instruction prefetching during the whole sequence, due to the fact that the number of enabled channels is always less than the available number of pipeline stages NPS.

The actual state of pipeline in cycle 9 and 10 depends on conditional branch instruction of memory location x + 3 (cycle 8 stage 6). If the branch is not taken, the linear code execution of MCS channel 0 is continued. However, if the branch to memory location x+4 is taken, as shown in the Figure, the scheduler will fetch the instruction C0(x+4) in cycle 9 at stage 0 and flush the stages 1 to NPS-1. Note, the flushing of the pipeline only concerns the prefetched instructions of the MCS channel that is currently executed in the last stage. If pipeline stage 1 of cycle 9 would belong to another channel than 0, only the stages greater than 2 would have been flushed.

28.17.3.3 Single Prioritization Scheduling

The Single Prioritization Scheduling mode is an extended variant of the Accelerated Scheduling mode, which additionally applies a task prioritization of a single MCS channel. In this mode, the bit field **SCD_CH** of register **MCS[i]_CTRL_STAT** is used to identify a dedicated MCS channel that is always preferred during scheduling. This

means, that the scheduler will assign preferred MCS channel **SCD_CH** to pipeline stage 0, as long as this channel is not suspended. If the preferred MCS channel is entering its suspended state, the scheduling algorithm switches to the accelerated scheduling as previously described in **Section 28.17.3.2**. Whenever the MCS channel **SCD_CH** is resuming from its suspended state, the scheduler switches back and assign the channel **SCD_CH** to pipeline stage 0 until the next suspension event occurs. If the bit field **SCD_CH** contains the value T or higher, the task scheduler will always prioritize CPU access to the RAM. This means, whenever the task scheduler detects that the CPU wants to access an MCS-RAM, the scheduler will assign an empty cycle into pipeline stage 0. If the CPU does not access the RAM any more, it switches back to the accelerated mode, as described previously in **Section 28.17.3.2**.

In consequence, the Single Prioritization Scheduling mode cannot guarantee a maximum time duration of an instruction cycle for the overall execution of all MCS channels, since it strongly depends on the activity of the prioritized MCS channel **SCD_CH**. However, the Single Prioritization Scheduling mode provides the fastest possible execution for MCS channel **SCD_CH**. Moreover, during the time spawn, in which the prioritized MCS channel **SCD_CH** is suspended, this mode guarantees a duration of 1 to T+1 clock cycles of an instruction cycle for all non-prioritized channels.

28.17.3.4 Multiple Prioritization Scheduling

The Multiple Prioritization Scheduling mode is an extended variant of the Accelerated Scheduling mode, which additionally applies a task prioritization for multiple MCS channels. In this mode, the bit field **SCD_CH** of register **MCS[i]_CTRL_STAT** is used to identify a set of dedicated MCS channels, which are always preferred during scheduling. The identifiers of the prioritized MCS channels are in the range [0; **SCD_CH**] and the non-prioritized channels are in the range [**SCD_CH**+1; T-1]. The individual priority for the set of prioritized MCS channels is applied in descending order, which means that MCS channel 0 has the highest priority, followed MCS channel 1, which has the second highest priority, and so on. The non-prioritized MCS channels do not have any priority. A value of T-1 or higher for the bit field **SCD_CH** means that all T MCS channels are prioritized MCS channels.

With each clock cycle, the Multiple Prioritization Scheduling mode will assign the non-suspended MCS channel with the highest priority from the set of prioritized MCS channels to pipeline stage 0, as long as there are non-suspended prioritized MCS channels available. If all prioritized MCS channels are suspended, the scheduling algorithm switches to the accelerated scheduling as previously described in **Section 28.17.3.2** and it schedules the non-prioritized channels. Whenever a prioritized MCS channel is resuming from its suspended state, the scheduler switches back and applies the described prioritization scheme until the next suspension event of occurs.

In consequence, the Multiple Prioritization Scheduling mode cannot guarantee a maximum time duration of an instruction cycle for the overall execution of all MCS channels, since it strongly depends on the activity of the prioritized MCS channels. However, the Multiple Prioritization Scheduling mode provides the fastest possible execution for prioritized MCS channels. Moreover, during the time spawn, in which all prioritized MCS channels are suspended, this mode guarantees a duration of 1 to T+1 clock cycles of an instruction cycle for all non-prioritized channels.

28.17.4 Memory Organization

The MCS module supports a memory layout of up to $2^{RAW+USR}$ memory locations each RDW bit wide leading to a maximum byte wise address range from 0 to $2^{RAW+USR+2}$-1.

If two RAM ports are used (USR = 1) the entire address space of the MCS is divided into two seamless memory pages. Further, if the GTM provides a memory configuration sub module (MCFG), memory page 0 begins from (byte wise) address 0 and ranges to address MP0-4 and memory page 1 ranges from MP0 to MP1-4, while MP0 and MP1 are configuration parameters provided by MCFG.

The RAM priority decoder of the MCS will always handle a RAM access from an MCS channel with a higher priority compared to a RAM access from AEI.

However, if a set of active MCS channels are only accessing one common RAM port, the MCS will grant any AEI accesses to the other RAM port in parallel to the related RAM accesses of the running MCS channels, which means that AEI may get the full bandwidth to a dedicated RAM.

Basically, the actual access time to the RAMs via AEI depends on the actual scheduling mode and the activity of tasks. In the modes Round Robin Scheduling and Accelerated Scheduling the scheduler guarantees a maximum write access time of T + 4 clock cycles and a maximum read access time of T + 6 clock cycles. In the scheduling modes Single Prioritization Scheduling and Multiple Prioritization Scheduling, the scheduler cannot guarantee a maximum access time for AEI RAM access.

Depending on the silicon vendor configuration, the connected RAM pages are initialized with zeros in the case of an MCS module reset.

If an ECC Error occurs while an MCS channel reads data from a memory module, the corresponding MCS channel is disabled and the ERR bit in register STA is raised.

If the GTM sub module CCM provides several so called address range protectors (ARPs), some code and data sections of the MCS RAM can be write protected. If an MCS channels x writes to such a protected memory region, the MCS channel x is halted, the **ERR** bit in register **STA** is set and the bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT** is updated.

28.17.5 AEI Bus Master Interface

The MCS module provides an AEI bus master interface, which enables to communicate with externally connected modules. The data width of this interface is BDW bit and the address width is BAW bit leading to a maximum byte wise address ranging from 0 to 2^{BAW+2}-1. The bus master interface is shared among all available MCS channels meaning that each MCS channel may initiate a read or write access on the bus but only one channel can be served at a specific point in time.

However, the AEI bus master interface guarantees, that a bus access is always completed within two instruction cycles and bus access of different MCS channels do not modify the latency of each other. The only exceptions are bus accesses to RAM modules (e.g. accessing memory location in a DPLL RAM or FIFO RAM). AEI bus master accesses to RAM modules cannot be completed within a single clock cycle and thus additional wait cycles have to be inserted into the bus protocol leading to the fact that the MCS channel that is accessing the RAM is entering a suspended state. Moreover, if an MCS channel is accessing a RAM module, the latency of a bus access in another MCS channel can also be modified even if the neighboring channel is accessing only a configuration register.

The AEI bus master interface of an MCS module is connected to AEI slave interface GTM-IP in order to control the sub modules of the GTM within MCS. However, it is not possible to access the entire GTM by a single MCS module. The n-th MCS instance can only access the GTM sub modules that are located within the n-th cluster of the GTM. Details about the available clusters of the GTM can be found inside the “GTM Architecture” Overview chapter.

Additionally, the address map for accessing GTM with the AEI bus master interface of an MCS differs from the address map for an externally connected CPU that is using the GTM's AEI slave interface. The address map for accessing GTM with the AEI bus master interface of an MCS can be found in device specific appendix.

Since the sub modules of the GTM can be accessed by the CPU and the AEI bus master of an MCS, the GTM-IP provides an additional arbitration scheme to manage parallel accesses from both master interfaces. If CPU and an MCS want to access a GTM sub module of the same cluster, the arbiter will grant the access to the MCS. However, if the CPU and all the MCS instances want to access GTM sub modules of different clusters, the accesses can be executed in parallel.

The AEI bus master interface can be controlled by the MCS instructions BRD, BRDI, BWR, and BWRI. These instructions are described in **Section 28.17.7**.

28.17.6 ADC Interface

The clusters of the GTM can optionally provide a dedicated interface for the connection of up to 32 external Analog-Digital-Converter (ADC) channels, which can be mapped arbitrarily to physical instances of single- or multi-channel ADCs.

An ADC Interface is directly mapped into the address map of the AEI bus master interface of the current cluster's MCS meaning that the available AEI bus master instructions (BRD and BRDI as described in **Section 28.17.7**) are used to control the connected ADCs.

Since the control of the connected ADCs is silicon vendor specific, the GTM specification does not provide a complete specification for controlling connected ADCs. However, to ensure software compatibility at least for the basic features of an ADC, the functionality described in the following are common to all silicon vendors.

28.17.6.1 Basic ADC Functions

The address map of the AEI bus master interface reserves two unique address items for each ADC channel. The address items can be referred by the labels **ADC_CH[y]_DATA** and **ADC_CH[y]_STA** for the channel y in the range from 0 to 31. The actual address can be found in the appendix.

The MCS can read from address **ADC_CH[y]_DATA** in order to get the conversion result of the ADC that is connected to ADC channel y. The conversion result is represented as a signed 24 bit value and it is stored in the register A (A $\in$ GREG) as referred by the corresponding MCS instruction BRD or BRDI. Additionally, each read access to **ADC_CH[y]_DATA** triggers the ADC that is connected to channel y. Any read access to **ADC_CH[y]_DATA** also provides 8 status bits that are stored in register **MHB**. The bit **MHB**[7] has always the mnemonic **ADC_ACK** and the bit **MHB**[6] has always the mnemonic **ADC_DATA_DATA**. If bit **ADC_ACK** is set the result of the data conversion (register A) and the corresponding status bits (bits **MHB**[6:0]) are validated. If **ADC_NEW_DATA** is set the current conversion result is new and has never been read by a previous bus read access. The meaning of the bits **MHB**[5:0] are vendor specific. Otherwise, if **ADC_ACK** is cleared the read data is invalid and the **MHB**[4:0] indicate the channel identifier (with **MHB**[4:0] <> y) that is currently processed by the ADC. A write access to **ADC_CH[y]_DATA** has no functionality and is always ignored.

The MCS can read from address **ADC_CH[y]_STA** to get additional 31-bit wide vendor specific status information of ADC channel y. The lower 24 bits of the status information is stored in register A (A $\in$ GREG) as referred by the corresponding MCS instruction BRD or BRDI. The upper 7 bit of the status information is stored in register **MHB**[6:0]. The bit **MHB**[7] has always the mnemonic **ADC_ACK**. If bit **ADC_ACK** is set the result of status information in register A and bits **MHB**[6:0] are validated. Otherwise, if **ADC_ACK** is cleared the status information is invalidated. A write access to **ADC_CH[y]_STA** has no functionality and is always ignored.

Any read or write access to a register **ADC_CH[y]_STA** or ADC_CH[y]_DATA updates the AEI status signal that is evaluated in the sub module CCM. The following status information is defined for the AEI status values:

- 00_B: no error occurred
- 01_B: optional information register not implemented (only register **ADC_CH[y]_STA**)
- 10_B: illegal ADC access (e.g. ADC not enabled)
- 11_B: unsupported address (ADC channel y not available)

Note: *Note: If the received status AEI is unequal to "00" ADC_NEW_DATA is always set and ADC_ACK is always cleared.*

28.17.7 Instruction Set

This section describes the entire instruction set of the MCS sub module. First, a brief overview over all available instructions is given and a detailed description of each instruction can be found in **Section 28.17.7.1** and the following sections.

In general, each instruction is RDW bit wide but the duration of each instruction varies between several instruction cycles. As already described in **Section 28.17.3**, the number of required clock cycles for an instruction cycle can be fixed or variable, depending on the selected scheduling mode. In the case of the Round Robin Scheduling, the duration is fixed with T+1 clock cycles, in the case of the Accelerated Scheduling the duration is variable in the range between 1 and T+1 clock cycles, and in all other Scheduling modes the duration is also variable and may even be more than T+1 clock cycles, depending on the application.

Before the available instructions are described, some commonly used terms, abbreviations and expressions are introduced:

OREG: The operation register set OREG = {R0, R1…R7} || {STA, ACB, CTRG, STRG, TBU_TS0, TBU_TS1, TBU_TS2, MHB} include all MCS accessible internal channel specific GPRs {R0, R1…R7} and the sub set {STA, ACB, CTRG, STRG, TBU_TS0, TBU_TS1, TDU_TS2, MHB} of SFRs.

XOREG: The extended operation register set XOREG = OREG || {RS0, RS1…RS7} || {GMI0, GMI1, DSTA, DSTAX} extends the operation registers set OREG by the GPRs of the succeeding MCS channel {RS0, RS1… RS7} and the SFRs {GMI0, GMI1, DSTA, DSTAX}.

WXREG: The extended wait instruction operation register set WXREG = OREG || {GMI0, GMI1, DSTA, DSTAX} extends the operation registers set OREG by the SFRs {GMI0, GMI1, DSTA, DSTAX}.

AREG: The ARU register set AREG = {R0, R1, R2…R7, ZERO} includes the all registers that can be written by incoming ARU transfers (ARD, ARDI, NARD, and NARDI instructions). These registers include all eight general purpose registers. The dummy register ZERO may be used to discard an incoming 24 bit ARU word.

GREG: The general purpose register set GREG = {R0, R1, R2…R7} includes the all channel specific GPRs without GPRs of neighboring channels.

BAREG: The base address register set BAREG = OREG || {RS0, RS1…RS7} extends the register set OREG by the GPRs of neighboring channels.

Note: If the extended operation register set XOREG is disabled (bit **EN_XOREG** of register **MCS[i]_CTRL_STAT** is cleared) the sets **XOREG**, **WXREG**, and **BAREG** only contains the operation register set OREG.

Note: In the following, the register sets **OREG**, **XOREG**, **GREG, WXREG**, **BAREG** and **AREG** are referred by the instructions. Typically, an operation announces W data bits. Whenever, a register of a register set implements less than W bits, it is assumed that these register bits only define the LSBs of an operation. The missing MSBs are always read and written as zeros.

WLIT: The set WLIT = {0,1…2^{W}-1} is a W bit wide literal value used for encoding immediate operands.

ALIT: The set ALIT = {0,1…$2^{RAW+USR}$-1} is a RAW + USR bit wide literal value used for encoding memory addresses.

AOLIT: The set AOLIT = {-$2^{RAW+USR-1}$…-1,0,1…$2^{RAW+USR-1}$-1} is a RAW + USR bit wide literal value used for encoding relative memory address offsets.

ARDLIT: The set ARDLIT = {0,1…2^{9}-1} is a 9 bit literal used for ARU read addresses.

AWRLIT: The set AWRLIT = {0,1…23} is used as ARU write indexes, selecting one of the 24 ARU write address.

BALIT: The set BALIT = {0,1…2^{BAW}-1} is a BAW bit wide literal used for encoding bus master addresses.

SFTLIT: The set SFTLIT = {0,1…W} is used as literal value for shift instructions.

BWSLIT: The set BWSLIT = {1…W} is used as literal value for multiplication instructions.

BITLIT: The set BITLIT = {0,1…15} is a 4 bit literal used for bit indexing.

XBITLIT: The set XBITLIT = {0,1…W-1} is a literal used for bit indexing of register bits.

MSKLIT: The set MSKLIT = {0,1…2^{15}-1} is a 16 bit literal used for bit-masking.

BIT SELECTION: The expression VAR[i] represents the i-th bit of a variable VAR.

BIT RANGE SELECTION: The expression VAR[m:n] represents the bit slice of variable VAR that is ranging from bit n to bit m.

MEMORY ADDRESSING: The expression MEM(X) represents the RDW bit wide value at location x (x ∈ ALIT) of the memory. The expression MEM(x)[m:n] represents the bit slice ranging from bit n to m of the RDW bit wide word at memory location x.

ARU ADDRESSING: In the case of ARU reading, the expression ARU(x) represents the 2*W+5 bit wide ARU word of ARU channel at read address x (x ∈ ARDLIT). In the case of ARU writing, the expression ARU(x) represents a 2*W+5 bit wide ARU word that is written to an ARU channel indexed by the index x (x ∈ AWRLIT). The index x selects a single ARU write channel from the pool of the MCS sub module's allocated ARU write channels. An MCS sub module has 24 dedicated ARU write channels, indexed by values 0 to 23. The expression ARU(x)[m:n] represents the bit slice ranging from bit n to m of the 2*W+5 bit wide ARU word.

BUS MASTER ADDRESSING: In the case of reading/writing from the bus master interface, the expression BUS(x) represents the BDW bit wide data word that is read/written at address x (x ∈ BALIT). The expression BUS(x)[m:n] represents the bit slice ranging from bit n to m of the BDW bit wide data word at the bus.

Figure 110 ff. summarize the entire instruction set of the MCS and **Table 113** ff. shows the encoding of the individual instructions.

Class	Mnemonic	Operation	instruction cycles	Synopsis
Data transfer	MOVL A, C	A ← C	1	Move Literal, A in OREG, C in WLIT
	MOV A, B	A ← B	1	Move, A in XOREG, B in XOREG
	MRD A, C	A ← MEM(C)[W-1:0]; MHB ← MEM(C)[RDW-1:W]	2[1)]	Memory Read, A in OREG, C in ALIT
	MWR A, C	MEM(C)[W-1:0] ← A; MEM(C)[RDW-1:W] ← MHB	2[1)]	Memory Write, A in OREG, C in ALIT
	MRDI A, B [, C]	A ← MEM(B[RAW+USR+1:2]+C)[W-1:0]; MHB ← MEM(B[RAW+USR+1:2]+C)[RDW-1:W]	2[1)]	Memory Read Indirect, A in OREG, B in OREG, C in AOLIT (default C=0)
	MWRI A, B [, C]	MEM(B[RAW+USR+1:2]+C)[W-1:0] ← A; MEM(B[RAW+USR+1:2]+C)[RDW-1:W] ← MHB	2[1)]	Memory Write Indirect, A in OREG, B in OREG, C in AOLIT (default C=0)
	MRDIO A, B	A ← MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[W-1:0]; MHB ← MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[RDW-1:W]	2[1)]	Memory Read Indirect with Offset, A in XOREG, B in BAREG
	MWRIO A, B	MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[W-1:0] ← A; MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[RDW-1:W] ← MHB	2[1)]	Memory Write Indirect with Offset, A in XOREG, B in BAREG
	POP A	A ← MEM(R7[RAW+USR+1:2]]); MHB ← MEM(R7[RAW+USR+1:2])[RDW-1:W] R7 ← R7 - 4	2[1)]	Pop from stack, A in OREG
	PUSH A	R7 ← R7 + 4 MEM(R7[RAW+USR+1:2])[W-1:0] ← A MEM(R7[RAW+USR+1:2])[RDW-1:W] ← MHB	2[1)]	Push to stack, A in OREG
	MWRL A, C	MEM(C)[W-1:0] ← A	3[2)]	Memory Write Literal, A in OREG, C in ALIT
	MWRIL A, B	MEM(B[RAW+USR+1:2])[W-1:0] ← A	3[2)]	Memory Write Indirect Literal, A in OREG, B in OREG
ARU Transfer	ARD A, B,C	A ← ARU(C)[W-1:0] B ← ARU(C)[2*W-1:W] ACB ← ARU(C)[5+2*W:2*W]	>= 1	Blocking ARU Read, A in AREG, B in AREG, C in ARDLIT
	AWR A, B, C	ARU(C)[W-1:0] ← A ARU(C)[2*W:W] ← B ARU(C)[5+2*W:2*W] ← ACB	>= 1	Blocking ARU Write, A in OREG, B in OREG, C in AWRLIT
	ARDI A, B	A ← ARU(R6[8:0])[W-1:0] B ← ARU(R6[8:0])[2*W-1:W] ACB ← ARU(R6[8:0])[5+2*W:2*W]	>= 1	Blocking ARU Read Indirect, A in AREG, B in AREG
	AWRI A, B	ARU(R6[4:0])[W-1:0] ← A ARU(R6[4:0])[2*W-1:W] ← B ARU(R6[4:0])[5+2*W:2*W] ← ACB	>= 1	Blocking ARU Write Indirect, A in OREG, B in OREG
	NARD A, B, C	A ← ARU(C[8:0])[W-1:0] B ← ARU(C[8:0])[2*W-1:W] ACB ← ARU(C[8:0])[5+2*W:2*W]	>= 1[7)]	Non-Blocking ARU Read, A in AREG, B in AREG
	NARDI A, B	A ← ARU(R6[8:0])[W-1:0] B ← ARU(R6[8:0])[2*W-1:W] ACB ← ARU(R6[8:0])[5+2*W:2*W]	>= 1[7)]	Non-Blocking ARU Read Indirect, A in AREG, B in AREG
Bus Master	BRD A, C	A ← BUS(C)[W-1:0] MHB ← BUS(C)[BDW-1:W]	>= 1[8)]	Bus Master Read, A in GREG, C in BALIT
	BWR A, C	BUS(C)[W-1:0] ← A BUS(C)[BDW-1:W] ← MHB	>= 1[8)]	Bus Master Write, A in GREG, B C in BALIT
	BRDI A, B	A ← BUS(B[BAW+1:2])[W-1:0] MHB ← BUS(B[BAW+1:2])[BDW-1:W]	>= 1[8)]	Bus Master Read Indirect, A in GREG, B in GREG
	BWRI A, B	BUS(B[BAW+1:2])[W-1:0] ← A BUS(B[BAW+1:2])[BDW-1:W] ← MHB	>= 1[8)]	Bus Master Write Indirect, A in GREG, B in GREG

Figure 110 Instruction Set Summary (part 1)

Class	Mnemonic	Operation	instruction cycles	Synopsis
Arith. / Logic	ADDL A, C	A ← A + C	1	Add Literal, A in OREG, C in WLIT
	ADD A, B	A ← A + B	1	Add, A in XOREG, B in XOREG
	ADDC A, B	A ← A + B + CY	1	Add with carry, A in XOREG, B in XOREG
	SUBL A, C	A ← A - C	1	Subtract Literal, A in OREG, C in WLIT
	SUB A, B	A ← A - B	1	Subtract, A in XOREG, B in XOREG
	SUBC A, B	A ← A - B - CY	1	Subtract with carry, A in XOREG, B in XOREG
	NEG A, B	A ← − B	1	Negate, A in XOREG, B in XOREG
	ANDL A, C	A ← A AND C	1	AND Litral, A in OREG, C in WLIT
	AND A, B	A ← A AND B	1	AND, A in XOREG, B in XOREG
	ORL A, C	A ← A OR C	1	OR Literal, A in OREG, C in WLIT
	OR A, B	A ← A OR B	1	OR, A in XOREG, B in XOREG
	XORL A, C	A ← A XOR C	1	XOR Literal, A in OREG, C in WLIT
	XOR A, B	A ← A XOR B	1	XOR, A in XOREG, B in XOREG
	SETB A, B	A[B[4:0]] ← 1	1	Set Bit, A in XOREG, B in XOREG
	CLRB A, B	A[B[4:0]] ← 0	1	Clear Bit, A in XOREG, B in XOREG
	XCHB A, B	A[B[4:0]] ↔ CY	1	Exchange Bit with CY, A in XOREG, B in XOREG
	SHR A, C	A ← A >> C	1	Shift Right, A in XOREG, C in SFTLIT
	SHL A, C	A ← A << C	1	Shift Left, A in XOREG, C in SFTLIT
	ASRU A, B	A ← A >> B	1	Shift Right, A in XOREG, B in XOREG
	ASRS A, B	A ← A >> B	1	Shift Right, A in XOREG, B in XOREG
	ASL A, B	A ← A << B	1	Shift Left, A in XOREG, B in XOREG
	MULU A, B[, C]	[[R4,] A] ← A[(C-1):0] * B[(C-1):0]	1	Multiply Unsigned, A in XOREG, B in XOREG, C in BWSLIT (default C=W)
	MULS A, B[, C]	[[R4,] A] ← A[(C-1):0] * B[(C-1):0]	1	Multiply Signed, A in XOREG, B in XOREG , C in BWSLIT (default C=W)
	DIVU A, B[, C]	R4 ← A[(C-1):0] - B[(C-1):0] * ⌊A[(C-1):0] / B[(C-1):0]⌋; A ← ⌊A[(C-1):0] / B[(C-1):0]⌋	C[9)]	Divide Unsigned, A in XOREG, B in XOREG, C in BWSLIT (default C=W)
	DIVS A, B[, C]	R4 ← A[(C-1):0] - B[(C-1):0] * ⌊A[(C-1):0] / B[(C-1):0]⌋; A ← ⌊A[(C-1):0] / B[(C-1):0]⌋	C+4[10)]	Divide Signed, A in XOREG, B in XOREG, C in BWSLIT (default C=W)
	MINU A, B	A ← MIN(A, B)	1	Minimum Unsigned, A in XOREG, B in XOREG
	MINS A, B	A ← MIN(A, B)	1	Minimum Signed, A in XOREG, B in XOREG
	MAXU A, B	A ← MAX(A, B)	1	Maximum Unsigned, A in XOREG, B in XOREG
	MAXS A, B	A ← MAX(A, B)	1	Maximum Signed, A in XOREG, B in XOREG
Test	ATUL A, C	A < C ⇔ CY is set A = C ⇔ Z is set	1	Arithmetic Test Unsigned Literal, A in OREG, C in WLIT
	ATU A, B	A < B ⇔ CY is set A = C ⇔ Z is set	1	Arithmetic Test Unsigned, A in XOREG, B in XOREG
	ATSL A, C	A < C ⇔ CY is set A = C ⇔ Z is set	1	Arithmetic Test Signed Literal, A in OREG, C in WLIT
	ATS A, B	A < B ⇔ CY is set A = C ⇔ Z is set	1	Arithmetic Test Signed, A in XOREG, B in XOREG
	BTL A, C	A AND C	1	Bit Test Literal, A in OREG, C in WLIT
	BT A, B	A AND B	1	Bit Test, A in XOREG, B in XOREG

Figure 111 Instruction Set Summary (part 2)

Class	Mnemonic	Operation	instruction cycles	Synopsis
Control Flow	JMP C	PC ← C << 2	1[3]	Uncondiational Jump, C in ALIT
	JBS A, B, C	PC ← C << 2 if A[B] is set	1[4]	Jump if Bit Set, A in OREG, B in BITLIT, C in ALIT
	JBC A, B, C	PC ← C << 2 if A[B] is clear	1[4]	Jump if Bit Cleared, A in OREG, B in BITLIT, C in ALIT
	CALL C	R7 ← R7 + 4 MEM(R7[RAW+USR+1:2])[RAW+USR+1:0] ← PC+ 4 PC ← C << 2	2[5]	Call Subroutine, C in ALIT
	RET	PC ← MEM(R7[RAW+USR+1:2])[RAW+USR+1:0] R7 ← R7 - 4	2[5]	Return from Subroutine
	JMPI	PC ← R6[RAW+USR+1:2] << 2	1[3]	Uncondiational Jump Indirect
	JBSI A, B	PC ← R6[RAW+USR+1:2] << 2 if A[B] is set	1[4]	Jump if Bit Set Indirect, A in XOREG, B in XBITLIT
	JBCI A, B	PC ← R6[RAW+USR+1:2] << 2 if A[B] is clear	1[4]	Jump if Bit Clear Indirect, A in OREG, B in XBITLIT
	CALLI	R7 ← R7 + 4 MEM(R7[RAW+USR+1:2])[RAW+USR+1:0] ← PC+ 4 PC ← R6[RAW+USR+1:2] << 2	2[5]	Call Subroutine Indirect
Others	WURM A, B, C	wait until A = (B AND ((0xFF << 16) + C))	>= 1[6]	Wait Until Register Match, A in OREG, B in OREG, C in MSKLIT
	WURMX A, B	wait until A = (B AND R6)	>= 1[6]	Wait Until Register Match, A in OREG, B in WXREG
	WURCX A, B	wait until A ≠ (B AND R6)	>= 1[6]	Wait Until Register Change, A in OREG, B in WXREG
	WUCE A, B	wait until cyclic event comparison matches	>= 1[6]	Wait Until Cyclic Event, A in OREG, B in OREG
	NOP		1	No Operation

Footnotes:

1) Not faster than 1+NPS clock cycles due to pipeline flushing.
2) Not faster than 1+2*NPS clock cycles due to pipeline flushing.
3) Not faster than NPS clock cycles due to pipeline flushing.
4) If the jump is executed, it is not faster than NPS clock cycles due to pipeline flushing.
5) Not faster than 2*NPS clock cycles due to pipeline flushing.
6) If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode the worst case latency for reactivating a prioritized MCS-channel is 2+NPS clock cycles.
7) Always faster than one ARU round trip cycle.
8) Suspends current MCS-channel if addressed slave inserts at least one wait cycle otherwise 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.
9) Not faster than C+NPS-1 clock cycles due to pipeline flushing.
10) Not faster than C+3+NPS clock cycles due to pipeline flushing.

Figure 112 Instruction Set Summary (part 3)

Menemonic	Instruction Code
MOVL	0001aaaaccccccccccccccccccccccccc
MOV	1010aaaabbbb0000-a-b----------
MRD	1010aaaa----0001-ccccccccccccc--
MWR	1010aaaa----0010-ccccccccccccc--
MRDI	1010aaaabbbb0011-ccccccccccccc--
MWRI	1010aaaabbbb0100-ccccccccccccc--
POP	1010aaaa----0101----------------
PUSH	1010aaaa----0110----------------
MWRL	1010aaaa----0111-ccccccccccccc--
MWRIL	1010aaaabbbb1000----------------
BRD	1010-aaa----1001ccccccccccccc--
BWR	1010-aaa----1010ccccccccccccc--
BRDI	1010-aaa-bbb1011----------------
BWRI	1010-aaa-bbb1100----------------
MRDIO	1010aaaabbbb1101-a-b----------
MWRIO	1010aaaabbbb1110-a-b----------
XCHB	1010aaaabbbb1111-a-b----------
ARD	1011aaaabbbb0000-------ccccccccc
AWR	1011aaaabbbb0001----------ccccc
NARD	1011aaaabbbb0010-------ccccccccc
NARDI	1011aaaabbbb0011----------------
ARDI	1011aaaabbbb0100----------------
AWRI	1011aaaabbbb0101----------------
SETB	1011aaaabbbb0110-a-b----------
CLRB	1011aaaabbbb0111-a-b----------
ADDL	0010aaaaccccccccccccccccccccccccc
ADD	1100aaaabbbb0000-a-b----------
SUBL	0011aaaaccccccccccccccccccccccccc
SUB	1100aaaabbbb0001-a-b----------
NEG	1100aaaabbbb0010-a-b----------
ANDL	0100aaaaccccccccccccccccccccccccc
AND	1100aaaabbbb0011-a-b----------
ORL	0101aaaaccccccccccccccccccccccccc
OR	1100aaaabbbb0100-a-b----------
XORL	0110aaaaccccccccccccccccccccccccc
XOR	1100aaaabbbb0101-a-b----------
SHR	1100aaaa----0110-a---------ccccc
SHL	1100aaaa----0111-a---------ccccc

Figure 113 Instruction Codes (part 1)

Menemonic	Instruction Code
MULU	1100aaaabbbb1000-a-b-------ccccc
MULS	1100aaaabbbb1001-a-b-------ccccc
DIVU	1100aaaabbbb1010-a-b-------ccccc
DIVS	1100aaaabbbb1011-a-b-------ccccc
MINU	1100aaaabbbb1100-a-b----------
MINS	1100aaaabbbb1101-a-b----------
MAXU	1100aaaabbbb1110-a-b----------
MAXS	1100aaaabbbb1111-a-b----------
ASL	1101aaaabbbb0011-a-b----------
ASRU	1101aaaabbbb0100-a-b----------
ASRS	1101aaaabbbb0101-a-b----------
ADDC	1101aaaabbbb0110-a-b----------
SUBC	1101aaaabbbb0111-a-b----------
ATUL	0111aaaaccccccccccccccccccccccc
ATU	1101aaaabbbb0000-a-b----------
ATSL	1000aaaaccccccccccccccccccccccc
ATS	1101aaaabbbb0001-a-b----------
BTL	1001aaaaccccccccccccccccccccccc
BT	1101aaaabbbb0010-a-b----------
JMP	1110--------0000-ccccccccccccc--
JBS	1110aaaabbbb0001-ccccccccccccc--
JBC	1110aaaabbbb0010-ccccccccccccc--
CALL	1110--------0011-ccccccccccccc--
RET	1110--------0100----------------
JMPI	1110--------0101----------------
JBSI	1110aaaabbbb0110-a-b----------
JBCI	1110aaaabbbb0111-a-b----------
CALLI	1110--------1000----------------
WURM	1111aaaabbbb0000ccccccccccccccc
WURMX	1111aaaabbbb0001---b----------
WURCX	1111aaaabbbb0010---b----------
WUCE	1111aaaabbbb0011----------------
NOP	0000----------------------------

Figure 114 Instruction Codes (part 2)

The individual instructions are decoded by evaluating the bits '0' and '1' at its expected positions, as mentioned in the table above. If the instruction decoder detects an invalid combination of these bits, the corresponding MCS channel is disabled and the ERR bit in the register STA is set. Bit positions marked as '-' are not relevant for the instruction. The bit position 'a', 'b', and 'c' are reserved for binary encoding of the instruction arguments A, B, and C.

Moreover, each instruction can set the **ERR** bit of register **STA** and stop the program execution, if a register write protection of an associated MCS channel is activated by the register **MCS[i]_REG_PROT**. This behavior is not explicitly mentioned in the instruction descriptions below. If an error occurs due to a write access to a protected register, it is ensured that the protected register is not overwritten. However, it is not ensured that other operations (e.g. PC updates) of the bad instruction are executed.

28.17.7.1 MOVL Instruction

Syntax	Operation	Status	Duration
MOVL A, C	A ← C	Z	1 instruction cycle

Transfer literal value C (C ∈ WLIT) to register A (A ∈ OREG).

The zero bit Z of status register STA is set, if the transferred value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.2 MOV Instruction

Syntax	Operation	Status	Duration
MOV A, B	A ← B	Z	1 instruction cycle

Transfer register B (B ∈ XOREG) to register A (A ∈ XOREG).

The zero bit Z of status register STA is set, if the transferred value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.3 MRD Instruction

Syntax	Operation	Status	Duration
MRD A, C	A ← MEM(C)[W-1:0]; MHB ← MEM(C)[RDW-1:W]	Z	2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Transfer the lower W bits of memory content at location C (C ∈ ALIT) to register A (A ∈ OREG).

The upper RDW-W bits of the memory content at location C are transferred to the MHB register.

The zero bit Z of status register STA is set, if the lower W bits of the transferred value are zero, otherwise the zero bit is cleared.

If the MHB register is selected as destination register A (A ∈ OREG), the bits 0 to RDW-W-1 of the referred memory location are transferred to MHB.

The program counter PC is incremented by the value 4.

28.17.7.4 MWR Instruction

Syntax	Operation	Status	Duration
MWR A, C	MEM(C)[W-1:0] ← A; MEM(C)[RDW-1:W] ← MHB	-	2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Transfer W bit value of register A (A ∈ OREG) together with the MHB register to the memory at location C (C ∈ ALIT).

The W bit value of register A is stored in the LSBs (bit 0 to W-1) of the memory location.

The MHB register is stored in bits W to RDW-W-1 of the referred memory location.

The program counter PC is incremented by the value 4.

28.17.7.5 MWRL Instruction

Syntax	Operation	Status	Duration
MWRL A, C	MEM(C)[W-1:0] ← A	-	3 instruction cycles but not faster than 1+2*NPS clock cycles due to pipeline flushing.

Transfer W bit value of register A (A ∈ OREG) to memory at location C (C ∈ ALIT).

The W bit value of register A is stored in the LSBs (bit 0 to W-1) of the memory location and the bits W to RDW-W are left unchanged.

The program counter PC is incremented by the value 4.

It should be noted that this operation is not an atomic instruction.

28.17.7.6 MRDI Instruction

Syntax	Operation	Status	Duration
MRDI A, B [, C]	A ← MEM(B[RAW+USR+1:2] + C)[W-1:0] MHB ← MEM(B[RAW+USR+1:2] + C)[RDW-1:W]	Z	2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Transfer the bits 0 to W-1 of a memory location to register A (A ∈ OREG) using indirect addressing.

The upper RDW-W bits of this memory location are transferred to MHB register.

The memory location where to read from depends on register B (B ∈ OREG) and literal C (C ∈ AOLIT) and it is defined as B[RAW+USR+1:2] + C.

If the optional operand C is not available in the assembler syntax, the MCS assembler generates code with a default value of 0 for operand C.

The zero bit Z of status register STA is set, if the transferred bits 0 to W-1 are zero, otherwise the zero bit is cleared.

If the MHB register is selected as destination register A (A ∈ OREG), the bits 0 to RDW-W-1 of the referred memory location are transferred to MHB.

The program counter PC is incremented by the value 4.

28.17.7.7 MRDIO Instruction

Syntax	Operation	Status	Duration
MRDIO A, B	A ← MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[W-1:0] MHB ← MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[RDW-1:W]	Z	2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Transfer the bits 0 to W-1 of a memory location to register A (A ∈ XOREG) using indirect addressing with offset calculation.

The upper RDW-W bits of this memory location are transferred to MHB register.

The memory location where to read from depends on register B (B ∈ BAREG) and register R5 and it is defined as B[RAW+USR+1:2] + R5[RAW+USR+1:2].

The zero bit Z of status register STA is set, if the transferred bits 0 to W-1 are zero, otherwise the zero bit is cleared.

If the MHB register is selected as destination register A, the bits 0 to RDW-W-1 of the referred memory location are transferred to MHB.

The program counter PC is incremented by the value 4.

28.17.7.8 MWRI Instruction

Syntax	Operation	Status	Duration
MWRI A, B [, C]	MEM(B[RAW+USR+1:2] + C)[W-1:0] ← A; MEM(B[RAW+USR+1:2] + C)[RDW-1:W] ← MHB	-	2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Transfer value of register A (A ∈ OREG) to the LSBs 0 to W-1 of a memory location using indirect addressing.

The MHB register is moved to the bits W to RDW-1 at the same memory location.

If the optional operand C is not available in the assembler syntax, the MCS assembler generates code with a default value of 0 for operand C.

The memory location where to write to depends on register B (B ∈ OREG) and literal C (C ∈ AOLIT) and it is defined as B[RAW+USR+1:2] + C.

The program counter PC is incremented by the value 4.

28.17.7.9 MWRIO Instruction

Syntax	Operation	Status	Duration
MWRIO A, B	MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[W-1:0] ← A; MEM(B[RAW+USR+1:2] + R5[RAW+USR+1:2])[RDW-1:W] ← MHB	-	2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Transfer value of register A (A ∈ XOREG) to the LSBs 0 to W-1 of a memory location using indirect addressing with offset calculation.

The MHB register is moved to the bits W to RDW-1 at the same memory location.

The memory location where to write to depends on register B (B ∈ BAREG) and register R5 and it is defined as B[RAW+USR+1:2] + R5[RAW+USR+1:2].

The program counter PC is incremented by the value 4.

28.17.7.10MWRIL Instruction

Syntax	Operation	Status	Duration
MWRIL A, B	MEM(B[RAW+USR+1:0])[W-1:0] ← A;	-	3 instruction cycles but not faster than 1+2*NPS clock cycles due to pipeline flushing.

Transfer W bit value of A (A ∈ OREG) to memory using indirect addressing.

The memory location where to write to is defined by the bits 2 to RAW+1 of register B (B$\in$ OREG).

The W bit value is stored in the LSBs (bit 0 to W-1) of the memory location and the bits W to RDW-1 are left unchanged.

The program counter PC is incremented by the value 4.

It should be noted that this operation is not an atomic instruction.

28.17.7.11 POP Instruction

Syntax	Operation	Status	Duration
POP A	A ← MEM(R7[RAW+USR+1:2])[W-1:0]; MHB ← MEM(R7[RAW+USR+1:2])[RDW-1:W]; R7 ← R7 - 4; SP_CNT ← SP_CNT - 1	Z, EN	2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Transfer the LSBs (bit 0 to W-1) from the top of stack to register A (A$\in$ OREG), followed by decrementing the stack pointer register R7 with the value 4.

The upper bits W to RDW-1 from the top of the stack are transferred to register MHB.

If the MHB register is selected as destination register A (A$\in$ OREG), the bits 0 to RDW-W-1 from the top of the stack are transferred to MHB.

The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register R7.

The zero bit Z of status register STA is set, if the lower W bit of the transferred value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

The **SP_CNT** bit field inside the **MCS[i]_CH[x]_CTRL** register is decremented.

If an underflow on the **SP_CNT** bit field occurs, the STK_ERR[i]_IRQ is raised.

If an underflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL** is set, the current MCS channel is disabled by clearing the **EN** bit of **STA**.

28.17.7.12 PUSH Instruction

Syntax	Operation	Status	Duration
PUSH A	R7 ← R7 + 4; MEM(R7[RAW+USR+1:2])[W-1:0] ← A; MEM(R7[RAW+USR+1:2])[RDW-1:W] ← MHB SP_CNT ← SP_CNT + 1;	EN	2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Increment the stack pointer register R7 with the value 4, followed by transferring a W bit value of operand A (A$\in$ OREG) together with a MHB register to the new top of the stack. The W bit value of A is stored in the bits 0 to W-1 of the memory location.

The content of the MHB register is stored in the bit W to RDW-1 of the memory location.

The memory location for the top of the stack is referred by the bits 2 to RAW+1 of the stack pointer register.

The program counter PC is incremented by the value 4.

The SP_CNT bit field inside the MCS[i]_CH[x]_CTRL register is incremented.

If an overflow on the SP_CNT bit field occurs, the STK_ERR[i]_IRQ is raised.

If an overflow on the SP_CNT bit field occurs and the bit HLT_SP_OFL of register MCS[i]_CTRL is set, the current MCS channel is disabled by clearing the EN bit of STA.

If an overflow on the SP_CNT bit field occurs and the bit HLT_SP_OFL of register MCS[i]_CTRL is set, the memory write operation for the A and MHB is discarded.

28.17.7.13ARD Instruction

Syntax	Operation	Status	Duration
ARD A, B, C	A ← ARU(C)[W-1:0]; B ← ARU(C)[2*W-1:W]; ACB ← ARU(C)[4+2*W:2*W]	CAT, SAT	Suspends current MCS channel until ARU transfer finished.

Perform a blocking read access to the ARU and transfer both W bit values received at the ARU port to the registers A and B (A $\in$ AREG, B $\in$ AREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.

If A and B refer to the same register, only the upper W bit ARU word is stored and the lower W bit ARU word is discarded.

If any transferred W bit value from the ARU should not be stored in a register, the dummy register ZERO $\in$ AREG can be selected in A or B to discard the corresponding ARU data. The binary encoding of the address for the dummy register ZERO can be chosen by an arbitrary value within the range 8 to 15.

The received ARU control bits are stored in the register ACB.

The literal C (C $\in$ ARDLIT) define the ARU address where to read from.

At the beginning of the instruction execution the CAT bit in register STA is always cleared. After the execution of the instruction the SAT flag of the register STA is updated in order to show if the transfer was successful (SAT = 1) or if the transfer failed (SAT = 0) due to a cancellation by the CPU.

The program counter PC is incremented by the value 4.

28.17.7.14ARDI Instruction

Syntax	Operation	Status	Duration
ARDI A, B	A ← ARU(R6[8:0])[W-1:0]; B ← ARU(R6[8:0])[2*W-1:W]; ACB ← ARU(R6[8:0])[4+2*W:2*W]	CAT, SAT	Suspends current MCS channel until ARU transfer finished.

Perform a blocking read access to the ARU and transfer both W bit values received at the ARU port to the registers A and B (A $\in$ AREG, B $\in$ AREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.

If A and B refer to the same register, only the upper W bit ARU word is stored and the lower W bit ARU word is discarded.

If any transferred W bit value from the ARU should not be stored in a register, the dummy register ZERO $\in$ AREG can be selected in A or B to discard the corresponding ARU data. The binary encoding of the address for the dummy register ZERO can be chosen by an arbitrary value within the range 8 to 15.

The received ARU control bits are stored in the register ACB.

The read address is obtained from the bits 0 to 8 of the channels register R6.

At the beginning of the instruction execution the CAT bit in register STA is always cleared. After the execution of the instruction the SAT flag of the register STA is updated in order to show if the transfer was successful (SAT = 1) or if the transfer failed (SAT = 0) due to a cancellation by the CPU.

The program counter PC is incremented by the value 4.

28.17.7.15AWR Instruction

Syntax	Operation	Status	Duration
AWR A, B, C	ARU(C)[W-1:0] ← A; ARU(C)[2*W-1:W] ← B; ARU(C)[4+2*W:2*W] ← ACB;	CAT, SAT	Suspends current MCS channel until ARU transfer finished.

Perform a blocking write access to the ARU and transfer two W bit values to the ARU port using the registers A and B (A$\in$ OREG, B$\in$ OREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.

The ARU control bits are taken from the register ACB.

The literal C (C$\in$ AWRLIT) defines an index into the pool of ARU write addresses that are used for writing data. This index is mapped to an ARU write address as shown in column "MCS write index" of table "ARU Write Addresses" in device specific appendix.

Each MCS sub module has a pool of several write addresses that can be shared between all MCS channels arbitrarily.

At the beginning of the instruction execution the CAT bit of the register STA is always cleared. After the execution of the instruction the SAT flag of the register STA is updated in order to show if the transfer was successful (SAT = 1) or if the transfer failed (SAT = 0) due to a cancellation by the CPU.

The program counter PC is incremented by the value 4.

28.17.7.16AWRI Instruction

Syntax	Operation	Status	Duration
AWRI A, B	ARU(R6[4:0])[W-1:0] ← A; ARU(R6[4:0])[2*W-1:W] ← B; ARU(R6[4:0])[4+2*W:2*W] ← ACB;	CAT, SAT	Suspends current MCS channel until ARU transfer finished.

Perform a blocking write access to the ARU and transfer two W bit values to the ARU port using the registers A and B (A$\in$ OREG, B$\in$ OREG), whereas A holds the lower W bit ARU word and B holds the upper W bit ARU word.

The ARU control bits are taken from the register ACB.

The bits 0 to 4 of the register R6 define an index into the pool of ARU write addresses that are used for writing data. This index is mapped to an ARU write address as shown in column "MCS write index" of table "ARU Write Addresses" in device specific appendix.

Each MCS sub module has a pool of several write addresses that can be shared between all MCS channels arbitrarily.

At the beginning of the instruction execution the CAT bit of the register STA is always cleared. After the execution of the instruction the SAT flag of the register STA is updated in order to show if the transfer was successful (SAT = 1) or if the transfer failed (SAT = 0) due to a cancellation by the CPU.

The program counter PC is incremented by the value 4.

28.17.7.17NARD Instruction

Syntax	Operation	Status	Duration
NARD A, B, C	A ← ARU(C)[W-1:0]; B ← ARU(C)[2*W:W]; ACB ← ARU(C)[4+2*W:2*W]	SAT	Suspends current MCS channel until the ARU is selecting the MCS channel.

Perform a non-blocking read access to the ARU trying to transfer both W bit values received at the ARU port to the registers A and B (A$\in$ AREG, B$\in$ AREG), whereas A holds the lower W bit ARU word, B holds the upper W bit ARU word, and the ACB register holds the received ARU control bits.

The literal C (C$\in$ ARDLIT) defines the ARU address where to read from.

Non-blocking ARU read means that the instruction is suspending the MCS channel until the ARU scheduler is selecting the requesting MCS channel.

If the transfer finished successfully, the bit SAT of the register STA is set and the transferred values are stored in the registers A, B, and ACB.

If the transfer failed due to missing data at the requested source, the bit SAT of the register STA is cleared and registers A, B, and ACB are not changed.

If A and B refer to the same register, only the upper W bit ARU word is stored and the lower W bit ARU word is discard.

If any transferred W bit value from the ARU should not stored in a register, the dummy register ZERO$\in$ AREG can be selected in A or B to discard the corresponding ARU data. The binary encoding of the address for the dummy register ZERO can be chosen by an arbitrary value within the range 8 to 15.

The program counter PC is incremented by the value 4.

28.17.7.18NARDI Instruction

Syntax	Operation	Status	Duration
NARDI A, B	A ← ARU(R6[8:0])[W-1:0]; B ← ARU(R6[8:0])[2*W-1:W]; ACB ← ARU(R6[8:0])[4+2*W:2*W]	SAT	Suspends current MCS channel until the ARU is selecting the MCS channel.

Perform a non-blocking read access to the ARU trying to transfer both W bit values received at the ARU port to the registers A and B (A$\in$ AREG, B$\in$ AREG), whereas A holds the lower W bit ARU word, B holds the upper W bit ARU word, and the ACB register holds the received ARU control bits. The read address is obtained from bits 0 to 8 of the channel register R6.

Non-blocking ARU read access means that the instruction suspends the MCS channel until the ARU scheduler selects the requesting MCS channel. If the transfer finished successfully, the bit SAT of the register STA is set and the transferred values are stored in the registers A, B, and ACB.

If the transfer failed due to missing data at the requested source, the bit SAT of the register STA is cleared and registers A, B, and ACB are not changed.

If A and B refer to the same register, only the upper W bit ARU word is stored and the lower 24 bit ARU word is discard.

If any transferred W bit value from the ARU should not stored in a register, the dummy register ZERO$\in$ AREG can be selected in A or B to discard the corresponding ARU data. The binary encoding of the address for the dummy register ZERO can be chosen by an arbitrary value within the range 8 to 15.

The program counter PC is incremented by the value 4.

28.17.7.19BRD Instruction

Syntax	Operation	Status	Duration
BRD A, C	A ← BUS(C)[W-1:0]; MHB ← BUS(C)[BDW-1:W]	-	Suspends current MCS channel if addressed slave inserts at least one wait cycle (e.g. accessing RAM module) otherwise 2 instruction cycles but not faster than 1+NPS clock cycles due to pipeline flushing.

Initiate a read access at the bus master interface using the address C (C $\in$ BALIT) and transfer the lower W bits of the received data to register A (A $\in$ GREG).

The upper BDW-W bits of the received data are transferred to the MHB register.

If the delay between a BRD instruction and its successor instruction is one or two system clock cycles (e.g. in accelerated scheduling mode) and the successor instruction is reading data resulting from the BRD instruction, a data hazard in the pipeline occurs resulting in a pipeline flush. this means, if very fast program execution is required (e.g. only one task is activated in accelerated scheduling mode) a program sequence like BRD R1,0x288; ADD R3,R1 (9 clock cycles) can be accelerated by reformulating the sequence as BRD R1,0x0288; NOP;NOP;NOP;ADD R3,R1; (4 clock cycles).

Since the MHB register is always transferred via AEI bus master it also figures out another data dependency, which can cause a data hazard resulting in a pipline flush. Therefore a sequence like BRD R1, 0x0288; BWR R3, 0x304 (9 clock cycles) could also be optimized by the sequence BRD R1, 0x0288; NOP; NOP; BWR R3, 0x304(4 clock cycles).

The program counter PC is incremented by the value 4.

28.17.7.20BWR Instruction

Syntax	Operation	Status	Duration
BWR A, C	BUS(C)[W-1:0] ← A; BUS(C)[BDW-1:W] ← MHB	-	Suspends current MCS channel if addressed slave inserts at least one wait cycle (e.g. accessing a RAM module) otherwise 1 cycle.

Initiate a write access at the bus master interface using the address C (C $\in$ BALIT) and transfer the content of register A (A $\in$ GREG) to the bits 0 to W-1 of the bus.

The content of the MHB register is transferred to the bits W to BDW-W-1 of the bus.

The program counter PC is incremented by the value 4.

28.17.7.21BRDI Instruction

Syntax	Operation	Status	Duration
BRDI A, B	A ← BUS(B[BAW+1:2])[W-1:0]; MHB ← BUS(B[BAW+1:2])[BDW-1:W]	-	Suspends current MCS channel if addressed slave inserts at least one wait cycle (e.g. accessing a RAM module) otherwise 1 clock cycles.

Initiate a read access at the bus master interface using indirect addressing and transfer the lower W bits of the received data to register A (A $\in$ GREG).

The upper BDW-W bits of the received data are transferred to the MHB register.

The address for the transfer is identified by the bits 2 to BAW+1 of register B (B$\in$ GREG).

If the delay between a BRDI instruction and its successor instruction is one or two system clock cycles (e.g. in accelerated scheduling mode) and the successor instruction is reading data resulting from the BRDI instruction, a data hazard in the pipeline occurs resulting in a pipeline flush. this means, if very fast program execution is required (e.g. only one task is activated in accelerated scheduling mode) a program sequence like BRD I R1,R6; ADD R3, R1 (9 clock cycles) can be accelerated by reformulating the sequence as BRDI R1, R6; NOP; NOP; NOP; ADD R3, R1; (4 clock cycles).

Since the MHB register is always transferred via AEI bus master it also figures out another data dependency, which can cause a data hazard resulting in a pipline flush. Therefore a sequence like BRDI R1, R2; BWRI R3, R4 (9 clock cycles) could also be optimized by the sequence BRDI R1, R2; NOP; NOP; BWRI R3, R4 (4 clock cycles).

The program counter PC is incremented by the value 4.

28.17.7.22BWRI Instruction

Syntax	Operation	Status	Duration
BWRI A, B	BUS(B[BAW+1:2])[W-1:0] ← A; BUS(B[BAW+1:2])[BDW-1:W] ← MHB	-	Suspends current MCS channel if addressed slave inserts at least one wait cycle (e.g. accessing a RAM module) otherwise 1 instruction cycle.

Initiate a write access at the bus master interface using the indirect addressing and transfer the content of register A (A$\in$ GREG) to the bits 0 to W-1 of the bus.

The content of the MHB register is transferred to the bits W to BDW-W-1 of the bus.

The address for the transfer is identified by the bits 2 to BAW+1 of register B (B$\in$ GREG).

The program counter PC is incremented by the value 4.

28.17.7.23ADDL Instruction

Syntax	Operation	Status	Duration
ADDL A, C	A ← A + C	Z, CY, N, V	1 instruction cycle

Perform addition operation of a register A (A$\in$ OREG) with a W bit literal value C (C$\in$ WLIT) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned overflow/underflow occurred during addition, otherwise the bit is cleared. An unsigned overflow has occurred when the result of the operation cannot be represented in the interval [0; 2^W-1], assuming that both operands A and C are unsigned values within the interval [0; 2^W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during addition, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and C are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

28.17.7.24ADD Instruction

Syntax	Operation	Status	Duration
ADD A, B	A ← A + B	Z, CY, N, V	1 instruction cycle

Perform addition operation of a register A (A ∈ XOREG) with an operand B (B ∈ XOREG). The result is stored in the register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned overflow occurred during addition, otherwise the bit is cleared. An unsigned overflow has occurred when the result of the operation cannot be represented in the interval [0; 2^W-1], assuming that both operands A and B are unsigned values within the interval [0; 2^W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during addition, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and B are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

28.17.7.25ADDC Instruction

Syntax	Operation	Status	Duration
ADDC A, B	A ← A + B + CY	Z, CY, N, V	1 instruction cycle

Perform addition operation of a register A (A ∈ XOREG) with an operand B (B ∈ XOREG) and the carry flag CY. The result is stored in the register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned overflow occurred during addition, otherwise the bit is cleared. An unsigned overflow has occurred when the result of the operation cannot be represented in the interval [0; 2^W-1], assuming that both operands A and B are unsigned values within the interval [0; 2^W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during addition, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and B are signed values within the interval [$-2^{W.1}$; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

28.17.7.26SUBL Instruction

Syntax	Operation	Status	Duration
SUBL A, C	A ← A - C	Z, CY, N, V	1 instruction cycle

Perform subtraction operation of a register A (A ∈ OREG) with a W bit literal value C (C ∈ WLIT). The result is stored in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned underflow occurred during subtraction, otherwise the bit is cleared. An unsigned underflow has occurred when the result of the operation cannot be represented in the interval [0; 2^W-1], assuming that both operands A and C are unsigned values within the interval [0; 2^W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and C are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

28.17.7.27SUB Instruction

Syntax	Operation	Status	Duration
SUB A, B	A ← A - B	Z, CY, N, V	1 instruction cycle

Perform subtraction operation of a register A (A ∈ XOREG) with an operand B (B ∈ XOREG). The result is stored in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned underflow occurred during subtraction, otherwise the bit is cleared. An unsigned underflow has occurred when the result of the operation cannot be represented in the interval [0; 2^W-1], assuming that both operands A and B are unsigned values within the interval [0; 2^W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^W-1; 2^W-1-1], assuming that both operands A and B are signed values within the interval [-2^W-1; 2^W-1-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

28.17.7.28SUBC Instruction

Syntax	Operation	Status	Duration
SUBC A, B	A ← A - B - CY	Z, CY, N, V	1 instruction cycle

Perform subtraction operation of a register A (A ∈ XOREG) with an operand B (B ∈ XOREG) and the carry flag CY. The result is stored in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if an unsigned underflow occurred during subtraction, otherwise the bit is cleared. An unsigned underflow has occurred when the result of the operation cannot be represented in the interval [0; 2W-1], assuming that both operands A and B are unsigned values within the interval [0; 2W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that both operands A and B are signed values within the interval [-2^{W-1}; 2^{W-1}-1].

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

28.17.7.29NEG Instruction

Syntax	Operation	Status	Duration
NEG A, B	A ← - B	Z, N, V	1 instruction cycle

Perform negation operation (2's Complement) with an operand B (B $\in$ XOREG) and store the result in a register A (A $\in$ XOREG).

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during subtraction, otherwise the bit is cleared. A signed overflow/underflow has occurred when the result of the operation cannot be represented in the interval $[-2^{W-1}; 2^{W-1}-1]$, assuming that both operands A and B are signed values within the interval $[-2^{W-1}; 2^{W-1}-1]$.

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0), assuming that no overflow/underflow occurred.

The program counter PC is incremented by the value 4.

28.17.7.30ANDL Instruction

Syntax	Operation	Status	Duration
ANDL A, C	A ← A AND C	Z	1 instruction cycle

Perform bitwise AND conjunction of a register A (A $\in$ OREG) with a W bit literal value C (C $\in$ WLIT) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.31AND Instruction

Syntax	Operation	Status	Duration
AND A, B	A ← A AND B	Z	1 instruction cycle

Perform bitwise AND conjunction of a register A (A $\in$ XOREG) with an operand B (B $\in$ XOREG) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.32ORL Instruction

Syntax	Operation	Status	Duration
ORL A, C	A ← A OR C	Z	1 instruction cycle

Perform bitwise OR conjunction of a register A (A$\in$ OREG) with a W bit literal value C (C$\in$ WLIT) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.33OR Instruction

Syntax	Operation	Status	Duration
OR A, B	A ← A OR B	Z	1 instruction cycle

Perform bitwise OR conjunction of a register A (A$\in$ XOREG) with an operand B (B$\in$ XOREG) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.34XORL Instruction

Syntax	Operation	Status	Duration
XORL A, C	A ← A XOR C	Z	1 instruction cycle

Perform bitwise XOR conjunction of a register A (A$\in$ OREG) with a W bit literal value C (C$\in$ WLIT) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.35XOR Instruction

Syntax	Operation	Status	Duration
XOR A, B	A ← A XOR B	Z	1 instruction cycle

Perform bitwise XOR conjunction of a register A (A$\in$ XOREG) with an operand B (B$\in$ XOREG) and store the result in register A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.36SHR Instruction

Syntax	Operation	Status	Duration
SHR A, C	A ← A >> C	Z, CY	1 instruction cycle

Perform right shift operation C (C$\in$ SFTLIT) times of register A (A$\in$ XOREG). The MSBs that are shifted into A are cleared.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is updated to the last LSB that is shifted out of the register. If the shift value C is 0 the carry bit CY is cleared.

The program counter PC is incremented by the value 4.

28.17.7.37SHL Instruction

Syntax	Operation	Status	Duration
SHL A, C	A ← A << C	Z, CY	1 instruction cycle

Perform left shift operation C ($C_{\in}$ SFTLIT) times of register A ($A_{\in}$ XOREG). The LSBs that are shifted into A are cleared.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is updated to the previous MSB that is shifted out of the register. If the register A contains less than W bits or if C is 0, the carry bit CY is always cleared.

The program counter PC is incremented by the value 4.

28.17.7.38ASRU Instruction

Syntax	Operation	Status	Duration
ASRU A, B	A ← A >> B	Z	1 instruction cycle

Perform arithmetic unsigned right shift operation, which means that the unsigned operand of register A ($A_{\in}$ XOREG) is right shifted B times ($B_{\in}$ XOREG). Operand B is also an unsigned type. The MSBs that are shifted into A are cleared.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.39 ASRS Instruction

Syntax	Operation	Status	Duration
ASRS A, B	A ← A >> B	Z	1 instruction cycle

Perform arithmetic signed right shift operation, which means that the signed operand of register A ($A_{\in}$ XOREG) is right shifted B times ($B_{\in}$ XOREG). Operand B is an unsigned type. The operation also performs a sign extension, which means that value of the MSBs that are shifted into A are determined by the MSB of the original operand A.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.40 ASL Instruction

Syntax	Operation	Status	Duration
ASL A, B	A ← A << B	Z, CY, V	1 instruction cycle

Perform arithmetic left shift operation for signed and unsigned numbers, which means that the operand of register A ($A_{\in}$ XOREG) is left shifted B times ($B_{\in}$ XOREG). Operand B is always an unsigned type.

The carry bit CY of status register STA is set, if an unsigned overflow occurred during shifting, otherwise the bit is cleared. An unsigned overflow has occurred if the calculated result A*2B cannot be represented in the interval [0; 2^W-1], assuming that both operands A and B are unsigned values within the interval [0; 2^W-1].

The overflow bit V of status register STA is set, if a signed overflow/underflow occurred during shifting, otherwise the bit is cleared. A signed overflow/underflow has occurred when the calculated result A*2B cannot be represented in the interval [-2^{W-1}; 2^{W-1}-1], assuming that signed operand A is within the interval [-2^{W-1}; 2^{W-1}-1] and the unsigned operand B is within the interval [0; 2^{W-1}-1].

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.41 MULU Instruction

Syntax	Operation	Status	Duration
MULU A, B[, C]	[[R4,] A] ← A[(C-1):0] * B[(C-1):0]	Z	1 instruction cycle

Perform an unsigned multiplication operation of an operand A (A$_\in$ XOREG) with an operand B (B$_\in$ XOREG). The multiplication is only performed with the bits 0 to C-1 (C$_\in$ BWSLIT) of both operands A and B and the bits C to W-1 are ignored.

If C is less than or equal to W/2, the product of the multiplication is stored in register A and register R4 is left unchanged.

If C is greater than W/2, the bits 0 to W-1 are stored in A and the bits W to 2*C-1 are stored in R4. The results stored in the registers are always zero extended to W bits.

If the optional operand C is not specified in the assembler code, the MCS assembler generates code with a default value of W for operand C.

The zero bit Z of status register STA is set, if the calculated product is zero, otherwise the zero bit is cleared.

If the delay between a MULU instruction and its successor instruction is one clock cycle (e.g. in accelerated scheduling mode) and the successor instruction is either a WURM, WURMX, WURCX, WUCE, BRDI, BWR or BWRI instruction that is accessing the multiplication result as argument, a data hazard in the pipeline occurs resulting in a pipeline flush.

This means, if very fast program execution is required (e.g. only one task is activated in accelerated scheduling mode) a program sequence like MULU R1, R2; BWRI R1, R3 (9 clock cycles) can be accelerated by reformulating the sequence as MULU R1, R2; NOP; BWRI R1, R3; (3 clock cycles).

The program counter PC is incremented by the value 4.

28.17.7.42 MULS Instruction

Syntax	Operation	Status	Duration
MULS A, B[, C]	[[R4,] A] ← A[(C-1):0] * B[(C-1):0]	Z, N	1 instruction cycle

Perform a signed multiplication operation of an operand A (A$_\in$ XOREG) with an operand B (B$_\in$ XOREG). The multiplication is only performed with the bits 0 to C-1 (C$_\in$ BWSLIT) of both operands A and B, in which bit C-1 is used as sign bit (-2^(C-1)) and the bits C to W-1 are ignored. If C is less than or equal to W/2, the product of the multiplication is stored in register A and register R4 is left unchanged. If C is greater than W/2, the bits 0 to W-1 are stored in A and the bits W to 2*C-1 are stored in R4. The results stored in the registers are always sign extended to W bits.

If the optional operand C is not specified in the assembler code, the MCS assembler generates code with a default value of W for operand C.

The zero bit Z of status register STA is set, if the calculated product is zero, otherwise the zero bit is cleared.

The negative bit N of status register STA equals the MSB of the operation result, in order to determine if a calculated signed result is negative (N=1) or positive (N=0).

If the delay between a MULS instruction and its successor instruction is one clock cycle (e.g. in accelerated scheduling mode) and the successor instruction is either a WURM,WURMX,WURCX,WUCE,BRDI,BWR or BWRI instruction that is accessing the multiplication result as argument, a data hazard in the pipeline occurs resulting in a pipeline flush.

This means, if very fast program execution is required (e.g. only one task is activated in accelerated scheduling mode) a program sequence like MULS R1, R2; BWRI R1, R3 (9 clock cycles) can be accelerated by reformulating the sequence as MULS R1, R2; NOP; BWRI R1,R 3; (3 clock cycles).

The program counter PC is incremented by the value 4.

28.17.7.43 DIVU Instruction

Syntax	Operation	Status	Duration
DIVU A, B[, C]	R4 ← A[(C-1):0] - B[(C-1):0] * | A[(C-1):0] / B[(C-1):0] |; A ← | A[(C-1):0] / B[(C-1):0] |	CY, Z, ERR	C instruction cycles but not faster than C+NPS-1 clock cycles due to pipeline flushing.

Perform an unsigned division operation of operand A (A $\in$ XOREG \\ {R4, B}) divided by operand B (B $\in$ XOREG \\ {R4, A}). The division is only performed with the bits 0 to C-1 (C $\in$ BWSLIT) of the operands and the remaining bits C to W-1 are ignored. This means that the dynamic range of A and B is defined in the interval [0; 2C-1]. The integral part of the quotient is stored in the register A and the remainder of the division is stored in register R4. The resulting quotient A and remainder R4 are always zero extended to W bits.

If the optional operand C is not specified in the assembler code, the MCS assembler generates code with a default value of W for operand C.

If the bits 0 to C-1 of operand B are zero, the MCS channel is disabled and the ERR bit in the status register STA is set.

The zero bit Z of status register STA is set, if the calculated quotient in A is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if the calculated remainder in R4 is not zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.44 DIVS Instruction

Syntax	Operation	Status	Duration
DIVS A, B[, C]	R4 ← A[(C-1):0] - B[(C-1):0] * | A[(C-1):0] / B[(C-1):0] |; A ← | A[(C-1):0] / B[(C-1):0] |	CY, Z, N, V, ERR	C + 4 instruction cycles but not faster than C+3+NPS clock cycles due to pipeline flushing.

Perform a signed division operation of operand A (A $\in$ XOREG \\ {R4, B}) divided by operand B (B $\in$ XOREG \\ {R4, A}). The division is only performed with the bits 0 to C-1 (C $\in$ BWSLIT) of both operands, in which bit C-1 is used as sign bit (-2^{C-1}) and the bits C to W-1 are ignored. This means that the dynamic range of A[(C-1):0] and B[(C-1):0] is defined in the interval [-2^{C-1}; $2^{C-1}-1$]. The integral part of the quotient is stored in the register A and the remainder

of the division is stored in register R4. The resulting quotient A and remainder R4 are always sign extended to W bits. The integral part of the quotient is always truncated towards 0. The sign of the remainder is always the same sign as the dividend A[(C-1):0]. The absolute value of the remained is always less than the divisor B[(C-1):0].

If the optional operand C is not specified in the assembler code, the MCS assembler generates code with a default value of W for operand C.

If the bits 0 to C-1 of operand B are zero, the MCS channel is disabled and the ERR bit in the status register STA is set.

The zero bit Z of status register STA is set, if the calculated quotient in A is zero, otherwise the zero bit is cleared.

The carry bit CY of status register STA is set, if the calculated remainder in R4 is not zero, otherwise the zero bit is cleared.

The overflow bit V of status register STA is set, if the calculated quotient in A cannot be represented in the interval $[-2^{W}-1; 2^{W-1}-1]$, otherwise the overflow bit is cleared.

The negative bit N of status register STA equals the MSB of the quotient, in order to determine if a calculated signed result is negative (N=1) or positive (N=0).

The program counter PC is incremented by the value 4.

28.17.7.45 MINU Instruction

Syntax	Operation	Status	Duration
MINU A, B	A ← MIN(A, B)	Z	1 instruction cycle

Determine the minimum of an unsigned operand A (A $\in$ XOREG) and an unsigned operand B (B $\in$ XOREG). If A is less than or equal to B, A is left unchanged. Otherwise, if A is greater than B, the operand B is moved to A.

The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

28.17.7.46 MINS Instruction

Syntax	Operation	Status	Duration
MINS A, B	A ← MIN(A, B)	Z	1 instruction cycle

Determine the minimum of a signed operand A (A $\in$ XOREG) and a signed operand B (B $\in$ XOREG). If A is less than or equal to B, A is left unchanged. Otherwise, if A is greater than B, the operand B is moved to A.

The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

28.17.7.47 MAXU Instruction

Syntax	Operation	Status	Duration
MAXU A, B	A ← MAX(A, B)	Z	1 instruction cycle

Determine the maximum of an unsigned operand A (A $\in$ XOREG) and an unsigned operand B (B $\in$ XOREG). If A is greater than or equal to B, A is left unchanged. Otherwise, if A is less than B, the operand B is moved to A.

The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

28.17.7.48 MAXS Instruction

Syntax	Operation	Status	Duration
MAXS A, B	A ← MAX(A, B)	Z	1 instruction cycle

Determine the maximum of a signed operand A (A ∈ XOREG) and a signed operand B (B ∈ XOREG). If A is greater than or equal to B, A is left unchanged. Otherwise, if A is less than B, the operand B is moved to A.

The zero bit Z of status register STA is set, if the calculated result of A is zero, otherwise the zero bit is cleared.

28.17.7.49ATUL Instruction

Syntax	Operation	Status	Duration
ATUL A, C	A - C	Z, CY	1 instruction cycle

Arithmetic test with an unsigned operand A (A ∈ OREG) and an unsigned W bit literal value C (C ∈ WLIT).

The carry bit CY of status register STA is set if unsigned operand A is less than unsigned literal C.

Otherwise, the carry bit CY of status register STA is cleared if unsigned operand A is greater than or equal to unsigned literal C.

The zero bit Z of status register STA is set, if A equals to C.

Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to C.

The program counter PC is incremented by the value 4.

28.17.7.50ATU Instruction

Syntax	Operation	Status	Duration
ATU A, B	A - B	Z, CY	1 instruction cycle

Arithmetic Test with an unsigned operand A (A ∈ XOREG) and an unsigned operand B (B ∈ XOREG).

The carry bit CY of status register STA is set if unsigned operand A is less than unsigned operand B.

Otherwise, the carry bit CY of status register STA is cleared if unsigned operand A is greater than or equal to unsigned operand B.

The zero bit Z of status register STA is set, if A equals to B.

Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to B.

The program counter PC is incremented by the value 4.

28.17.7.51ATSL Instruction

Syntax	Operation	Status	Duration
ATSL A, C	A - C	Z, CY	1 instruction cycle

Arithmetic Test with a signed operand A (A ∈ OREG) and a signed W bit literal value C (C ∈ WLIT).

The carry bit CY of status register STA is set if signed operand A is less than signed literal C.

Otherwise, the carry bit CY of status register STA is cleared if signed operand A is greater than or equal to signed literal C.

The zero bit Z of status register STA is set, if A equals to C.

Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to C.

The program counter PC is incremented by the value 4.

28.17.7.52ATS Instruction

Arithmetic Test with a signed operand A (A$\in$ XOREG) and a signed operand B (B$\in$ XOREG).

Syntax	Operation	Status	Duration
ATS A, B	A - B	Z, CY	1 instruction cycle

The carry bit CY of status register STA is set if unsigned operand A is less than signed operand B.

Otherwise, the carry bit CY of status register STA is cleared if signed operand A is greater than or equal to signed operand B.

The zero bit Z of status register STA is set, if A equals to B.

Otherwise, the zero bit Z of status register STA is cleared, if A is unequal to B.

The program counter PC is incremented by the value 4.

28.17.7.53BTL Instruction

Syntax	Operation	Status	Duration
BTL A, C	A AND C	Z	1 instruction cycle

Bit test of an operand A (A$\in$ OREG) with a W bit literal bit mask C (C$\in$ WLIT).

The bit test is performed by applying a bitwise logical AND operation with operand A and the bit mask C without storing the result.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.54BT Instruction

Syntax	Operation	Status	Duration
BT A, B	A AND B	Z	1 instruction cycle

Bit test of an operand A (A$\in$ XOREG) with an operand B (B$\in$ XOREG), whereas usually one of the operands is a register holding a bit mask.

The bit test is performed by applying a bitwise logical AND operation with register A and register B without storing the result.

The zero bit Z of status register STA is set, if the calculated value is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.55SETB Instruction

Syntax	Operation	Status	Duration
SETB A, B	A[B[4:0]] ←1	Z	1 instruction cycle

Set the B[4:0]-th bit (B ∈ XOREG) of an operand A (A ∈ XOREG) to true. Only the bits 0 to 4 of operand B are used as bit index of operand A and the other bits of B are ignored. If the value B[4:0] is greater than or equal to W, the operation of SETB does not modify operand A but the status flag Z is updated.

The zero bit Z of status register STA is set if the modified value of A is zero, otherwise the bit Z is cleared. The Z bit is set e.g. if the B[4:0]-th bit of A is not writable, its value is zero and all other bits of A are cleared.

The program counter PC is incremented by the value 4.

28.17.7.56CLRB Instruction

Syntax	Operation	Status	Duration
CLRB A, B	A[B[4:0]] ← 0	Z	1 instruction cycle

Clear the B[4:0]-th bit (B ∈ XOREG) of an operand A (A ∈ XOREG). Only the bits 0 to 4 of operand B are used as bit index of operand A and the other bits of B are ignored. If the value B[4:0] is greater than or equal to W, the operation of CLRB does not modify operand A but the status flag Z is updated.

The zero bit Z of status register STA is set if the modified value of A is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.57XCHB Instruction

Syntax	Operation	Status	Duration
XCHB A, B	A[B[4:0]] ← CY	Z, CY	1 instruction cycle

Exchange the B[4:0]-th bit (B ∈ XOREG) of an operand A (A ∈ XOREG) with the CY bit in the status register. Only the bits 0 to 4 of operand B are used as bit index of operand A and the other bits of B are ignored. If the value B[4:0] is greater than or equal to W, the operation of XCHB does not modify operand A but the status flag Z is updated and the bit CY is cleared.

The zero bit Z of status register STA is set if the modified value of A is zero, otherwise the zero bit is cleared.

The program counter PC is incremented by the value 4.

28.17.7.58JMP Instruction

Syntax	Operation	Status	Duration
JMP C	PC ← C << 2	-	1 instruction cycle but not faster than NPS clock cycles due to pipeline flushing.

Execute unconditional jump to the memory location C (C ∈ ALIT).

The program counter PC is loaded with literal C.

28.17.7.59JBS Instruction

Syntax	Operation	Status	Duration
JBS A, B, C	PC ← C << 2 if A[B] is set PC ← PC + 4 if A[B] is clear	-	1 instruction cycle but if the jump is executed, it is not faster than NPS clock cycles due to pipeline flushing.

Execute conditional jump to the memory location C (C $\in$ ALIT).

The program counter PC is loaded with literal C if the bit at position B (B $\in$ BITLIT) of operand A (A $\in$ OREG) is set.

Otherwise, if the bit is cleared the program counter PC is incremented by the value 4.

28.17.7.60JBC Instruction

Syntax	Operation	Status	Duration
JBC A, B, C	PC ← C << 2 if A[B] is clear PC ← PC + 4 if A[B] is set	-	1 instruction cycle but if the jump is executed it is not faster than NPS clock cycles due to pipeline flushing.

Execute conditional jump to the memory location C (C $\in$ ALIT).

The program counter PC is loaded with literal C if the bit at position B (B $\in$ BITLIT) of operand A (A $\in$ OREG) is cleared.

Otherwise, if the bit is set the program counter PC is incremented by the value 4.

28.17.7.61CALL Instruction

Syntax	Operation	Status	Duration
CALL C	R7 ← R7 + 4; MEM(R7[RAW+USR+1:2])[RAW+USR+1:0] ← PC + 4; PC ← C << 2; SP_CNT ← SP_CNT + 1	EN	2 instruction cycles but not faster than 2*NPS clock cycles due to pipeline flushing.

Call subprogram at memory location C (C $\in$ ALIT).

The stack pointer register R7 is incremented by the value 4.

The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register.

After the stack pointer is incremented, the incremented value of the PC is transferred to the top of the stack.

The program counter PC is loaded with literal C.

The **SP_CNT** bit field inside the **MCS[i]_CH[x]_CTRL** register is incremented.

If an overflow on the **SP_CNT** bit field occurs, the *STK_ERR[i]_IRQ* is raised.

If an overflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL_STAT** is set, the channel current MCS channel is disabled by clearing the EN bit of STA.

If an overflow on the SP_CNT bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL_STAT** is set, the memory write operation of the incremented PC is discarding.

28.17.7.62RET Instruction

Syntax	Operation	Status	Duration
RET	PC ← MEM(R7[RAW+USR+1:2])[RAW+USR+1:2] << 2; R7 ← R7 - 4; SP_CNT ← SP_CNT - 1	EN	2 instruction cycles but not faster than 2*NPS clock cycles due to pipeline flushing.

Return from subprogram.

The program counter PC is loaded with current value on the top of the stack.

Finally, the stack pointer register R7 is decremented by the value 4.

The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register.

The SP_CNT bit field inside the **MCS[i]_CH[x]_CTRL** register is decremented.

If an underflow on the SP_CNT bit field occurs, the *STK_ERR[i]_IRQ* is raised.

If an underflow on the SP_CNT bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL** is set, the channel current MCS channel is disabled by clearing the **EN** bit of **STA**.

28.17.7.63JMPI Instruction

Syntax	Operation	Status	Duration
JMPI	PC ← R6[RAW+USR+1:2] << 2	-	1 instruction cycle but not faster than NPS clock cycles due to pipeline flushing.

Execute indirect unconditional jump to the memory location provided in register R6. The destination address is only defined by the bits 2 to RAW+USR+1 of R6 and the other bits are ignored.

The program counter PC is loaded with (R6[RAW+USR+1:2] << 2).

28.17.7.64JBSI Instruction

Syntax	Operation	Status	Duration
JBSI A, B	PC ← R6[RAW+USR+1:2] << 2 if A[B] is set PC ← PC + 4 if A[B] is clear	-	1 instruction cycle but if the jump is executed it is not faster than NPS clock cycles due to pipeline flushing.

Execute indirect conditional jump to the memory location provided in register R6. The destination address is only defined by the bits 2 to RAW+USR+1 of R6 and the other bits are ignored.

The program counter PC is loaded with (R6[RAW+USR+1:2] << 2) only if the bit at position B ($B_{\in}$ XBITLIT) of operand A ($A_{\in}$ XOREG) is set.

Otherwise, if the bit is cleared the program counter PC is incremented by the value 4.

28.17.7.65JBCI Instruction

Syntax	Operation	Status	Duration
JBCI A, B	PC ← R6[RAW+USR+1:2] << 2 if A[B] is set PC ← PC + 4 if A[B] is clear	-	1 instruction cycle but if the jump is executed it is not faster than NPS clock cycles due to pipeline flushing.

Execute indirect conditional jump to the memory location provided in register R6. The destination address is only defined by the bits 2 to RAW+USR+1 of R6 and the other bits are ignored.

The program counter PC is loaded with (R6[RAW+USR+1:2] << 2) only if the bit at position B (B$\in$ XBITLIT) of operand A (A$\in$ XOREG) is cleared.

Otherwise, if the bit is set the program counter PC is incremented by the value 4.

28.17.7.66CALLI Instruction

Syntax	Operation	Status	Duration
CALLI	R7 ← R7 + 4; MEM(R7[RAW+USR+1:2])[RAW+USR+1:0] ← PC + 4; PC ← R6[RAW+USR+1:2] << 2; SP_CNT ← SP_CNT + 1	EN	2 instruction cycles but not faster than 2*NPS clock cycles due to pipeline flushing.

Call subprogram indirectly, where the register R6 is identifying the target memory location. The destination address is only defined by the bits 2 to RAW+USR+1 of R6 and the other bits are ignored.

The stack pointer register R7 is incremented by the value 4.

The memory location for the top of the stack is identified by the bits 2 to RAW+1 of the stack pointer register.

After the stack pointer is incremented, the incremented value of the PC is transferred to the top of the stack.

The program counter PC is loaded with (R6[RAW+USR+1:2] << 2),

The **SP_CNT** bit field inside the **MCS[i]_CH[x]_CTRL** register is incremented.

If an overflow on the **SP_CNT** bit field occurs, the *STK_ERR[i]_IRQ* is raised.

If an overflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL_STAT** is set, the channel current MCS channel is disabled by clearing the **EN** bit of **STA**.

If an overflow on the **SP_CNT** bit field occurs and the bit **HLT_SP_OFL** of register **MCS[i]_CTRL_STAT** is set, the memory write operation of the incremented PC is discarding.

28.17.7.67WURM Instruction

Syntax	Operation	Status	Duration
WURM A, B, C	Wait until register match.	CWT	Suspends current MCS channel. If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode, the worst case latency for reactivating a prioritized MCS channel is 2+NPS clock cycles. This is the delay between the match event of the corresponding WURM instruction and the beginning of the following MCS instruction.

Suspend current MCS channel until the following register match condition occurs:

A = (B AND MASK),

whereas A ∈ OREG, B ∈ OREG, AND is a bitwise AND operation with bitmask MASK.

The bits 16 to 23 of MASK are set to true and the bits 0 to 15 are copied from the instructions literal C ∈ MSKLIT. If the match condition evaluates to true, the suspended MCS channel is resumed and the program counter PC is incremented by the value 4 meaning that the MCS channel continues its program. However, if the match condition is true at the beginning of the instruction execution, the instruction does not suspend the channel and the program counter PC is incremented by the value 4.

At the beginning of the instruction execution the CWT bit in the register STA is always cleared. After the execution of the instruction the CWT bit is updated in order to show if the instruction finished successfully (CWT = 0) or it was canceled by the CPU (CWT = 1).

This instruction can be used to wait for one or more trigger events generated by other MCS channels or the CPU. In this case register B is the trigger register **STRG**, A is a general purpose register holding the bits with the trigger condition to wait for and C is the bitmask that enables trigger bits of interest. The trigger bits can be set by other MCS channels with a write access (e.g. using a MOVL instruction) to the **STRG** register or the CPU with a write access to the **MCS[i]_STRG** register. The trigger bits are not cleared automatically by hardware after resuming an MCS channel, but they have to be cleared explicitly with a write access to the register **CTRG** by the MCS channel or with a write access to the register **MCS[i]_CTRG** by the CPU. Please note that more than one channel can wait for the same trigger bit to continue.

The instruction can also be used to wait on a specific time/angle event provided by the TBU. In this case register B is the interesting TBU register TBU_TS0 or TBU_TS1, register A is a general purpose register holding the value to wait for and bitmask C should be set to 0xFFFF.

28.17.7.68WURMX Instruction

Syntax	Operation	Status	Duration
WURMX A, B	Wait until extended register match.	CWT	Suspends current MCS channel. If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode, the worst case latency for reactivating a prioritized MCS channel is 2+NPS clock cycles. This is the delay between the match event of the corresponding WURMX instruction and the beginning of the following MCS instruction.

Suspend current MCS channel until the following register match condition occurs:

A = B AND R6,

whereas A ∈ OREG, B ∈ WXREG, and AND is a bitwise AND operation.

If the match condition evaluates to true, the suspended MCS channel is resumed and the program counter PC is incremented by the value 4 meaning that the MCS channel continues its program. However, if the match condition is true at the beginning of the instruction execution, the instruction does not suspend the channel and the program counter PC is incremented by the value 4.

At the beginning of the instruction execution the CWT bit in the register STA is always cleared. After the execution of the instruction the CWT bit is updated in order to show if the instruction finished successfully (CWT = 0) or it was canceled by the CPU (CWT = 1).

28.17.7.69WURCX Instruction

Syntax	Operation	Status	Duration
WURCX A, B	Wait until extended register change.	CWT	Suspends current MCS channel. If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode, the worst case latency for reactivating a prioritized MCS channel is 2+NPS clock cycles. This is the delay between the match event of the corresponding WURCX instruction and the beginning of the following MCS instruction.

Suspend current MCS channel until the following register change condition occurs:

$A \neq B$ AND R6,

whereas $A \in$ OREG, $B \in$ WXREG, and AND is a bitwise AND operation.

If the change condition evaluates to true, the suspended MCS channel is resumed and the program counter PC is incremented by the value 4 meaning that the MCS channel continues its program. However, if the change condition is true at the beginning of the instruction execution, the instruction does not suspend the channel and the program counter PC is incremented by the value 4.

At the beginning of the instruction execution the CWT bit in the register STA is always cleared. After the execution of the instruction the CWT bit is updated in order to show if the instruction finished successfully (CWT = 0) or it was canceled by the CPU (CWT = 1).

The WURCX instruction can be used for observation of volatile registers (e.g. register DSTAX) in order to react on status signal changes.

28.17.7.70WUCE Instruction

Syntax	Operation	Status	Duration
WUCE A, B	Wait until cyclic event.	CWT	Suspends current MCS channel. If the MCS is configured in Single Prioritization or Multiple Prioritization Scheduling Mode, the worst case latency for reactivating a prioritized MCS channel is 2+NPS clock cycles. This is the delay between the match event of the corresponding WURE instruction and the beginning of the following MCS instruction.

Suspend current MCS channel until a cyclic event compare matches.

The meaning of a cyclic event is described in section ‘Cyclic Event Compare’. The WUCE instruction can be used to synchronize an MCS program to a cyclic event generated by a TBU channel. If the event is in the future, the MCS channel suspends until the event occurs. If the event is in the past, the WUCE instruction is finished immediately.

The cyclic event compare is used to detect time base overflows and to guarantee, that a compare match event can be set up for the future even when the time base will first overflow and then reach the compare value. Please note, that for a correct behavior of this cyclic event compare, the compare value must not be specified larger/smaller than half of the range of the total time base value (0x7FFFFF).

The actual implementation of the WUCE implementation simply performs the subtraction B - A with each clock cycle and it suspends the MCS channel as long as bit W-1 of the subtraction result is set. If the subtration result is cleared the MCS channel is resumed immediatly.

In order to setup a WUCE instruction correctly, the counting direction of the TBU channel has to be considered. If the TBU channel is counting forward (incrementing), the operand A (A$_{\in}$ OREG) must refer the compare value and operand B (B$_{\in}$ OREG) must refer the desired TBU counter register (e.g. TBU_TS0). On the other hand, if the TBU channel is counting backward (decrementing), the operand A refers the desired TBU counter register and operand B refers the compare value.

If the comparison condition evaluates to true, the suspended MCS channel is resumed and the program counter PC is incremented by the value 4 meaning that the MCS channel continues its program. However, if the condition is true at the beginning of the instruction execution, the instruction does not suspend the channel and the program counter PC is incremented by the value 4.

At the beginning of the instruction execution the CWT bit in the register STA is always cleared. After the execution of the instruction the CWT bit is updated in order to show if the instruction finished successfully (CWT = 0) or it was canceled by the CPU (CWT = 1).

28.17.7.71NOP Instruction

Syntax	Operation	Status	Duration
NOP	-	-	1 instruction cycle

No operation is performed.

The program counter PC is incremented by the value 4.

28.17.8 MCS Internal Register Description

This section describes the MCS internal registers that can be directly addressed with the MCS instruction set. Many of the registers can also be addressed by the CPU but with another Register Label (for details see section 15.11). Some of the internal registers are also shared between neighboring MCS channels.

28.17.8.1 MCS Internal Register Overview

Table 65 MCS Internal Register Overview

Register Name	Description	see Page
R[y] (y: 0…7)	General Purpose Register y	391
RS[y] (y: 0…7)	Mirror of succeeding channels register R[y]	391
STA	Status Register	392
ACB	ARU Control Bit Register	395
CTRG	Clear Trigger Bits Register	396
STRG	Set Trigger Bits Register	400
TBU_TS0	TBU Timestamp TS0 Register	400
TBU_TS1	TBU Timestamp TS1 Register	401
TBU_TS2	TBU Timestamp TS2 Register	401
MHB	Memory High Byte Register	401
GMI0	GTM Module Interrupt 0 Register	402

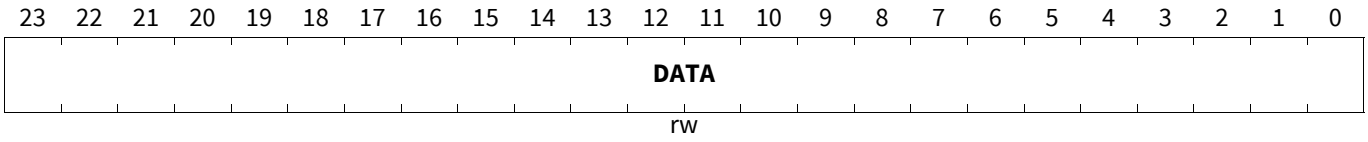
Table 65 MCS Internal Register Overview (cont'd)

Register Name	Description	see Page
GMI1	GTM Module Interrupt 1 Register	403
DSTA	DPLL Status Register	405
DSTAX	DPLL Extended Status Register	406

28.17.9 MCS Internal Register Description

28.17.9.1 Register R[y]

Ry (index)
General Purpose Register y **(0_H+y)** **Reset Value: 000000_H**



Field	Bits	Type	Description
DATA	23:0	rw	**Data field of general purpose register**

*Note: Register **R4** is also used as destination register of upper multiplication result from instructions MULU and MULS.*

*Note: Register **R5** is also used as offset register for the instructions MRDIO and MWRIO.*

*Note: Register **R6** is also used as a mask register for the instruction WURMX and WURCX.*

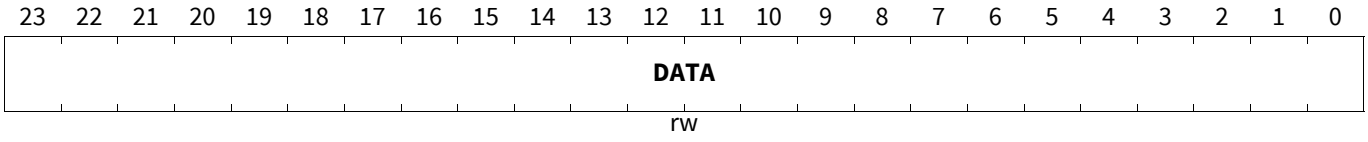
*Note: Register **R6** is also used as address destination register for the instructions JMPI, JBSI, JBCI, and CALLI.*

*Note: Register **R6** used also as index/address register for indirect ARU addressing instructions.*

*Note: Register **R7** is also used as stack pointer register, if stack operations are used in the MCS micro program.*

28.17.9.2 Register RS[y]

RSy (index)
Mirror of succeeding channels register R[y] **(10_H+y)** **Reset Value: 000000_H**



Field	Bits	Type	Description
DATA	23:0	rw	**Data field of general purpose register**

*Note: The register **RS[y]** (with y = 0…7) mirrors the internal general purpose register **R[y]** of the succeeding MCS channel. The successor of MCS channel T-1 is MCS channel 0.*

*Note: The registers **RS[y]** can only be accessed if bit **EN_XOREG** of register **MCS[i]_CTRL_STAT** is set.*

28.17.9.3 Register STA

STA
Status Register **(08_H)** **Reset Value: 000000_H**

23	22	21	20	19	18	17	16
Reserved					SP_CNT		
		r				r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					SAT	CWT	CAT	N	V	Z	CY	MCA	ERR	IRQ	EN
r					r	r	r	r	r	r	r	rw	rw	rw	rw

Field	Bits	Type	Description
EN	0	rw	**Enable current MCS channel** 0_B Disable current MCS channel. 1_B Enable current MCS channel.
IRQ	1	rw	**Trigger IRQ** 0_B No triggered IRQ signal. 1_B Trigger IRQ signal. Note: An MCS channel triggers an IRQ by writing value 1 to bit IRQ. Writing a value 0 to this bit does not cancel the IRQ, and thus has no effect. Note: This bit mirrors bit 0 of the register **MCS[i]_CH[x]_IRQ_NOTIFY**. Note: The IRQ bit can only be cleared by CPU, by writing a 1 to the corresponding **MCS[i]_CH[x]_IRQ_NOTIFY** register (see **Register MCS[i]_CH[x]_IRQ_NOTIFY**). Note: An MCS channel can read the IRQ bit in order to determine the current state of the IRQ handling. The MCS channel reads a value 1 if an IRQ was released but not cleared by CPU. If an MCS channel reads a value 0 no IRQ was released or it has been cleared by CPU. Note: If NPS > 5 and an MCS program triggers the IRQ (e.g. by MOVL STA, 0x2) the actual interrupt event is delayed by NPS-5 clock cycles, which means that an immediate read of the interrupt notify flag (e.g. by MOV R2, STA) may signalize the state of the IRQ bit before the trigger.

Field	Bits	Type	Description
ERR	2	rw	**Set Error Signal** 0_B No Error occurred. 1_B Error occurred. Note: The **ERR** bit of an MCS channel reflects an Error status that may be caused by one of the following conditions: • MCS channel sets the ERR bit by software (e.g. with instruction ORL STA, 0x4) • ECC RAM Error occurred while accessing the connected RAM (also disables the MCS channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**) • Decoding an instruction with an invalid opcode (also disables the MCS channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**) • A memory address range overflow occurred (also disables the MCS channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**) • Division by zero resulting from a **DIVU**, **DIVS** or **MODU** instruction (also disables the MCS channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**). • MCS channel wants to write to a GPR that is write protected by register **MCS[i]_REG_PROT** (also disables the MCS channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**) • MCS channel wants to write to a protected memory range defined by an address range protector (ARP) of the sub module CCM (also disables the MCS channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**) • MCS channel performs an invalid AEI bus master access while the bit field **HLT_AEIM_ERR** of register **MCS[i]_CTRL_STAT** is set (also disables the MCS channel by clearing bit **EN** and the first error occurred updates bit field **ERR_SRC_ID** of register **MCS[i]_CTRL_STAT**) Note: If the ERR bit is set due to a memory address range overflow any read or write access to the RAM is blocked. Note: If the GTM includes a MON sub module, the ERR signal is always captured by this module. Note: An MCS channel can set the error bit by writing value 1 to bit ERR. Writing a value 0 to this bit does not cancel the error signal, and thus has no effect. In Addition, writing a value 1 to ERR always triggers the ERR interrupt, independently from the current state of the error signal. Note: The ERR bit can only be cleared by CPU, by writing a 1 to the **MCS[i]_ERR** register (see **Register MCS[i]_ERR**). Note: An MCS channel can read the ERR bit in order to determine the current state of the error signal. The MCS channel reads a value 1 if an ERR occurred previously, but not cleared by CPU. If an MCS channel reads a value 0 no error was set or it has been cleared by CPU.

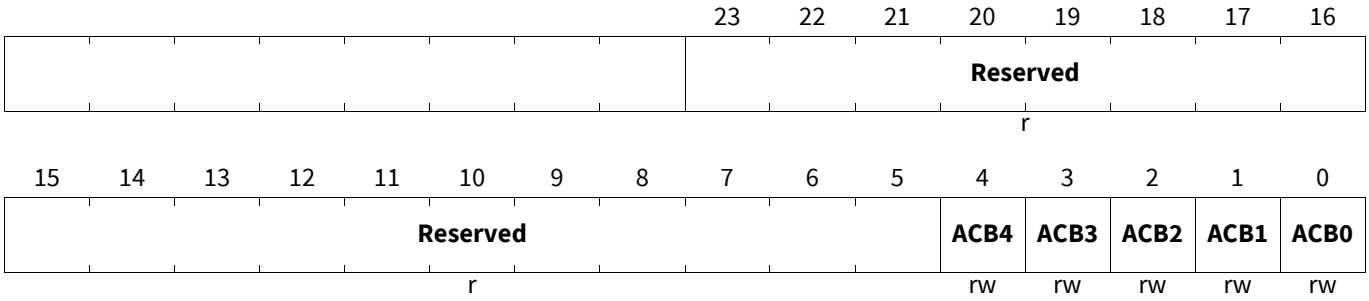
Field	Bits	Type	Description
MCA	3	rw	**MON Activity signaling for MCS channel** 0_B No activity signaled to sub module MON. 1_B Activity singled to sub module MON. *Note: When this bit is set the corresponding channel in the MON sub module register* **MON_ACTIVITY** *is set (see register MON_ACTIVITY_0. This bit is automatically cleared after writing it by the MCS channel program.*
CY	4	r	**Carry bit** The carry bit is updated by several arithmetic and logic instructions. In arithmetic operations, the carry bit indicates an unsigned under/overflow.
Z	5	r	**Zero bit** The zero bit is updated by several arithmetic, logic and data transfer instructions to indicate a result of zero.
V	6	r	**Overflow bit** The overflow bit is updated by arithmetic instructions in order to indicate a signed under/overflow.
N	7	r	**Negative bit** The negative bit is updated by arithmetic instructions in order to indicate a negative result.
CAT	8	r	**Cancel ARU transfer bit** 0_B No cancellation request for ARU transfer. 1_B CPU requests cancellation for ARU transfer. Note: This bit is always cleared at the beginning of the execution of a blocking ARU instruction.
CWT	9	r	**Cancel WURM instruction bit** 0_B Last WURM instruction was not canceled 1_B CPU canceled last WURM instruction of channel. Note: This bit is updated after each WURM instruction and it should be evaluated immediately after the WURM instruction. Otherwise, the CPU could set the bit leading to a bad status information in the MCS program.
SAT	10	r	**Successful ARU transfer bit** 0_B ARU data transfer failed. 1_B ARU data transfer finished successfully. Note: This bit is always updated after the execution of ARU instructions in order to show if the ARU data transfer was successful or not. In the case of non-blocking ARU instructions (NARD, NARDI) a cleared SAT flag signalizes that the data source has no data available and in the case of blocking ARU instructions (ARD, ARDI, AWR, AWRI) a cleared SAT flag signalizes that the data transfer was canceled by the CPU.
Reserved	15:11	r	**Reserved** Read as zero, shall be written as zero.
SP_CNT	18:16	r	**Stack pointer counter value** Note: Actual stack depth of channel. The bit field is incremented on behalf of a CALL or PUSH instruction and decremented on behalf of a RET or POP instruction. The MCS channel STK_ERR_IRQ is raised, when an overflow or underflow is detected on this bit field.

Field	Bits	Type	Description
Reserved	23:19	r	**Reserved** Read as zero, shall be written as zero.

Note: Writing to bits of the register STA with instructions that do implicitly a read-modify-write operation (e.g. "ANDL STA 0xFFFFFE" or "OR STA R0") is dangerous, since writing back the possibly modified content of the read access (which reflects status information) may cause undesirable results. A secure way for writing to bits of the register STA is to use instructions that do not read the content of STA (e.g. "MOVL STA 0x0, MOV STA R1, CLRB STA R0, or SETB STA R1").

28.17.9.4 Register ACB

ACB
ARU Control Bit Register **(09_H)** **Reset Value: 000000_H**



Field	Bits	Type	Description
ACB0	0	rw	**ARU Control bit 0.** Note: This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 48 of the ARU word.
ACB1	1	rw	**ARU Control bit 1.** Note: This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 49 of the ARU word.
ACB2	2	rw	**ARU Control bit 2.** Note: This bit is updated by each ARUread access and its value is sent to ARU by each ARU write access on bit 50 of the ARU word.
ACB3	3	rw	**ARU Control bit 3.** Note: This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 51 of the ARU word.
ACB4	4	rw	**ARU Control bit 4.** Note: This bit is updated by each ARU read access and its value is sent to ARU by each ARU write access on bit 52 of the ARU word.
Reserved	23:5	r	**Read as zero, shall be written as zero.**

28.17.9.5 Register CTRG

CTRG
Clear Trigger Bits Register **($0A_H$)** **Reset Value: 000000_H**

23	22	21	20	19	18	17	16
TRG23	**TRG22**	**TRG21**	**TRG20**	**TRG19**	**TRG18**	**TRG17**	**TRG16**
rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRG15	**TRG14**	**TRG13**	**TRG12**	**TRG11**	**TRG10**	**TRG9**	**TRG8**	**TRG7**	**TRG6**	**TRG5**	**TRG4**	**TRG3**	**TRG2**	**TRG1**	**TRG0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TRG0	0	rw	**Trigger bit 0** READ access: state of current trigger bit **TRG0 if EN_TIM_FOUT** = 0 state of input signal **TIM[i]_CH0_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG0**
TRG1	1	rw	**Trigger bit 1** READ access: state of current trigger bit **TRG1** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i]_CH1_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG1**
TRG2	2	rw	**Trigger bit 2** READ access: state of current trigger bit **TRG2** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i]_CH2_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG2**
TRG3	3	rw	**Trigger bit 3** READ access: state of current trigger bit **TRG3** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i]_CH3_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG3**

Field	Bits	Type	Description
TRG4	4	rw	**Trigger bit 4** READ access: state of current trigger bit **TRG4** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i]_CH4_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG4**
TRG5	5	rw	**Trigger bit 5** READ access: state of current trigger bit **TRG5** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i]_CH5_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG5**
TRG6	6	rw	**Trigger bit 6** READ access: state of current trigger bit **TRG6** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i]_CH6_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG6**
TRG7	7	rw	**Trigger bit 7** READ access: state of current trigger bit **TRG7** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i]_CH7_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG7**
TRG8	8	rw	**Trigger bit 8** READ access: state of current trigger bit **TRG8** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i+1]_CH0_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG8**
TRG9	9	rw	**Trigger bit 9** READ access: state of current trigger bit **TRG9** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i+1]_CH1_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG9**

Field	Bits	Type	Description
TRG10	10	rw	**Trigger bit 10** READ access: state of current trigger bit **TRG10** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i+1]_CH2_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG10**
TRG11	11	rw	**Trigger bit 11** READ access: state of current trigger bit **TRG11** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i+1]_CH3_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG11**
TRG12	12	rw	**Trigger bit 12** READ access: state of current trigger bit **TRG12** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i+1]_CH4_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG12**
TRG13	13	rw	**Trigger bit 13** READ access: state of current trigger bit **TRG13** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i+1]_CH5_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG13**
TRG14	14	rw	**Trigger bit 14** READ access: state of current trigger bit **TRG14** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i+1]_CH6_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG14**
TRG15	15	rw	**Trigger bit 15** READ access: state of current trigger bit **TRG15** if **EN_TIM_FOUT** = 0 state of input signal **TIM[i+1]_CH7_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B do nothing 1_B clear trigger bit **TRG15**
TRG16	16	rw	**Trigger bit 16** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: clear trigger bit

Field	Bits	Type	Description
TRG17	17	rw	**Trigger bit 17** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: clear trigger bit
TRG18	18	rw	**Trigger bit 18** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: clear trigger bit
TRG19	19	rw	**Trigger bit 19** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: clear trigger bit
TRG20	20	rw	**Trigger bit 20** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: clear trigger bit
TRG21	21	rw	**Trigger bit 21** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: clear trigger bit
TRG22	22	rw	**Trigger bit 22** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: clear trigger bit
TRG23	23	rw	**Trigger bit 23** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: clear trigger bit

*Note: The trigger bits **TRGx** are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the **STRG** register, in the case of an MCS channel or the **MCS[i]_STRG** register in the case of the CPU. Clearing a trigger bit can be performed with the **CTRG** register, in the case of an MCS channel or the **MCS[i]_CTRG** register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS channels or the CPU. An MCS channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.*

*Note: Besides setting the trigger bits with register **STRG/MCS[i]_STRG**, the k-th trigger bit **TRGk** (with k < 16) can also be set by the external capture event that is enabled by the k-th bit of register **CCM[i]_EXT_CAP_EN**. If bit k bit is disabled, the k-th trigger bit TRGk can only be set by MCS or CPU.*

*Note: The result of a read access to this register differs in dependency of the bit field **EN_TIM_FOUT** of register **MCS[i]_CTRL_STAT**.*

28.17.9.6 Register STRG

STRG
Set Trigger Bits Register **($0B_H$)** **Reset Value: 000000_H**

23	22	21	20	19	18	17	16
TRG23	**TRG22**	**TRG21**	**TRG20**	**TRG19**	**TRG18**	**TRG17**	**TRG16**
rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRG15	**TRG14**	**TRG13**	**TRG12**	**TRG11**	**TRG10**	**TRG9**	**TRG8**	**TRG7**	**TRG6**	**TRG5**	**TRG4**	**TRG3**	**TRG2**	**TRG1**	**TRG0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TRGx (x=0-23)	x	rw)	**Trigger bit x** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: set trigger bit

*Note: The trigger bits **TRGx** are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the **STRG** register, in the case of an MCS channel or the **MCS[i]_STRG** register in the case of the CPU. Clearing a trigger bit can be performed with the **CTRG** register, in the case of an MCS channel or the **MCS[i]_CTRG** register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS channels or the CPU. An MCS channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.*

*Note: Besides setting the trigger bits with register **STRG/MCS[i]_STRG**, the k-th trigger bit **TRGx** (with k < 16) can also be set by the external capture event that is enabled by the k-th bit of register **CCM[i]_EXT_CAP_EN**. If bit k bit is disabled, the k-th trigger bit **TRGx** can only be set by MCS or CPU.*

28.17.9.7 Register TBU_TS0

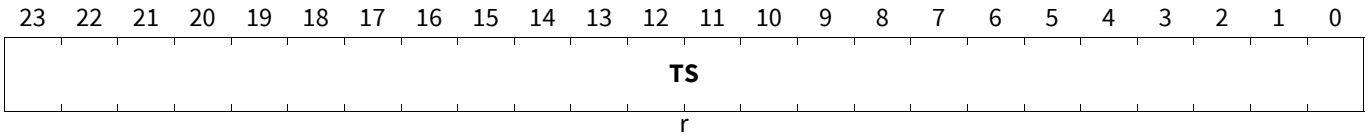
TBU_TS0
TBU Timestamp TS0 Register **($0C_H$)** **Reset Value: 000000_H**

23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS																							
r																							

Field	Bits	Type	Description
TS	23:0	r	**Current TBU time stamp 0.**

28.17.9.8 Register TBU_TS1

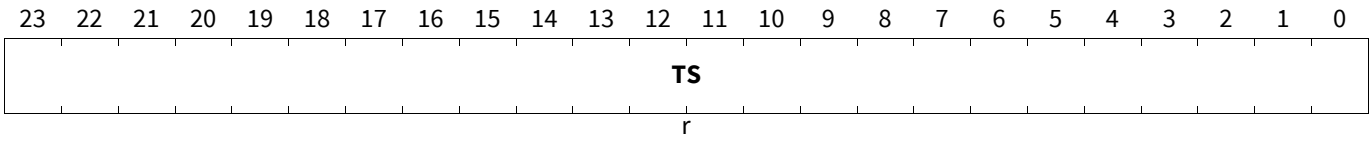
TBU_TS1
TBU Timestamp TS1 Register **($0D_H$)** **Reset Value: 000000_H**



Field	Bits	Type	Description
TS	23:0	r	**Current TBU time stamp 1.**

28.17.9.9 Register TBU_TS2

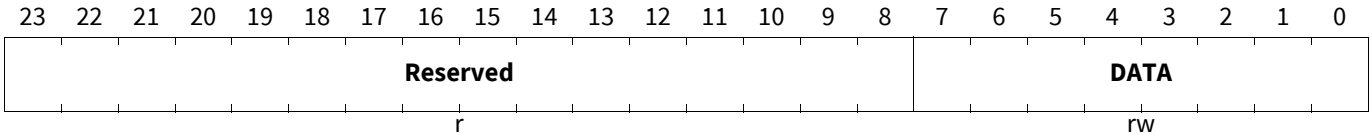
TBU_TS2
TBU Timestamp TS2 Register **($0E_H$)** **Reset Value: 000000_H**



Field	Bits	Type	Description
TS	23:0	r	**Current TBU time stamp 2**

28.17.9.10Register MHB

MHB
Memory High Byte Register **($0F_H$)** **Reset Value: 000000_H**



Field	Bits	Type	Description
DATA	7:0	rw	**High Byte of a memory transfer**
Reserved	23:8	r	**Reserved** Read as zero, shall be written as zero.

28.17.9.11Register GMI0

GMI0
GTM Module Interrupt 0 Register **(18_H)** **Reset Value: 000000_H**

23	22	21	20	19	18	17	16
ATOM_CH7_IRQ	ATOM_CH6_IRQ	ATOM_CH5_IRQ	ATOM_CH4_IRQ	ATOM_CH3_IRQ	ATOM_CH2_IRQ	ATOM_CH1_IRQ	ATOM_CH0_IRQ
rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOM_CH14_IRQ	TOM_CH12_IRQ	TOM_CH10_IRQ	TOM_CH8_IRQ	TOM_CH6_IRQ	TOM_CH4_IRQ	TOM_CH2_IRQ	TOM_CH0_IRQ	TIM_CH7_IRQ	TIM_CH6_IRQ	TIM_CH5_IRQ	TIM_CH4_IRQ	TIM_CH3_IRQ	TIM_CH2_IRQ	TIM_CH1_IRQ	TIM_CH0_IRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TIM_CHx_IRQ (x=0-7)	x	rw	**TIM[i]_CHx IRQ** READ access: IRQ signal *TIM[i]_CHx_IRQ.* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TOM_CH0_IRQ	8	rw	**TOM[i]_CH0 or TOM[i]_CH1 IRQ** READ access: Logical OR conjunction of IRQ signals *T0M[i]_CH0_IRQ* and *T0M[i]_CH1_IRQ*. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TOM_CH2_IRQ	9	rw	**TOM[i]_CH2 or TOM[i]_CH3 IRQ** READ access: Logical OR conjunction of IRQ signals *T0M[i]_CH2_IRQ* and *T0M[i]_CH3_IRQ*. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TOM_CH4_IRQ	10	rw	**TOM[i]_CH4 or TOM[i]_CH5 IRQ** READ access: Logical OR conjunction of IRQ signals *T0M[i]_CH4_IRQ* and *T0M[i]_CH5_IRQ*. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TOM_CH6_IRQ	11	rw	**TOM[i]_CH6 or TOM[i]_CH7 IRQ** READ access: Logical OR conjunction of IRQ signals *T0M[i]_CH6_IRQ* and *T0M[i]_CH7_IRQ*. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.

Field	Bits	Type	Description
TOM_CH8_IRQ	12	rw	**TOM[i]_CH8 or TOM[i]_CH9 IRQ** READ access: Logical OR conjunction of IRQ signals *T0M[i]_CH8_IRQ* and *T0M[i]_CH9_IRQ*. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TOM_CH10_IRQ	13	rw	**TOM[i]_CH10 or TOM[i]_CH11 IRQ** READ access: Logical OR conjunction of IRQ signals *T0M[i]_CH10_IRQ* and *T0M[i]_CH11_IRQ*. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TOM_CH12_IRQ	14	rw	**TOM[i]_CH12 or TOM[i]_CH13 IRQ** READ access: Logical OR conjunction of IRQ signals *T0M[i]_CH12_IRQ* and *T0M[i]_CH13_IRQ*. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TOM_CH14_IRQ	15	rw	**TOM[i]_CH14 or TOM[i]_CH15 IRQ** READ access: Logical OR conjunction of IRQ signals *T0M[i]_CH14_IRQ* and *T0M[i]_CH15_IRQ*. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
ATOM_CHx_IRQ (x=0-7)	16 + x	rw	**ATOM[i]_CHx IRQ** READ access: IRQ signal ATOM[i]_CHx_IRQ. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.

28.17.9.12Register GMI1

GMI1

GTM Module Interrupt 1 Register **(19_H)** **Reset Value: 000000_H**

23	22	21	20	19	18	17	16
MCS0_CH7_IRQ	**MCS0_CH6_IRQ**	**MCS0_CH5_IRQ**	**MCS0_CH4_IRQ**	**MCS0_CH3_IRQ**	**MCS0_CH2_IRQ**	**MCS0_CH1_IRQ**	**MCS0_CH0_IRQ**
rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTA_IP1_CH7_IRQ	**TTA_IP1_CH6_IRQ**	**TTA_IP1_CH5_IRQ**	**TTA_IP1_CH4_IRQ**	**TTA_IP1_CH3_IRQ**	**TTA_IP1_CH2_IRQ**	**TTA_IP1_CH1_IRQ**	**TTA_IP1_CH0_IRQ**	**MCS_IP1_CH7_IRQ**	**MCS_IP1_CH6_IRQ**	**MCS_IP1_CH5_IRQ**	**MCS_IP1_CH4_IRQ**	**MCS_IP1_CH3_IRQ**	**MCS_IP1_CH2_IRQ**	**MCS_IP1_CH1_IRQ**	**MCS_IP1_CH0_IRQ**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MCS_IP1_CHx_IRQ (x=0-7)	x	rw	**MCS[i+1]_CHx IRQ** READ access: IRQ signal *MCS[i+1]_CH0_IRQ* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TTA_IP1_CH0_IRQ	8	rw	**Neighboring TIM, TOM, ATOM IRQs** READ access: Logical OR conjunction of IRQ signals *TIM[i+1]_CH0_IRQ*, *TOM[i+1]_CH0_IRQ*, *TOM[i+1]_CH1_IRQ*, and *ATOM[i+1]_CH0_IRQ* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TTA_IP1_CH1_IRQ	9	rw	**Neighboring TIM, TOM, ATOM IRQs** READ access: Logical OR conjunction of IRQ signals *TIM[i+1]_CH1_IRQ*, *TOM[i+1]_CH2_IRQ*, *TOM[i+1]_CH3_IRQ*, and *ATOM[i+1]_CH1_IRQ* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TTA_IP1_CH2_IRQ	10	rw	**Neighboring TIM, TOM, ATOM IRQs** READ access: Logical OR conjunction of IRQ signals *TIM[i+1]_CH2_IRQ*, *TOM[i+1]_CH4_IRQ*, *TOM[i+1]_CH5_IRQ*, and *ATOM[i+1]_CH2_IRQ* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TTA_IP1_CH3_IRQ	11	rw	**Neighboring TIM, TOM, ATOM IRQs** READ access: Logical OR conjunction of IRQ signals *TIM[i+1]_CH3_IRQ*, *TOM[i+1]_CH6_IRQ*, *TOM[i+1]_CH7_IRQ*, and *ATOM[i+1]_CH3_IRQ* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TTA_IP1_CH4_IRQ	12	rw	**Neighboring TIM, TOM, ATOM IRQs** READ access: Logical OR conjunction of IRQ signals *TIM[i+1]_CH4_IRQ*, *TOM[i+1]_CH8_IRQ*, *TOM[i+1]_CH9_IRQ*, and *ATOM[i+1]_CH4_IRQ* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TTA_IP1_CH5_IRQ	13	rw	**Neighboring TIM, TOM, ATOM IRQs** READ access: Logical OR conjunction of IRQ signals *TIM[i+1]_CH5_IRQ*, *TOM[i+1]_CH10_IRQ*, *TOM[i+1]_CH10_IRQ*, and *ATOM[i+1]_CH5_IRQ* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.

Field	Bits	Type	Description
TTA_IP1_CH6_IRQ	14	rw	**Neighboring TIM, TOM, ATOM IRQs** READ access: Logical OR conjunction of IRQ signals *TIM[i+1]_CH6_IRQ*, *TOM[i+1]_CH12_IRQ*, *TOM[i+1]_CH13_IRQ*, and *ATOM[i+1]_CH6_IRQ* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TTA_IP1_CH7_IRQ	15	rw	**Neighboring TIM, TOM, ATOM IRQs** READ access: Logical OR conjunction of IRQ signals *TIM[i+1]_CH7_IRQ*, *TOM[i+1]_CH14_IRQ*, *TOM[i+1]_CH15_IRQ*, and *ATOM[i+1]_CH7_IRQ* WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
MCS0_CHx_IRQ (x=0-7)	16 + x	rw	**MCS0_CHx IRQs.** READ access: IRQ signal *MCS0_CHx_IRQ* .WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.

28.17.9.13Register DSTA

DSTA
DPLL Status Register **($1A_H$)** **Reset Value: 000000_H**

23	22	21	20	19	18	17	16
Reserved			CDTI	SISI	SASI	TISI	TASI
r			rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STA_S								STA_T							
r								r							

Field	Bits	Type	Description
STA_T	7:0	r	**Status Trigger FSM** Actual status of DPLL Trigger FSM. The description of the FSM states can be found in section “DPLL_STA”.
STA_S	15:8	r	**Status State FSM** Actual status of DPLL State FSM. The description of the FSM states can be found in section “DPLL_STA”.
TASI	16	rw	**Trigger Active Slope Interrupt.** READ access: IRQ signal TASI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.

Field	Bits	Type	Description
TISI	17	rw	**Trigger Inactive Slope Interrupt.** READ access: IRQ signal TISI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
SASI	18	rw	**State Active Slope Interrupt.** READ access: IRQ signal SASI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
SISI	19	rw	**State Inactive Slope Interrupt.** READ access: IRQ signal SISI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
CDTI	20	rw	**Calculation of trigger duration interrupt.** READ access: IRQ signal CDTI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
Reserved	23:21	r	**Reserved** Read as zero, shall be written as zero.

Note: This register is only implemented in MCS instance 0. In other MCS instances, a read access always returns 0 and a write access is always ignored.

28.17.9.14Register DSTAX

DSTAX
DPLL Extended Status Register **($1B_H$)** **Reset Value: 000000_H**

23	22	21	20	19	18	17	16
Reserved			**CDTI**	**SISI**	**SASI**	**TISI**	**TASI**
r			rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												**INC_CNT2_FLAG**	**INC_CNT1_FLAG**	**STA_FLAG_S**	**STA_FLAG_T**
r												rw	rw	rw	rw

Field	Bits	Type	Description
STA_FLAG_T	0	rw	**DPLL status trigger flag** DPLL status trigger flag as described in bit field definition **STA_FLAG_T** of register **DPLL_STA_FLAG** (section ‘Register DPLL_STA_FLAG’).

Field	Bits	Type	Description
STA_FLAG_S	1	rw	**DPLL status state flag** DPLL status state flag as described in bit field definition **STA_FLAG_S** of register **DPLL_STA_FLAG** (section 'Register DPLL_STA_FLAG').
INC_CNT1_FLAG	2	rw	**DPLL INC_CNT1 Flag** DPLL status state flag as described in bit field definition **INC_CNT1_FLAG** of register **DPLL_STA_FLAG** (section 'Register DPLL_STA_FLAG').
INC_CNT2_FLAG	3	rw	**DPLL INC_CNT2 Flag** DPLL status state flag as described in bit field definition **INC_CNT2_FLAG** of register **DPLL_STA_FLAG** (section 'Register DPLL_STA_FLAG').
Reserved	15:4	r	**Reserved** Read as zero, shall be written as zero.
TASI	16	rw	**Trigger Active Slope Interrupt.** READ access: IRQ signal TASI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
TISI	17	rw	**Trigger Inactive Slope Interrupt.** READ access: IRQ signal TISI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
SASI	18	rw	**State Active Slope Interrupt.** READ access: IRQ signal SASI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
SISI	19	rw	**State Inactive Slope Interrupt.** READ access: IRQ signal SISI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
CDTI	20	rw	**Calculation of trigger duration interrupt.** READ access: IRQ signal CDTI of DPLL. WRITE access: 0_B do nothing 1_B issue hw_clear on the connected IRQs.
Reserved	23:21	r	**Reserved** Read as zero, shall be written as zero.

Note: ***Note**: This register is only implemented in MCS instance 0. In other MCS instances, a read access always returns 0 and a write access is always ignored.*

28.17.10 MCS Configuration Register Overview

The MCS Configuration registers of the MCS module are accessible by the AEI bus interface. Some of these registers simply mirror MCS Internal registers to the AEI. Details can be found in the table below and in the individual register descriptions.

Table 66 MCS Configuration Register Overview$_B$

Register Name	Description	see Page
MCS[i]_CH[x]_CTRL	MCSi channel x control register	409
MCS[i]_CH[x]_ACB	MCSi channel x ARU control Bit register	411
MCS[i]_CH[x]_MHB	MCSi channel x memory high byte register	412
MCS[i]_CH[x]_PC	MCSi channel x program counter register	410
MCS[i]_CH[x]_R[y] (y=0..7)	MCSi channel x general purpose register y (y=0..7)	411
MCS[i]_CH[x]_IRQ_NOTIFY	MCSi channel x interrupt notification register	413
MCS[i]_CH[x]_IRQ_EN	MCSi channel x interrupt enable register	414
MCS[i]_CH[x]_IRQ_FORCINT	MCSi channel x force interrupt register	414
MCS[i]_CH[x]_IRQ_MODE	MCSi channel x IRQ mode configuration register	415
MCS[i]_CH[x]_EIRQ_EN	MCSi channel x error interrupt enable register	416
MCS[i]_CTRL_STAT	MCSi control and status register	417
MCS[i]_REG_PROT	MCSi write protection register	419
MCS[i]_CTRG	MCSi clear trigger control register	420
MCS[i]_STRG	MCSi set trigger control register	421
MCS[i]_RESET	MCSi reset register	422
MCS[i]_ERR	MCSi error register	425
MCS[i]_CAT	MCSi cancel ARU transfer instruction	423
MCS[i]_CWT	MCSi cancel WURM instruction	424

28.17.11 MCS Configuration Register Description

28.17.11.1Register MCS[i]_CH[x]_CTRL

MCSi Channel x Control Register

MCSi_CHx_CTRL (i=0-9;x=0-7)
MCSi Channel x Control Register **($0F0020_H$+i*1000_H+x*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													SP_CNT		
r													rh		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					SAT	CWT	CAT	N	V	Z	CY	0	ERR	IRQ	EN
r					rh	rh	rh	rh	rh	rh	rh	r	rh	rh	rwh

Field	Bits	Type	Description
EN	0	rwh	**Enable MCS-channel** *Note: Enabling or disabling of an MCS-channel is synchronized to the ending of an instruction, and thus it may take several clock cycles, e.g. active memory transfers or pending WURM transfers have to be finished before disabling the MCS-channel. The internal state of a channel can be obtained by reading bit EN.* *Note: To disable an MCS channel reliably, the EN bit should be cleared, followed by setting the CAT and CWT bits in order to cancel any pending WURM or ARU instructions.* *Note: The EN bit is write protected during RAM reset phase.* 0_B Disable current MCS-channel 1_B Enable current MCS-channel
IRQ	1	rh	**Interrupt state** This bit is read only, and it mirrors the internal IRQ state. 0_B No interrupt pending in MCS-channel x 1_B Interrupt is pending in MCS-channel x
ERR	2	rh	**Error state** This bit is read only, and it mirrors the internal error state. 0_B No error signal pending in MCS-channel x 1_B Error signal is pending in MCS-channel x
CY	4	rh	**Carry bit state** This bit is read only and it mirrors the internal carry flag CY.
Z	5	rh	**Zero bit state** This bit is read only and it mirrors the internal zero flag Z.
V	6	rh	**Overflow bit state** This bit is read only and it mirrors the internal carry flag V.
N	7	rh	**Negative bit state** This bit is read only and it mirrors the internal zero flag N.

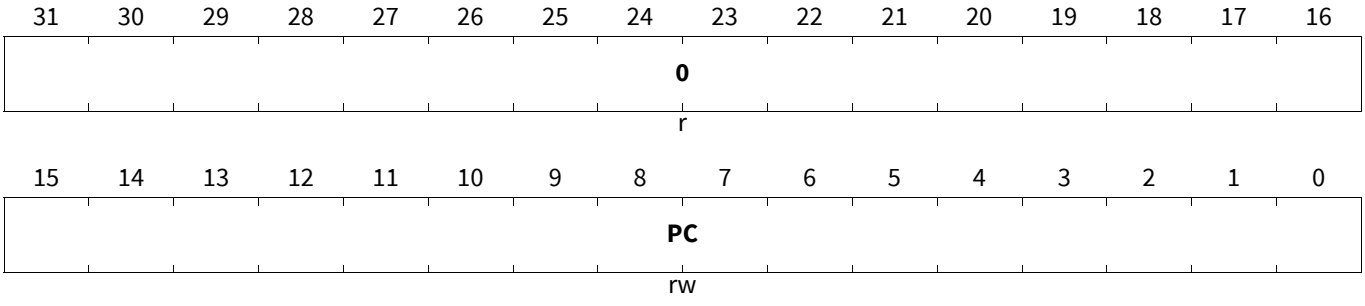
Field	Bits	Type	Description
CAT	8	rh	**Cancel ARU transfer state** This bit is read only and it mirrors the internal cancel ARU transfer status flag CAT.
CWT	9	rh	**Cancel WURM instruction state** This bit is read only and it mirrors the internal cancel WURM instruction status flag CWT.
SAT	10	rh	**Successful ARU transfer bit** This bit is read only, and it mirrors the internal state of the ARU transfer status flag SAT. 0_B ARU data transfer failed 1_B ARU data transfer finished successfully
SP_CNT	18:16	rh	**Stack pointer counter value** Actual stack depth of channel. The bit field is incremented on behalf of a CALL or PUSH instruction and decremented on behalf of a RET or POP instruction. The MCS channel STK_ERR_IRQ is raised, when an overflow or underflow is detected on this bit field.
0	3, 15:11, 31:19	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.2Register MCS[i]_CH[x]_PC

MCSi Channel x Program Counter Register

MCSi_CHx_PC (i=0-9;x=0-7)

MCSi Channel x Program Counter Register($0F0040_H$+i*1000_H+x*80_H) **Reset Value: Table 67**



Field	Bits	Type	Description
PC	15:0	rw	**Current Program Counter** *Note: The program counter is only writable if the corresponding MCS-channel is disabled. The bits 0 and 1 are always written as zeros.* *Note: The actual width of the program counter depends on the MCS configuration. The actual width is RAW+USR+2 bits meaning that only the bits 0 to RAW+USR+1 are available and the other bits (RAW+USR+2 to 31) are reserved.*
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

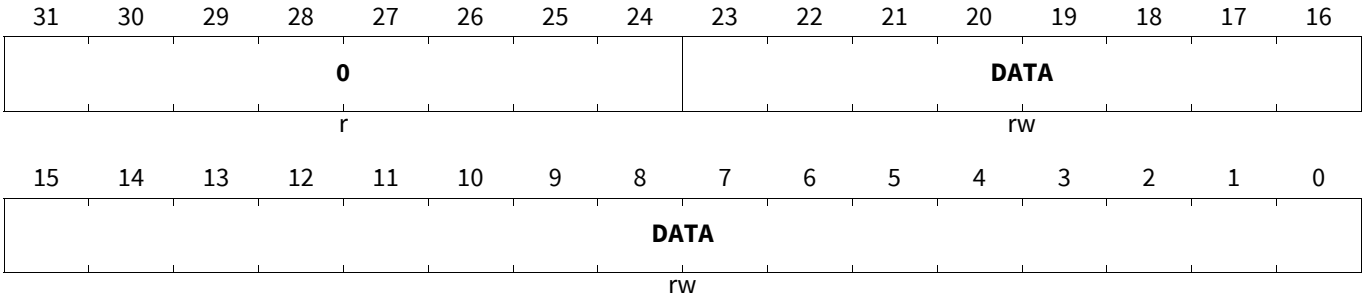
Table 67 Reset Values of MCSi_CHx_PC (i=0-9;x=0-7)

Reset Type	Reset Value	Note
Application Reset	4 * x	

28.17.11.3Register MCS[i]_CH[x]_R[y]

MCSi Channel x General Purpose Register y

MCSi_CHx_Ry (i=0-9;x=0-7;y=0-7)
MCSi Channel x General Purpose Register y($0F0000_H$+i*1000_H+x*80_H+y*4) Application Reset Value: $0000\ 0000_H$

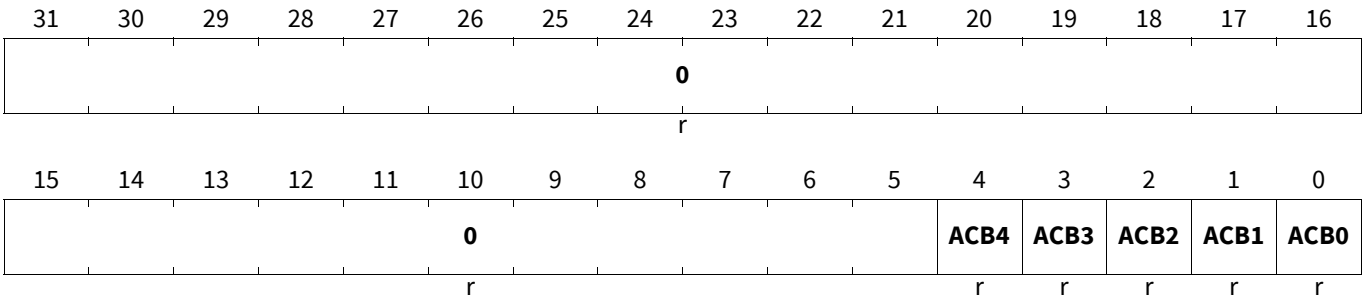


Field	Bits	Type	Description
DATA	23:0	rw	**Data of general purpose register R[y]** *Note: This register is the same as described in internal register section.* *Note: For the register MCS[i]_CH[x]_R6 (see internal register section) an additional write protection during an active ARDI or NARDI instruction is applied*
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.4Register MCS[i]_CH[x]_ACB

MCSi Channel x ARU Control Bit Register

MCSi_CHx_ACB (i=0-9;x=0-7)
MCSi Channel x ARU Control Bit Register($0F0024_H$+i*1000_H+x*80_H) Application Reset Value: $0000\ 0000_H$



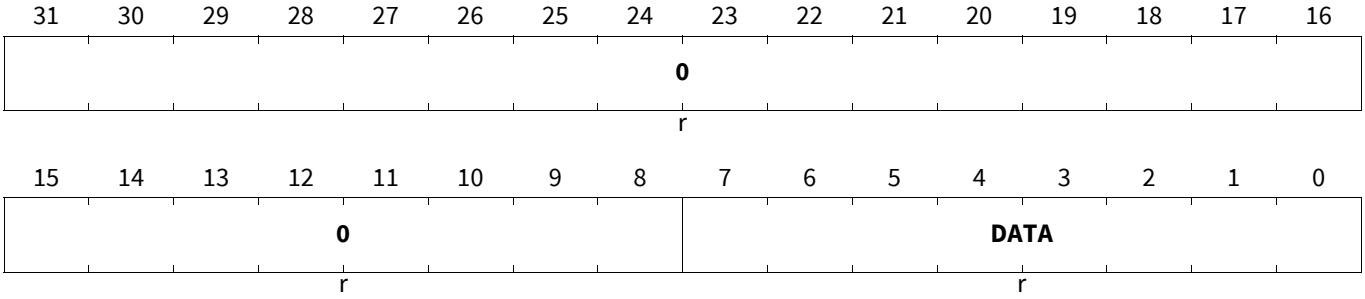
Field	Bits	Type	Description
ACB0	0	r	**ARU Control bit 0** This bit is read only, and it mirrors the internal state.
ACB1	1	r	**ARU Control bit 1** This bit is read only, and it mirrors the internal state.
ACB2	2	r	**ARU Control bit 2** This bit is read only, and it mirrors the internal state.
ACB3	3	r	**ARU Control bit 3** This bit is read only, and it mirrors the internal state.
ACB4	4	r	**ARU Control bit 4** This bit is read only, and it mirrors the internal state.
0	31:5	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.5Register MCS[i]_CH[x]_MHB

MCSi Channel x Memory High Byte Register

MCSi_CHx_MHB (i=0-9;x=0-7)

MCSi Channel x Memory High Byte Register($0F003C_H$+i*1000_H+x*80_H) Application Reset Value: 0000 0000_H

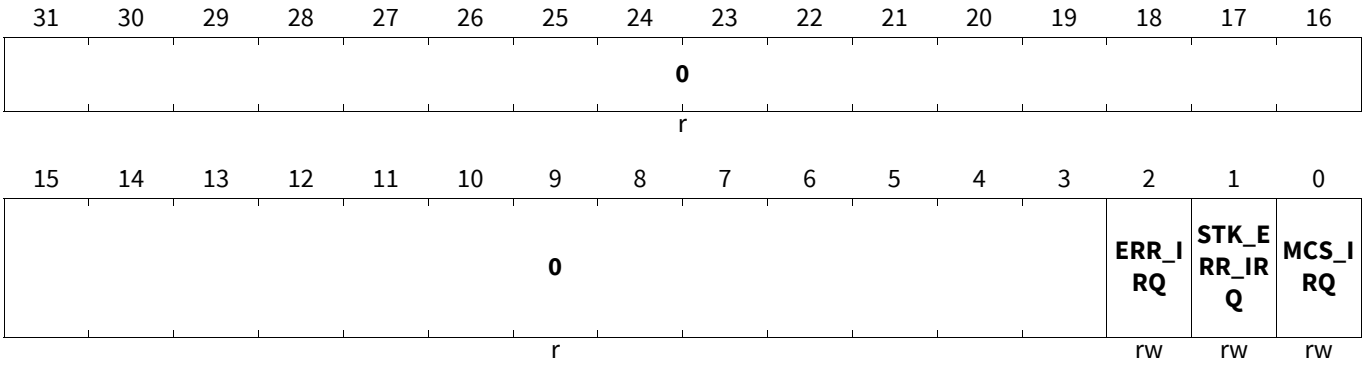


Field	Bits	Type	Description
DATA	7:0	r	**Data of memory high bit register MHB**
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.6Register MCS[i]_CH[x]_IRQ_NOTIFY

MCSi Channel x Interrupt Notification Register

MCSi_CHx_IRQ_NOTIFY (i=0-9;x=0-7)
MCSi Channel x Interrupt Notification Register($0F0044_H$+i*1000_H+x*80_H) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
MCS_IRQ	0	rw	**Interrupt request by MCS-channel x** *Note: This bit will be cleared on a CPU write access with a value of '1'. A read access leaves the bit unchanged.* *Note: By writing a '1' to this register, the IRQ flag in the MCS channel status register STA is cleared.* 0_B No IRQ released 1_B IRQ released by MCS-channel
STK_ERR_IRQ	1	rw	**Stack counter overflow/underflow of channel x** *Note: This bit will be cleared on a CPU write access with a value of '1'. A read access leaves the bit unchanged.* 0_B No IRQ released 1_B A stack counter overflow or underflow occurred
ERR_IRQ	2	rw	**MCS channel x ERR interrupt** *Note: If the ERR bit of register STA is triggered, the ERR_IRQ will also be set.* *Note: This bit will be cleared on a CPU write access with a value of '1'. A read access leaves the bit unchanged.* 0_B No IRQ released 1_B MCS-channel ERR IRQ released
0	31:3	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.7Register MCS[i]_CH[x]_IRQ_EN

MCSi Channel x Interrupt Enable Register

MCSi_CHx_IRQ_EN (i=0-9;x=0-7)

MCSi Channel x Interrupt Enable Register($0F0048_H$+i*1000_H+x*80_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													ERR_IRQ_EN	STK_ERR_IRQ_EN	MCS_IRQ_EN
r													rw	rw	rw

Field	Bits	Type	Description
MCS_IRQ_EN	0	rw	**MCS channel x MCS_IRQ interrupt enable** 0_B Disable interrupt 1_B Enable interrupt
STK_ERR_IRQ_EN	1	rw	**MCS channel x STK_ERR_IRQ interrupt enable** 0_B Disable interrupt 1_B Enable interrupt
ERR_IRQ_EN	2	rw	**MCS channel x ERR_IRQ interrupt enable** 0_B Disable interrupt 1_B Enable interrupt
0	31:3	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.8Register MCS[i]_CH[x]_IRQ_FORCINT

MCSi Channel x Force Interrupt Register

MCSi_CHx_IRQ_FORCINT (i=0-9;x=0-7)

MCSi Channel x Force Interrupt Register($0F004C_H$+i*1000_H+x*80_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													TRG_ERR_IRQ	TRG_STK_ERR_IRQ	TRG_MCS_IRQ
r													rw	rw	rw

Field	Bits	Type	Description
TRG_MCS_IRQ	0	rw	**Trigger IRQ bit in MCS_CH_[x]_IRQ_NOTIFY register by software** *Note: This bit is cleared automatically after write.* *Note: This bit is write protected by bit RF_PROT of register GTM_CTRL.* 0_B No interrupt triggering 1_B Assert corresponding bit in MCS[i]_CH[x]_IRQ_NOTIFY register
TRG_STK_ERR_IRQ	1	rw	**Trigger IRQ bit in MCS_CH_[x]_IRQ_NOTIFY register by software** *Note: This bit is cleared automatically after write.* *Note: This bit is write protected by bit RF_PROT of register GTM_CTRL.* 0_B No interrupt triggering 1_B Assert corresponding bit in MCS[i]_CH[x]_IRQ_NOTIFY register
TRG_ERR_IRQ	2	rw	**Trigger IRQ bit in MCS_CH_[x]_IRQ_NOTIFY register by software** *Note: This bit is cleared automatically after write.* *Note: This bit is write protected by bit RF_PROT of register GTM_CTRL.* 0_B No interrupt triggering 1_B Assert corresponding bit in MCS[i]_CH[x]_IRQ_NOTIFY register
0	31:3	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.9Register MCS[i]_CH[x]_IRQ_MODE

MCSi Channel x Interrupt Mode Configuration Register

MCSi_CHx_IRQ_MODE (i=0-9;x=0-7)
MCSi Channel x Interrupt Mode Configuration Register($0F0050_H$+i*1000_H+x*80_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														IRQ_MODE	
r														rw	

Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** The interrupt modes are described in **Section 28.4.5**. 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.10Register MCS[i]_CH[x]_EIRQ_EN

MCSi Channel x Error Interrupt Enable Register

MCSi_CHx_EIRQ_EN (i=0-9;x=0-7)
MCSi Channel x Error Interrupt Enable Register($0F0054_H$+i*1000_H+x*80_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													ERR_EIRQ_EN	STK_ERR_EIRQ_EN	MCS_EIRQ_EN0
r													rw	rw	rw

Field	Bits	Type	Description
MCS_EIRQ_EN0	0	rw	**MCS channel x MCS_EIRQ error interrupt enable** 0_B Disable error interrupt 1_B Enable error interrupt
STK_ERR_EIRQ_EN	1	rw	**MCS channel x STK_ERR_IRQ error interrupt enable** 0_B Disable error interrupt 1_B Enable error interrupt
ERR_EIRQ_EN	2	rw	**MCS channel x ERR_EIRQ error interrupt enable** 0_B Disable error interrupt 1_B Enable error interrupt
0	31:3	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.11Register MCS[i]_CTRL_STAT

MCSi Control and Status Register

MCSi_CTRL_STAT (i=0-9)
MCSi Control and Status Register **($0F0064_H$+i*1000_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					HLT_AEIM_ERR	EN_XOREG	EN_TIM_FOUT	0	ERR_SRC_ID			0		HLT_SP_OFL	RAM_RST
r					rw	rw	rw	r	r			r		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				SCD_CH				0						SCD_MODE	
r				rw				r						rw	

Field	Bits	Type	Description
SCD_MODE	1:0	rw	**Select MCS scheduling mode** 00_B Accelerated Scheduling 01_B Round Robin Scheduling 10_B Single Priority Scheduling 11_B Multiple Priority Scheduling
SCD_CH	11:8	rw	**Channel selection for scheduling algorithm** MCS-channel identifier used by several scheduling modes. *Note: The actual width of the bit field SCD_CH is calculated as ceiling[log_2(T+1)].* Unused MSBs are reserved and read as zero.
RAM_RST	16	rw	**RAM reset bit** *Note: The RAM reset initializes the memory content with zeros. RAM access and enabling of MCS channels is disabled during active RAM reset.* *Note: This bit is only writable if the bit RF_PROT in register GTM_CTRL is cleared, and all MCS-channels are disabled.* *Note: The actual reset value of this bit depends on the silicon vendor configuration. The reset value is 1, if the RAM reset is performed together with the sub-module reset, otherwise, the reset value is 0. If the reset value is 1, the reset value is changed to 0 by hardware when the RAM reset has finished.* 0_B READ: no RAM reset is active / WRITE: do nothing 1_B READ: MCS currently resets RAM content / WRITE: trigger RAM reset

Field	Bits	Type	Description
HLT_SP_OFL	17	rw	**Halt on stack pointer overflow** 0_B No halt on MCS-channel stack pointer counter over/underflow 1_B MCS-channel is disabled if a stack pointer counter over/underflow occurs
ERR_SRC_ID	22:20	r	**Error source identifier** *Note: This register is updated once, if an error was detected by the MCS. The register is set to its initial value 000_B after each write access to an existing ERR bit in register MCS[i]_ERR. If multiple errors occur, ERR_SRC_ID is holding the first type of error which has occurred.* 000_B No HW generated Error occurred 001_B Detected ECC error 010_B Detected memory overflow 011_B Detected invalid opcode 100_B Divide by zero 101_B Invalid register write access to GPR from write protected channel 110_B Invalid memory write access to protected memory region 111_B Invalid AEI bus master access
EN_TIM_FOUT	24	rw	**Enable routing of TIM[i]_CH[x]_F_OUT signal** 0_B Read access to register CTRG/MCS[i]_CTRG provides state of the internal trigger registers 1_B Read access to register CTRG/MCS[i]_CTRG provides state of the external signal TIM[i]_CH[x]_F_OUT
EN_XOREG	25	rw	**Enable extended register set** *Note: If the extended operation register sets are disabled, the MCS instructions can only use the subset OREG of the register set as arguments in the instructions. In this case, the upper address bits in the instructions are always read as zeros, which leads to unexpected results of the MCS program if arguments A or B refer to a register that is not part of OREG.* 0_B Extended operation register sets XOREG, BAREG, and WXREG are disabled 1_B Extended operation register sets XOREG, BAREG, and WXREG are enabled

Field	Bits	Type	Description
HLT_AEIM_ERR	26	rw	**Halt on AEI bus master error** *Note: If the register HLT_AEIM_ERR is set, and an MCS channel x is executing an invalid bus master access, the MCS channel x is halted, the ERR bit of its register STA is set, and the bit field ERR_SRC_ID of this register is updated.* *Note: If the bus master is accessing a slave that does not insert wait cycles (e.g. register access), it takes two additional clock cycles until the MCS channel is halted. Within that time span, the MCS channel can continue with its program execution, depending on the selected scheduling mode.* *Note: The registers AEIM_XPT_STA and AEIM_XPT_ADDR of the GTM sub-module CCM are always updated on the first invalid AEI bus master access, independently of the state of HLT_AEIM_ERR* 0_B Ignore invalid AEI bus master access 1_B Halt MCS-channel on invalid AEI bus master access
0	7:2, 15:12, 19:18, 23, 31:27	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.12Register MCS[i]_REG_PROT

MCSi Write Protection Register

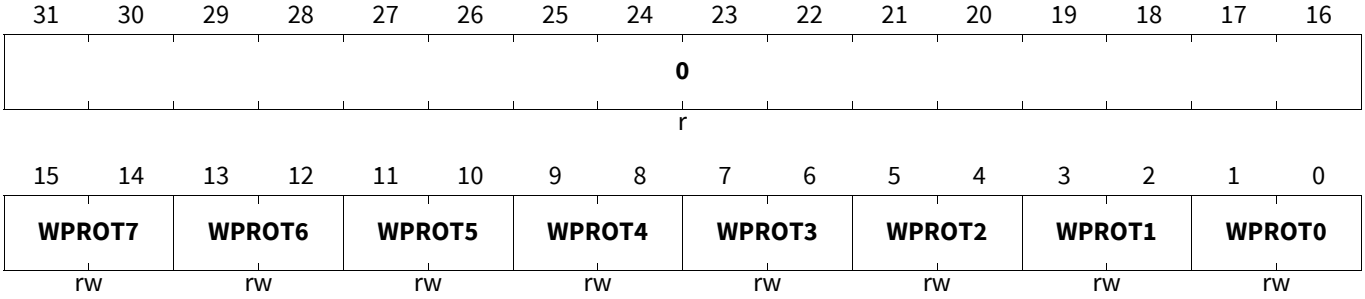
*Note: Only the first T bit fields of this register (bit 0 to 2*T-1) are functionally implemented. The other bits (bit 2*T to 31) are reserved bit fields.*

Note: The predecessor channel of MCS channel 0 is MCS channel T-1.

Note: If an MCS channel x is writing to a general purpose register that is write protected by register MCS[i]_REG_PROT the ERR bit of the register STA is set, the MCS channel x is halted and the ERR_SRC_ID bit field of register MCS[i]_CTRL_STAT is updated.

Note: This register is only writable if the bit RF_PROT in register GTM_CTRL is cleared

MCSi_REG_PROT (i=0-9)
MCSi Write Protection Register **($0F0060_H$+i*1000_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
WPROTx (x=0-7)	2*x+1:2*x	rw	**Register Write Protection of MCS-channel x** 00_B No register write protection activated 01_B Predecessor MCS channel cannot write to its RS[y] registers 10_B Current MCS channel cannot write to its R[y] registers 11_B Reserved
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.13Register MCS[i]_CTRG

MCSi Clear Trigger Control Register

Note: The trigger bits TRGx are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the STRG register, in the case of an MCS-channel or the MCS[i]_STRG register in the case of the CPU. Clearing a trigger bit can be performed with the CTRG register, in the case of an MCS-channel or the MCS[i]_CTRG register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.

Note: Besides setting the trigger bits with register STRG/MCS[i]_STRG, the k-th trigger bit TRGx (with k < 16) can also be set by the external capture event that is enabled by the k-th bit of register GTM_EXT_CAP_EN_[i]. If bit k bit is disabled, the k-th trigger bit TRGk can only be set by MCS or CPU.

Note: In the scheduling modes Accelerated Scheduling and Round Robin Scheduling, a write access to MCS[i]_CTRG may take up to T + 1 clock cycles, since the write access is scheduled to the next CPU time slot determined by the MCS scheduler. In the modes Single Prioritization Scheduling and Multiple Prioritization Scheduling, no upper limit access time for a write access to MCS[i]_CTRG can be guaranteed. The High Prioritized tasks have to be disabled in order to guarantee fast write access to MCS[i]_CTRG.

Note: Note: The result of a read access to this register differs in dependency of the bit field EN_TIM_FOUT of register MCS[i]_CTRL_STAT.

MCSi_CTRG (i=0-9;x=0-7)

MCSi Clear Trigger Control Register (0F0028$_H$+i*1000$_H$+x*80$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TRG23	TRG22	TRG21	TRG20	TRG19	TRG18	TRG17	TRG16
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRG15	TRG14	TRG13	TRG12	TRG11	TRG10	TRG9	TRG8	TRG7	TRG6	TRG5	TRG4	TRG3	TRG2	TRG1	TRG0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TRGk (k=0-7)	k	rw	**Trigger bit k** READ access: State of current trigger bit**TRGk if EN_TIM_FOUT**= 0 State of input signal**TIM[i]_CH**[**k**]**_F_OUT if EN_TIM_FOUT**= 1 WRITE access: 0_B Do nothing 1_B Clear trigger bit TRGx
TRGk (k=8-15)	k	rw	**Trigger bit k** READ access: State of current trigger bit **TRGk if EN_TIM_FOUT** = 0 State of input signal **TIM[i+1]_CH**[k-8]**_F_OUT if EN_TIM_FOUT** = 1 WRITE access: 0_B Do nothing 1_B Clear trigger bit TRGk
TRGk (k=16-23)	k	rw	**Trigger bit k** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: clear trigger bit
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.14Register MCS[i]_STRG

MCSi Set Trigger Control Register

Note: The trigger bits TRGx are accessible by all MCS channels as well as the CPU. Setting a trigger bit can be performed with the STRG register, in the case of an MCS-channel or the MCS[i]_STRG register in the case of the CPU. Clearing a trigger bit can be performed with the CTRG register, in the case of an MCS channel or the MCS[i]_CTRG register in the case of the CPU. Trigger bits can be used for signalizing specific events to MCS-channels or the CPU. An MCS-channel suspended with a WURM instruction can be resumed by setting the appropriate trigger bit.

Note: Besides setting the trigger bits with register STRG/MCS[i]_STRG, the k-th trigger bit TRGk (with k < 16) can also be set by the external capture event that is enabled by the k-th bit of register CCM[I]_EXT_CAP_EN. If bit k bit is disabled, the k-th trigger bit TRGk can only be set by MCS or CPU.

Note: In the scheduling modes Accelerated Scheduling and Round Robin Scheduling, a write access to MCS[i]_STRG may take up to T + 1 clock cycles, since the write access is scheduled to the next CPU time slot determined by the MCS scheduler. In the modes Single Prioritization Scheduling and Multiple Prioritization Scheduling, no upper limit access time for a write access to MCS[i]_STRG can be guaranteed. The High Prioritized tasks have to be disabled in order to guarantee fast write access to MCS[i]_STRG.

MCSi_STRG (i=0-9;x=0-7)
MCSi Set Trigger Control Register **(0F002C$_H$+i*1000$_H$+x*80$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TRG23	TRG22	TRG21	TRG20	TRG19	TRG18	TRG17	TRG16
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRG15	TRG14	TRG13	TRG12	TRG11	TRG10	TRG9	TRG8	TRG7	TRG6	TRG5	TRG4	TRG3	TRG2	TRG1	TRG0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TRGk (k=0-23)	k	rw	**Trigger bit k** 0_B READ: trigger bit is cleared / WRITE: do nothing 1_B READ: trigger bit is set / WRITE: set trigger bit
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.15Register MCS[i]_RESET

MCSi Reset Register

Note: Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits.

Note: The RSTx (x = 0 ...T-1) bits is cleared automatically after write access of CPU. All channel related registers of channel x are set to their reset values and channel operation is stopped immediately.

Note: Channel related registers of channel x are all registers MCS[i]_CH[x]_*, all MCS internal registers accessible by the corresponding channel, with exception of the common trigger register (accessed by MCS[i]_CTRG/MCS[i]_STRG) and the commonly used general purpose registers MCS[i]_CH[x]_R4 and MCS[i]_CH[x]_R5.

MCSi_RESET (i=0-9)

MCSi Reset Register **(0F0068$_H$+i*1000$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								RST7	RST6	RST5	RST4	RST3	RST2	RST1	RST0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RSTx (x=0-7)	x	rw	**Software reset of channel x** 0_B No action 1_B Reset channel
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.16Register MCS[i]_CAT

MCSi Cancel ARU Transfer Instruction Register

Note: Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits.

MCSi_CAT (i=0-9)

MCSi Cancel ARU Transfer Instruction Register(0F006C$_H$+i*1000$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								CAT7	CAT6	CAT5	CAT4	CAT3	CAT2	CAT1	CAT0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CATx (x=0-7)	x	rw	**Cancel ARU transfer for channel x** *Note: The CATx (x = 0..T-1) bit inside the STA register of the corresponding MCS channel is set, and any pending blocking ARU read or write request is canceled. The MCS channel resumes with the instruction after the blocking ARU transfer instruction.* *Note: The CATx (x = 0..T-1) bit is cleared by the corresponding MCS channel when the channel is entering a blocking ARU read or write instruction.* 0_B Do nothing 1_B Cancel any pending ARU read or write transfer
0	31:8	r	**Reserved** Read as zero, shall be written as zero

28.17.11.17Register MCS[i]_CWT

MCSi Cancel WURM Instruction Register

Note: Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits.

MCSi_CWT (i=0-9)
MCSi Cancel WURM Instruction Register (0F0070$_H$+i*1000$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								CWT7	CWT6	CWT5	CWT4	CWT3	CWT2	CWT1	CWT0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CWTx (x=0-7)	x	rw	**Cancel WURM instruction for channel x** *Note: The CWTx (x = 0..T-1) bit inside the STA register of the corresponding MCS channel is set, and any pending WURM instruction is canceled. The MC -channel resumes with the instruction after the WURM instruction.* *Note: The CWTx (x = 0..T-1) bit is cleared by the corresponding MCS channel when the channel reaches a WURM instruction.* 0_B Do nothing 1_B Cancel any pending WURM instruction
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.17.11.18Register MCS[i]_ERR

MCSi error register

Note: Only the first T bits of this register (bit 0 to T-1) are functionally implemented. The other bits (bit T to 31) are reserved bits.

MCSi_ERR (i=0-9)
MCSi error register **(0F007C$_H$+i*1000$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ERRx (x=0-7)	x	rw	**Error State of MCS-channel x** *Note: The CPU can read the ERRx (x = 0..T-1) bits in order to determine the current error state of the corresponding MCS-channel x.* *Note: The error state is also evaluated by the sub-module MON, if this module is available.* *Note: Writing the value 1 to this bit resets the corresponding error state, and resets the channel internal ERR bit in the STA and channel CTRL registers. Moreover, each write access to this bit also sets the ERR_SRC_ID bit field of register MCS[i]_CTRL_STAT to its reset value.* 0_B No error signal 1_B Error signal is pending
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.18 Memory Configuration (MCFG)

28.18.1 Overview

The Memory Configuration submodule (MCFG) is an infrastructure module that organizes physical memory blocks and maps them to the RAM ports 0 and 1 of available Multi Channel Sequencer (MCS) modules.

The following parameters are design variables for the MCFG hardware structure that can vary in its range for different devices:

Constant	Description	Value
MAW	Memory address width of a large physical memory block.	11
ERM	Enable RAM1 MSB (0 - RAM1 MSB disabled, 1 - RAM1 MSB enabled,)	0

It should be noted that the actual value of the parameter ERM can be obtained by the bit **ERM** of the register **CCM[i]_HW_CONF**.

Depending on the value of parameter ERM, the MCFG module assumes externally connected physical RAM modules with different sizes. If ERM = 0, MCFG assumes that each MCS instance provides a large physical memory block with 2^{MAW} memory locations each 32 bit wide which leads to a RAM module with 2^{MAW+2} (byte wise) memory addresses. Further each MCS instance provides a small physical memory block with 2^{MAW-1} memory locations each 32 bit wide leading to a RAM module with 2^{MAW+1} (byte wise) memory addresses. If ERM = 1, MCFG assumes that each MCS instance provides two large physical memory block each with 2^{MAW} memory locations each 32-bit leading to a RAM module with 2^{MAW+2} (byte wise) memory addresses.

In order to support different memory sizes for different MCS instances, the MCFG module provides three layout configurations for reorganization of memory pages mapped to the RAM ports of neighboring MCS modules. Figure **Figure 115** shows all layout configurations for the case that ERM = 0 and **Figure 117** shows the layout configurations for the case that ERM = 1. Each box in these pictures represents a physical memory block.

The layout configuration DEFAULT is always assigning a memory block of size 2^{MAW}x32 bits to MCS RAM port 0. Depending on ERM, RAM port 1 of each MCS is assigned to a memory block of size 2^{MAW-1}x32 bits (ERM = 0) or a memory block of size 2^{MAW}x32 bits (ERM = 1).

The layout configuration SWAP is swapping the memory block assigned to RAM port 1 of the current MCS instance with the memory block assigned to RAM port 0 of the successive MCS instance. If ERM = 0, this means that the memory of the current MCS instance is increased by 2^{MAW-1}x32 bits but the memory of the successor is decreased by 2^{MAW-1}x32 bits compared to the DEFAULT configuration. If ERM = 1, the SWAP configuration has no effect on the memory sizes of the individual MCS instances.

The layout configuration BORROW is borrowing the memory block assigned to RAM port 0 of the successive MCS instance for the current instance. This means, the memory of the current MCS module is increased by 2^{MAW}x32 bits but the memory of the successor is decreased by 2^{MAW}x32 bits compared to the DEFAULT configuration.

Considering the order the mentioned MCS modules, it should be noted that the successor of the last MCS instance is the first MCS instance MCS0.

The actual sizes of the memory pages mapped to the MCS RAM ports 0 and 1 depends on the layout configuration for of current instance MCS[i] and the layout configuration of the preceding memory instance MCS[i-1]. The sizes of these memory pages can be obtained by the layout parameters MP0 and MP1, as described in the specification of the MCS.

Figure 116 and **Figure 118** summarize the layout parameters MP0 and MP1 of MCS instance MCS[i] for the case that ERM = 0 and ERM = 1. Note that the predecessor of instance MCS0 is last available MCS instance.

The addressing of memory port 0 ranges from 0 to MP0-4 and the addressing of memory page 1 ranges from MP0 to MP1-4.

This document assumes that the GTM implementation embeds 8 MCS instances. However, the actual number of implemented MCS instances is specified in device specific appendix.

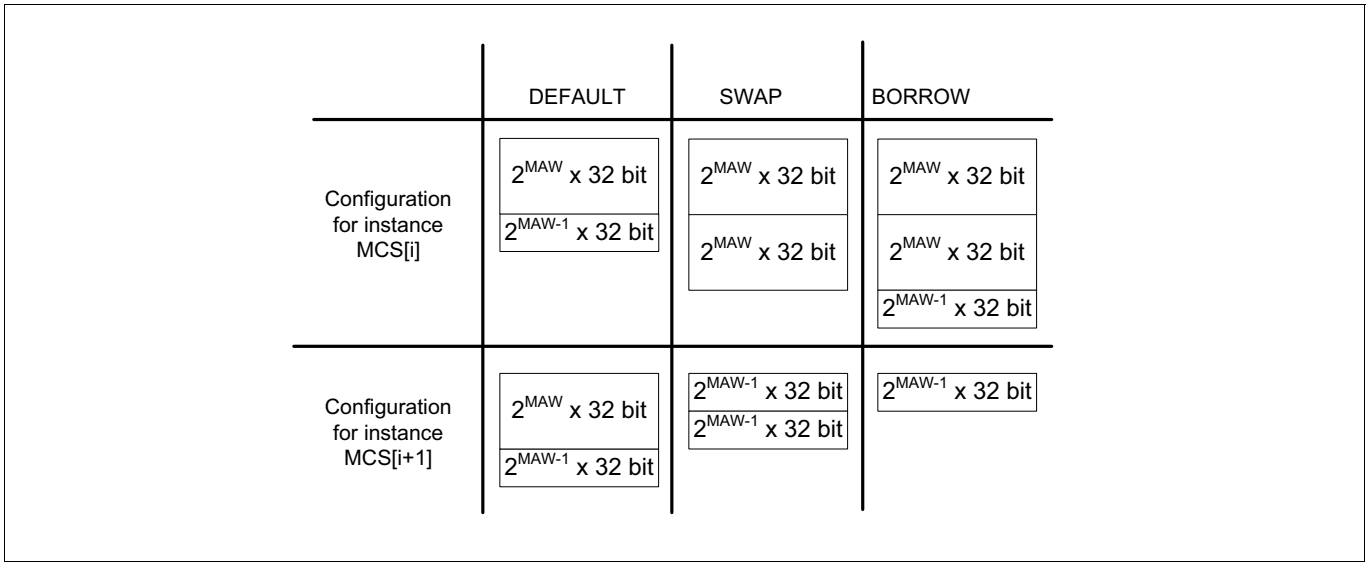


Figure 115 Memory Layout Configurations (ERM = 0)

Memory Layout Option of current MCS instance MCS[i]			Memory Layout Option of preceding MCS instance MCS[i-1]: DEFAULT	SWAP	BORROW
DEFAULT		MP0	2^{MAW+2}	2^{MAW+1}	0
		MP1	$2^{MAW+2} + 2^{MAW+1}$	2^{MAW+2}	2^{MAW+1}
SWAP		MP0	2^{MAW+2}	2^{MAW+1}	0
		MP1	2^{MAW+3}	$2^{MAW+2} + 2^{MAW+1}$	2^{MAW+2}
BORROW		MP0	2^{MAW+2}	2^{MAW+1}	0
		MP1	$2^{MAW+3} + 2^{MAW+1}$	2^{MAW+3}	$2^{MAW+2} + 2^{MAW+1}$

GTM_562_EP1

Figure 116 Memory Layout Parameters (ERM = 0)

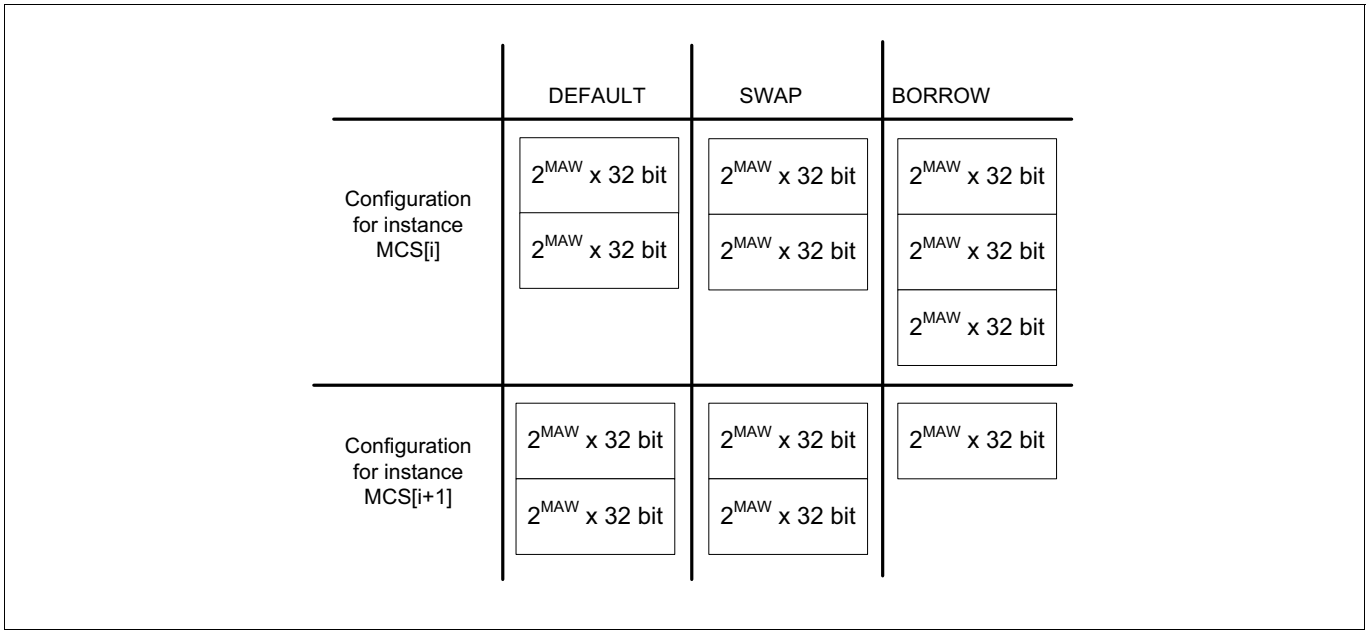


Figure 117 Memory Layout Configurations (ERM = 1)

Memory Layout Option of current MCS instance MCS[i]			Memory Layout Option of preceding MCS instance MCS[i-1]		
			DEFAULT	SWAP	BORROW
DEFAULT		MP0	2^{MAW+2}	2^{MAW+2}	0
		MP1	2^{MAW+3}	2^{MAW+3}	2^{MAW+2}
SWAP		MP0	2^{MAW+2}	2^{MAW+2}	0
		MP1	2^{MAW+3}	2^{MAW+3}	2^{MAW+2}
BORROW		MP0	2^{MAW+2}	2^{MAW+2}	0
		MP1	$2^{MAW+2} + 2^{MAW+3}$	$2^{MAW+2} + 2^{MAW+3}$	2^{MAW+3}

GTM_569_EP1

Figure 118 Memory Layout Parameters (ERM = 1)

28.18.2 MCFG Register Overview

Table 68 MCFG Configuration Registers Overview

Register Name	Description	see Page
MCFG_CTRL	MCFG Memory layout configuration.	430

28.18.3 MCFG Register Description

28.18.3.1 Register MCFG_CTRL

MCFG Memory Layout Configuration Register

It should be noted that the actual GTM implementation may embed less MCS instances than mentioned in this register (see product specific appendix). In this case this register only implements the register bits for available MCS instances.

This register is only writable if the bit **RF_PROT** in register **GTM_CTRL** is cleared.

MCFG_CTRL

MCFG Memory Layout Configuration Register (000F40$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												MEM9		MEM8	
r												rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEM7		MEM6		MEM5		MEM4		MEM3		MEM2		MEM1		MEM0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
MEMx (x=0-9)	2*x+1:2*x	rw	**Configure Memory pages for MCS-instance MCSx** 00_B DEFAULT configuration 01_B SWAP configuration 10_B BORROW configuration 11_B Reserved, do not use.
0	31:20	r	**Reserved** Read as zero, shall be written as zero.

28.19 TIM0 Input Mapping Module (MAP)

28.19.1 Overview

The MAP submodule generates the two input signals *TRIGGER* and *STATE* for the submodule DPLL by evaluating the output signals of the channel 0 up to channel 5 of submodule TIM0. By using the TIM as input submodule, the filtering of the input signals can be done inside the TIM channels themselves. The MAP submodule architecture is depicted in **Figure 119**.

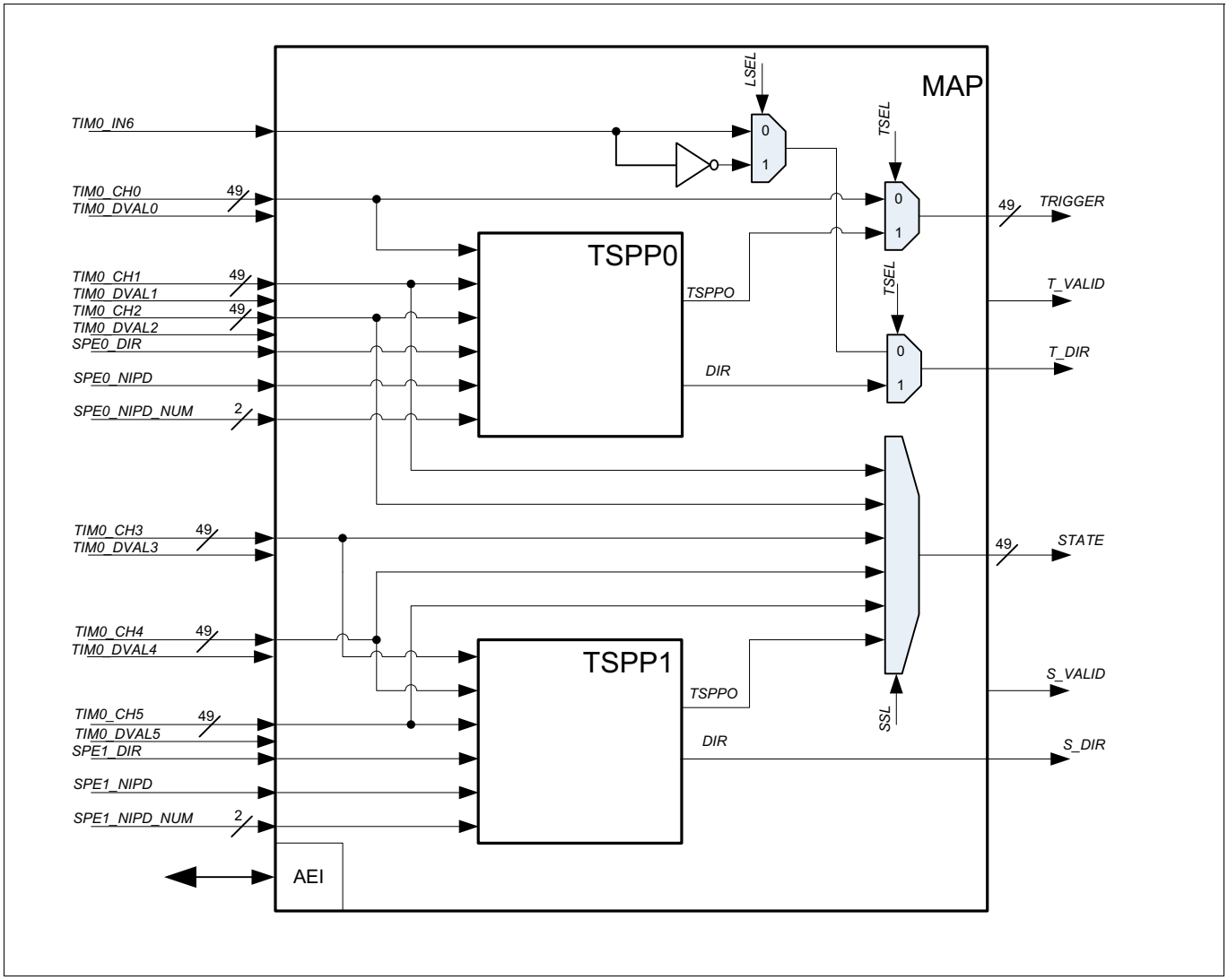


Figure 119 MAP Submodule architecture

Generally, the MAP submodule can route the channel signals coming from TIM0 in three ways. First, it is possible to route the whole 49 bits of data coming from channel 0 of module TIM0 (TIM0_CH0) to the *TRIGGER* signal which is then provided to the DPLL together with the *T_VALID* signal.

Second, the MAP module can route one of the five signals coming from the module TIM0 (i.e. the signals coming from channel 1 up to channel 5) to the output signal *STATE* which is then provided to the module DPLL together with the *S_VALID* signal.

Third, the *TRIGGER*, *T_VALID*, *STATE* and *S_VALID* signals can be generated out of the TIM Signal Preprocessing (TSPP) subunits. This is done in combination with the Sensor Pattern Evaluation (SPE) submodule described in chapter “Sensor Pattern Evaluation “.

There, the signal *TRIGGER* is generated in subunit TSPP0 out of the TIM0 signals coming from channel 0 up to 2.

The signal *STATE* is generated in subunit TSPP1 out of the TIM signals coming from channel 3 up to channel 5.

This is only be done, when the TSSPx subunits are enabled and when the *SPEx_NIPD* signal is raised by the SPE submodule. The *SPEx_NIPD_NUM* signal encodes, which of the 3 *TIMx_CHy* input signals has been changed. The *SPEx_DIR* signal is routed through the TSPPx subunit and implements the *T_DIR* or *S_DIR* signal.

A third method to provide a direction signal to DPLL is to use TIM0 channel 6 input (*TIM0_IN6*) and to route it instead of the *DIR* signal coming from TSSOP0 to the MAP output *T_DIR* (set TSEL=0)

28.19.2 TIM Signal Preprocessing (TSPP)

The TSPP combines the three 49 bit input streams coming from the TIM0 submodule and generates one combined 49 bit output stream *TSPPO*. The input stream combination is done in the unit Bit Stream Combination (BSC). The architecture of the TSPP is shown in **Figure 120**.

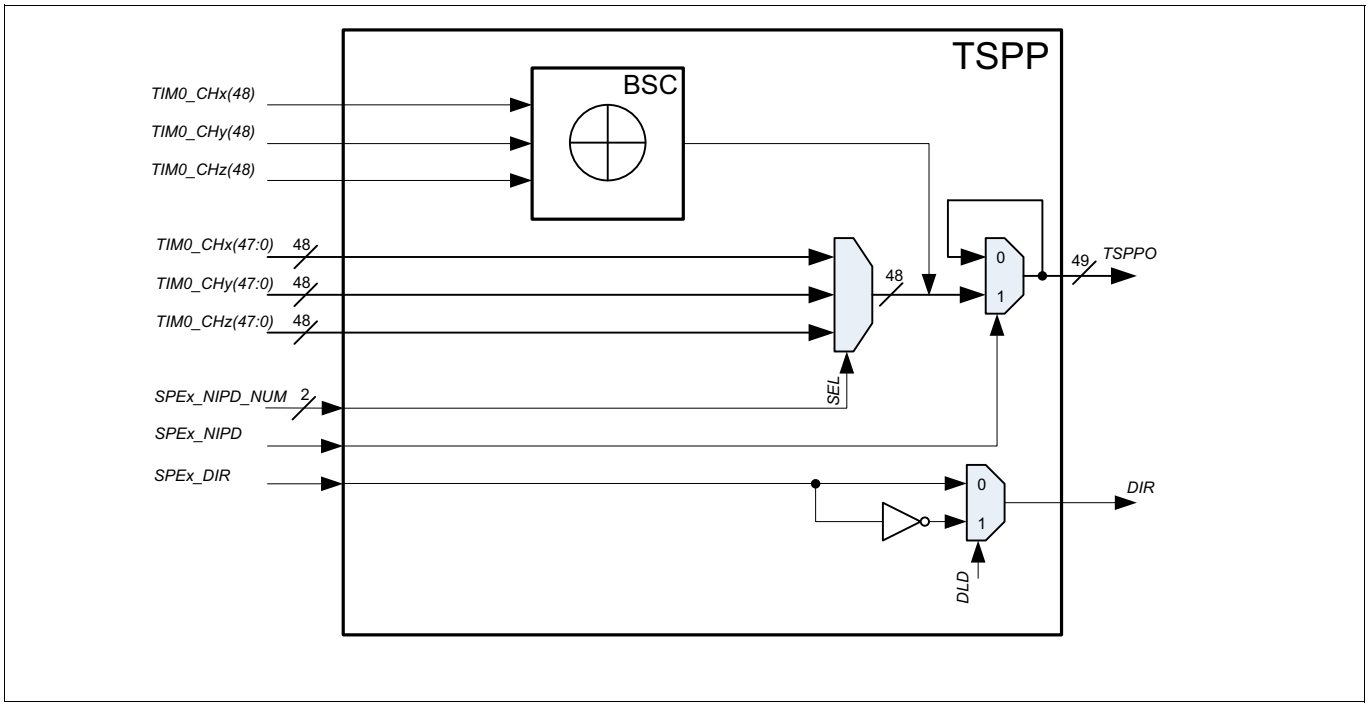


Figure 120 TIM Signal Preprocessing (TSPP) subunit architecture

28.19.2.1 Bit Stream Combination

The BSC subunit is used to XOR-combine the three most significant bits *TIM0_CHx(48)*, *TIM0_CHy(48)* and *TIM0_CHz(48)* of the TIM0 inputs. The XOR-combined signal is merged with the remaining 48 bits of one of the three input signals *TIM0_CHx(47…0)*, *TIM0_CHy(47…0)* or *TIM0_CHz(47…0)* the *TSPPO* signal. The selection is done with the *SPEx_NIPD_NUM* input signal coming from the SPE submodule. The action, when the 49 bits are transferred to the TSPPO and the T_VALID or S_VALID signal is raised is determined by the SPEx_NIPD signal coming from the SPE submodule. The *TSPPO* output signal generation is shown in the example in **Figure 121**.

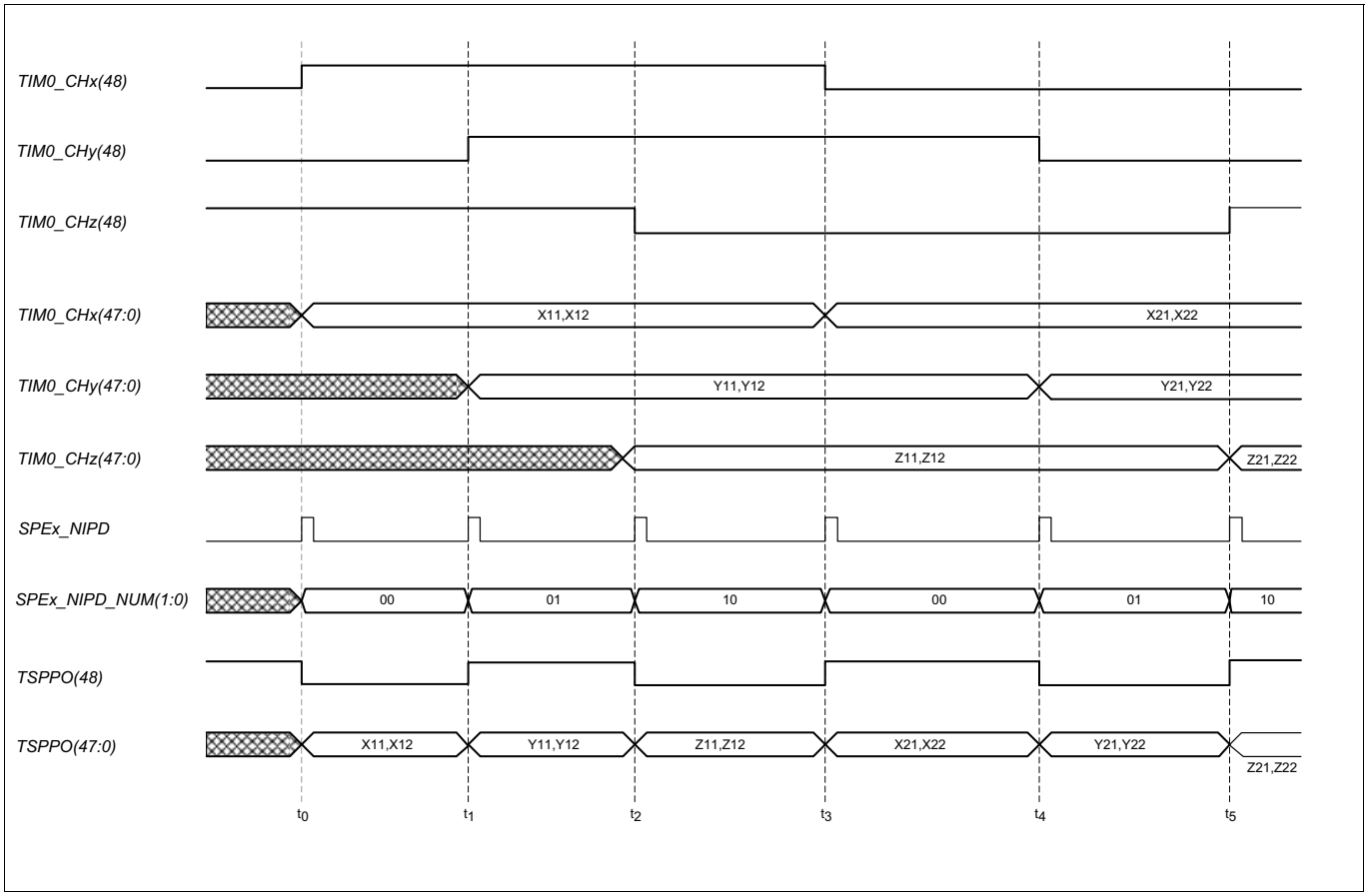


Figure 121 TSPP Signal generation for signal TSPPO

The *SPEx_NIPD_NUM* input signal determines, which data is routed to the *TSPPO* signal. At the first edge of *TIM0_CHx(48)* the new data X11 and X12 are routed to *TSPPO(47:0)*. The values X11 and X12 are the two 24 bit values coming from the TIM input channel TIM0_CHx. The next edge is at time t_1 on signal *TIM0_CHy(48)*. Therefore, at time t_1 the *TSPPO(48)* signal level changes and the *TSPPO(47:0)* is set to Y11 and Y12 and so forth.

28.19.3 MAP Register overview

Table 69 MAP Register overview

Register name	Description	see Page
MAP_CTRL	MAP Control register	434

28.19.4 MAP Register description

28.19.4.1 Register MAP_CTRL

MAP Control Register

MAP_CTRL
MAP Control Register **(000F00$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	TSPP1_I2V	TSPP1_I1V	TSPP1_I0V	0		TSPP1_DLD	TSPP1_EN	0	TSPP0_I2V	TSPP0_I1V	TSPP0_I0V	0		TSPP0_DLD	TSPP0_EN
r	rw	rw	rw	r		rw	rw	r	rw	rw	rw	r		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											LSEL	SSL			TSEL
r											rw	rw			rw

Field	Bits	Type	Description
TSEL	0	rw	**TRIGGER signal output select** 0_B TIM0_CH0 selected as TRIGGER output signal. TIM0_IN6 (TIM0 channel 6 input) is used as direction signal T_DIR. 1_B TSPP0_TSPPO selected as TRIGGER output signal
SSL	3:1	rw	**STATE signal output select** 000_B TIM0_CH1 selected as STATE output signal 001_B TIM0_CH2 selected as STATE output signal 010_B TIM0_CH3 selected as STATE output signal 011_B TIM0_CH4 selected as STATE output signal 100_B TIM0_CH5 selected as STATE output signal 101_B TSPP1_TSPPO selected as STATE output signal 110_B Same as 0b000 111_B Same as 0b000
LSEL	4	rw	**TIM0_IN6 input level selection** 0_B TIM0_IN6 input level '0' encodes TRIGGER in forward direction 1_B TIM0_IN6 input level '1' encodes TRIGGER in forward direction
TSPP0_EN	16	rw	**Enable of TSPP0 subunit** 0_B TSPP0 disabled 1_B TSPP0 enabled
TSPP0_DLD	17	rw	**DIR level definition bit** 0_B SPEx_DIR signal is routed through as is 1_B SPEx_DIR signal is inverted
TSPP0_I0V	20	rw	**Disable of TSPP0 TIM0_CHx(48) input line** 0_B Input line enabled 1_B Input line disabled; input for TSPP0 is set to zero (0)

Field	Bits	Type	Description
TSPP0_I1V	21	rw	**Disable of TSPP0 TIM0_CHy(48) input line** 0_B Input line enabled 1_B Input line disabled; input for TSPP0 is set to zero (0)
TSPP0_I2V	22	rw	**Disable of TSPP0 TIM0_CHz(48) input line** 0_B Input line enabled 1_B Input line disabled; input for TSPP0 is set to zero (0)
TSPP1_EN	24	rw	**Enable of TSPP1 subunit** 0_B TSPP1 disabled 1_B TSPP1 enabled
TSPP1_DLD	25	rw	**DIR level definition bit** 0_B SPEx_DIR signal is routed through as is 1_B SPEx_DIR signal is inverted
TSPP1_I0V	28	rw	**Disable of TSPP1 TIM0_CHx(48) input line** 0_B Input line enabled 1_B Input line disabled; input for TSPP1 is set to zero (0)
TSPP1_I1V	29	rw	**Disable of TSPP1 TIM0_CHy(48) input line** 0_B Input line enabled 1_B Input line disabled; input for TSPP1 is set to zero (0)
TSPP1_I2V	30	rw	**Disable of TSPP1 TIM0_CHz(48) input line** 0_B Input line enabled 1_B Input line disabled; input for TSPP1 is set to zero (0)
0	15:5, 19:18, 23, 27:26, 31	r	**Reserved** Read as zero, shall be written as zero.

28.20 Digital PLL Module (DPLL)

28.20.1 Overview

The digital PLL (DPLL) sub-module is used for frequency multiplication. The purpose of this module is to get a higher precision of position or value information also in the case of applications with rapidly changed input frequencies. There are two input signals *TRIGGER* and *STATE* for which periodic events are processed. The time period between two active events is called an increment. Each increment is divided into a given number of sub increments by pulses called SUB_INC. The resolution of the generated pulses is restricted by the period of the CMU_CLK0 clock or the TS_CLK respectively (see description of the modules TBU, CMU). The input signals *TRIGGER* and *STATE* can have the meaning of position information of linear or angle motions, mass flow values, temperature, pressure or level of liquids.By means of the DPLL the load of the CPU can be reduced essentially by relieving it from repeated or periodic standard tasks.

The DPLL has to perform the following tasks:

- prediction of the duration of the current increment in **Section 28.20.6**
- generation of SUB_INC1,2 pulses for up to 2 position counters in normal or emergency mode (see **Section 28.20.8.3**)
- synchronization of the actual position (under CPU control, see **Section 28.20.8.6.1**)
- possibility of seamless switch to emergency mode and back under CPU control, see configuration register DPLL_CTRL_0 at **Section 28.20.12**
- prediction of position and time related events in **Section 28.20.7**

28.20.2 Requirements and demarcation

The two input signals *TRIGGER* and *STATE* can be sensor signals from the same device or from two independent devices. When they come from the same device the *TRIGGER* input is typically a more frequent signal and *STATE* is a less frequent signal. In such a case the *STATE* signal can support an emergency mode, when no *TRIGGER* signal is available. There are also applications supported when *STATE* and *TRIGGER* are independent signals from different devices. Both input signals are combined with a validation signal *T_VALID* or *S_VALID* respectively, which shows the appearance of new data and must result in a data fetch and a start of the correspondent state machine to perform the calculations (see explanation below).

When *STATE* is a redundant signal of the same device only the *TRIGGER* input is used to generate the SUB_INC1 pulses in normal mode. There is a configuration possible, called emergency mode, for which the SUB_INC1 pulses are generated using the *STATE* input signal.

The decision to switch in the emergency mode and back is made outside the DPLL. The CPU must switch the configuration bit RMO (reference mode) in the DPLL_CTRL_0 register (see **Section 28.20.12**). Because a switch in emergency mode can appear suddenly, the information of the last increment duration of the *STATE* input up to FULL_SCALE should be stored always as a precaution.

The filtering as well as the combination or choice of the input signals is made in the TIM sub-module (see chapter “Timer Input Module (TIM)”) by use of a configurable filter algorithm for each slope and signal as well as in the MAP module (see chapter “TIM0 Input Mapping Module (MAP)”) the right *TRIGGER* or *STATE* signal is selected by a multiplexer or in the SPE module (see chapter “Sensor Pattern Evaluation (SPE)”) different signals are combined to a *TRIGGER* or *STATE* signal by using an antivalence operation.

The filter delay value of the signal is transmitted from the TIM module in the *FT* part of the corresponding signal, because the delay conditions of the signals can change during application.

The filter delays depend also on the filter algorithms used. Only the effective filter delay can be considered in the DPLL.

In order to provide the timing conditions to the DPLL the input trigger signals should have a time stamp (and optional in addition a filter value and a signal level value, as stated above) with an appropriate resolution. The resolution of the time stamps can be either the same resolution as the input time base TBU_TS0 (see **Figure 123**) or 8 times higher, selected by configuration bits in the DPLL_CTRL_1 register (see **Register DPLL_CTRL_1**). The time base TBU_TS0 is used to predict events in the future, called actions.

At the SUB_INCx outputs a predefined number of pulses between each active slope of the *TRIGGER/STATE* signal is generated, when the correspondent pulse generator is enabled by the enable bits SGEx=1 in the DPLL_CTRL_1 register (see **Register DPLL_CTRL_1**).

Dependent on configuration different strategies can be used to correct a wrong pulse number.

The FULL_SCALE range is divided into a fix number of nominal increments. Nominal increments do have the same size. The number of nominal increments in HALF_SCALE is specified in the DPLL_CTRL_0 register (see **Register DPLL_CTRL_0**).For synchronization purposes some *TRIGGER/STATE* input signals can be suppressed in dependency on the current position. Therefore an increment as duration between two active input events can be either a nominal increment or it can consist of more than one nominal increment. While a true nominal increment starts with an active event a virtual increment (of always nominal size) is an increment which starts with a missing event. Each increment which represents a gap (e.g. for synchronization purposes) consists of exactly one true nominal increment and at least one virtual increment, each of them having the same nominal duration (see figure below).

28.20.3 Input signal courses

Typical input signal courses are shown in the figure below.

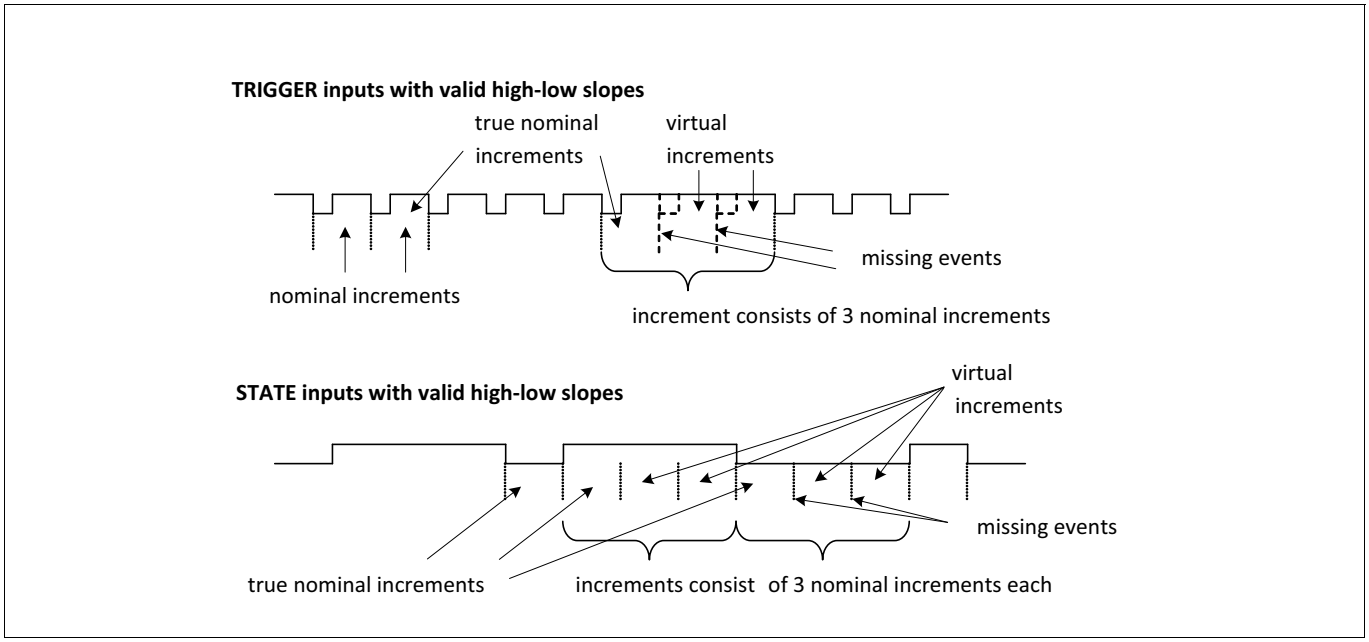


Figure 122 Trigger and State Input Signal

28.20.4 Block and interface description

The block description of the DPLL is shown in the following figure.

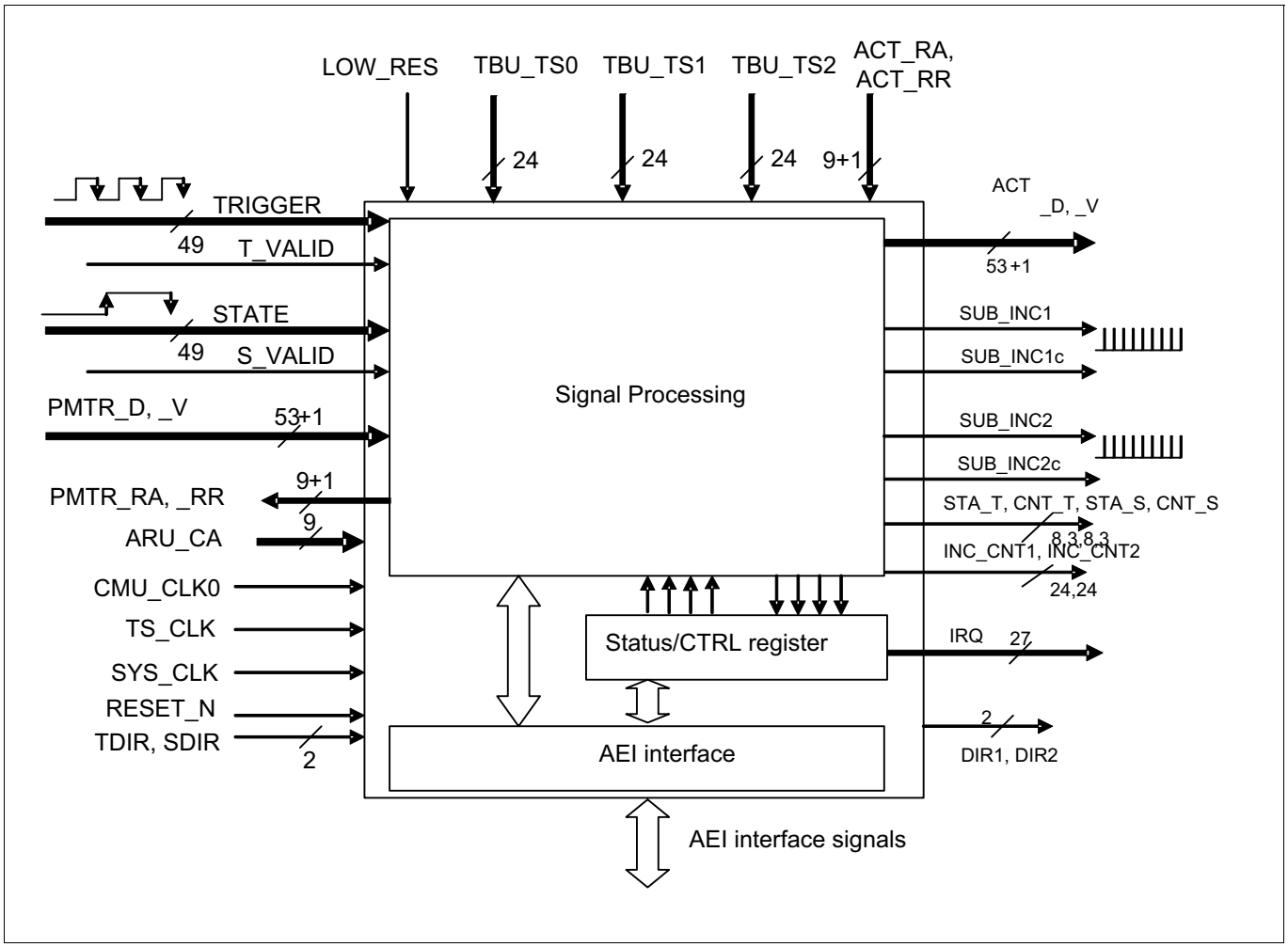


Figure 123 DPLL Block Diagram

Section 28.20.4.1 summarizes the interface signals of the DPLL shown by the block diagram above.

28.20.4.1 Interface description of DPLL

Table 70 Interface description of DPLL

Name	Width	I/O	Description	Comment
TRIGGER	49	I	Normal Signal for triggering DPLL by positions/values Bit(48)= TRIGGER_S Bits(47:24)= TRIGGER_FT Bits(23:0)= TRIGGER_TS	One bit signal value (SV), 24 bits filter delay value info and 24 bits time stamp, filtered in different modes.
STATE	49	I	Assistance signal for synchronization STATE(48)= STATE_S STATE(47:24)= STATE_FT STATE(23:0)= STATE_TS	Replacement of signal *TRIGGER* for emergency situations, or signal from an independent device; bits like above, corresponding
T_VALID	1	I	The values of *TRIGGER* are valid	Announces the arrival of a new *TRIGGER* value
S_VALID	1	I	The values of STATE are valid	Announces the arrival of a new STATE value
PMTR_D	53	I	Position minus time request data, delivered by ARU on request for up to 24 requests PMTR_RR; SV_i=PMTR_D(52:48): ACB bits, directly written to the correspondent DPLL_ACB_j registers PSA[i]=PMTR_D(47:24): position value for action DLA[i]=PMTR_D(23:0) time delay value for action	Data values for calculation of actual Actions; the values are requested by AENi=1[1)] and CAIP=0[2)]; a served request is shown by PMTR_V which signals that valid PMTR data arrived and they are written immediately after that to the corresponding RAM regions and registers; The DLA[i] values must have the same resolution as the TBU_TS0 input.
PMTR_V	1	I	signals a valid PMTR_D value, that means data is delivered on request	when valid: PMTR_D overwrites data in the PSA[i] and DLA[i] registers, also when the corresponding ACT_N[i][3)] bit =1;
ARU_CA	9	I	Channel address; for valid PMTR addresses: demand data by setting PMTR_RR=1 when enabled by AENi=1[1)] and CAIP=0[2)];	counter value of ARU selects PMTR_RA and PMTR_RR when a valid address
PMTR_RA	9	O	read address of PMTR access	reflects ID_PMTR_i according to the selected channel address
PMTR_RR	1	O	read request of PMTR access; suppressed for CAIPi=1 (see DPLL_STATUS register)	reflects the value of the corresponding AENi[1)] bit while the correspondent bit CAIPi=0[2)]

Table 70 Interface description of DPLL (cont'd)

Name	Width	I/O	Description	Comment
ACT_D	53	O	Output of a time stamp, a position and a control signal for a calculated action; SV_i=ACT_D(52:48): ACB bits, directly written from the correspondent PMTR_D signals; ACT_D(47:24) is the calculated position value PSAC[i] for the action in relation to TBU_TS1 or 2[4] and ACT_D(23:0) is the time stamp value TSAC[i] for the action in relation to TBU_TS0 [4]	Future time stamp with the resolution as TBU_TS0 input, additional position information and additional control bits;
ACT_V	1	O	ACT_D value is available and valid; blocking read access	for a valid action address: ACT_V reflects the shadow value of ACT_N[i][3] (ACT_N[i] is 1 when new PMTR values are available and the shadow register is updated, when a calculation of the actual PMTR values was done); reset after reading of the ACT_D values
ACT_RA	9	I	ACTION read address;	address bits for selection of all 24 action channels
ACT_RR	1	I	read request of selected action	the action data is demanded from another module
IRQ	27	O	Interrupt request output	Interrupts of DPLL;
SUB_INC1	1	O	Pulse output for *TRIGGER* input filter	sub-position increment provided continuously
SUB_INC2	1	O	Pulse output for *STATE* input filter (when *TRIGGER* and *STATE* are used for 2 independent devices)	sub-position increment provided continuously
SUB_INC1c	1	O	Pulse output for time base unit 1 in compensation mode (can stop in automatic end mode)	sub-position increment related to *TRIGGER* input
SUB_INC2c	1	O	Pulse output for time base unit 2 in compensation mode (can stop in automatic end mode)	sub-position increment related to *STATE* input (when *TRIGGER* and *STATE* are used for 2 independent devices)
TS_CLK	1	I	Time stamp clock	used for generation of the time stamps; this clock is used to generate the SUB_INC1,2 pulses
CMU_CLK0	1	I	CMU clock 0	used for rapid pulse correction of SUB_INC1,2
SYS_CLK	1	I	System clock	High frequency clock

Table 70 Interface description of DPLL (cont'd)

Name	Width	I/O	Description	Comment
RESET_N	1	I	Asynchronous reset signal	Low active; After Reset he DPLL is available only after performing the RAM reset procedures by the DPLL hardware.
LOW_RES	1	I	low resolution of TBU_TS0 selected; shows which of the 27 bits of TBU_TS0 are connected to the DPLL	LOW_RES=0: T BU_TS0(DPLL)= lower 24 Bits of TBU_TS0(TBU); LOW_RES=1: TBU_TS0(DPLL)= higher 24 Bits of TBU_TS0(TBU); In the case LOW_RES=1 the TS0_HRT and/or TS0_HRS bits can be set [5]
TBU_TS0	24	I	Actual time stamp from TBU; is needed to decide, if a calculated action is already in the past	24 bit time input, with a resolution of the time stamp clock
TBU_TS1	24	I	Actual position/value stamp 1; for calculation of position stamps (*TRIGGER*/*STATE*)	24 bit pos./val. input, with a resolution of the SUB_INC1 pulses
TBU_TS2	24	I	Actual position/value stamp 2; to be implemented for an additional independent position	ditto for SUB_INC2 for calculation of position stamps (*STATE*) for SMC[6]=RMO[5]=1
TDIR	1	I	Direction of *TRIGGER* input values (TDIR=0 does mean a forward direction and TDIR=1 a backward direction)	direction information from multiple sensors valid only for SMC[6]=1 or IDDS=1
SDIR	1	I	Direction of *STATE* input values (SDIR=0 does mean a forward direction and SDIR=1 a backward direction)	direction information from multiple sensors valid only for SMC[6]=1
DIR1	1	O	Direction information of SUB_INC1 (count forwards for DIR1=0 and backwards for DIR1=1)	count direction of TBU_CH1_BASE; DIR1 changes always after the evaluation of the corresponding valid *TRIGGER* slope and after incrementing/decrementing of the address pointer
DIR2	1	O	Direction information of SUB_INC2 (count forwards for DIR2=0 and backwards for DIR2=1)	count direction of TBU_CH2_BASE; DIR2 changes always after the evaluation of the corresponding valid *STATE* slope and after incrementing/decrementing of the address pointer
STA_T	8	O	Status of TRIGGER state machine	Output to MCS0. Signals accessible via µC interface as well (DPLL_STA)

Table 70 Interface description of DPLL (cont'd)

Name	Width	I/O	Description	Comment
CNT_T	3	O	Count TRIGGER	Output to MCS0.This reflects the count of active *TRIGGER* slopes (mod8).Signals accessible via µC interface as well (DPLL_STA)
STA_S	8	O	Status of STATE state machine	Output to MCS0.Signals accessible via µC interface as well (DPLL_STA)
CNT_S	3	O	Count STATE	Output to MCS0.This reflects the count of active *STATE* slopes (mod8).Signals accessible via µC interface as well (DPLL_STA)
INC_CNT1	24	O	Increment counter of pulse generator 1 (automatic end mode)	Output to MCS0.Signals accessible via µC interface as well (DPLL_INC_CNT1)
INC_CNT2	24	O	Increment counter of pulse generator 2 (automatic end mode)	Output to MCS0.Signals accessible via µC interface as well (DPLL_INC_CNT2)

1) see DPLL_CTRL_x register, x=2,3,4; see **Section 28.20.12.3**, **Section 28.20.12.4**, **Section 28.20.12.5**
2) see DPLL_STATUS register; see **Section 28.20.12.30**
3) see DPLL_ACT_STA register; see **Section 28.20.12.7**
4) see DPLL input signal description; see **Section 28.20.1**
5) see DPLL_CTRL_0 register; see **Section 28.20.12.1**
6) see DPLL_CTRL_1 register; see **Section 28.20.12.2**

For references above the following hints are used: [1)] see DPLL_CTRL_x register, x=2,3,4; see **Register DPLL_CTRL_2**,**Register DPLL_CTRL_3**,**Register DPLL_CTRL_4**[2)] see DPLL_STATUS register; see **Register DPLL_STATUS**[3)] see DPLL_ACT_STA register; see **Register DPLL_ACT_STA**[4)] see DPLL_CTRL_0 register; see **Register DPLL_CTRL_0**[5)] see DPLL_CTRL_1 register; see **Register DPLL_CTRL_1**[6)] see DPLL input signal description; see **Section 28.20.1**

28.20.5 DPLL Architecture

28.20.5.1 Purpose of the module

The DPLL generates a predefined number of incremental signal pulses within the period between two events of an input *TRIGGER* or *STATE* signal, when the corresponding pulse generator is enabled. The resolution of the pulses is restricted by the frequency of the time stamp clock (TS_CLK). Changes in the period length of the predicted time period of the current increment will result in a change of the pulse frequency in order to get the same number of pulses. This adoption can be performed by DPLL hardware, software or with support of DPLL hardware in different modes.

The basic part of a DPLL is to make a prediction of the current period between two *TRIGGER* and/or *STATE* signal edges. Disturbances and systematic failures must be considered as well as changes of increment duration caused by acceleration and deceleration of the supervised process. Therefore, a good estimation is to be done using some measuring values from the past. When the process to be predicted takes a steady and differentiable course not only the current increment but also some more increments for the future can be predicted. In utilization of such calculations actions for the future can be predicted.

28.20.5.2 Explanation of the prediction methodology

As already shown in **Section 28.20.1** the DPLL has to perform different tasks. The basic function for all these tasks is the prediction of the current increment which is based on a relation between increments in the past. Because the relation between two succeeding intervals at a fixed position remains also valid in the case of acceleration or deceleration the prediction of the duration of the current time interval is done by a similarity transformation. Having a good estimation of the current time interval, all the other tasks can be done easily by calculations explained in **Section 28.20.6**.

28.20.5.3 Clock topology

All registers are read using the system clock *SYS_CLK*. The SUB_INC1,2 pulses generated have in the normal case the highest frequency not higher then *CMU_CLK0* or the half of *TS_CLK* respectively. For individual pulses the frequency can be doubled. All operations can be performed using the system clock.

28.20.5.4 Clock generation

The clock is generated outside the DPLL.

28.20.5.5 Typical frequencies

For the system clock a reasonable clock frequency should be applied to give the DPLL module sufficient computational power to calculate all needed values (prediction of next increment, actions) in time. The typical system clock frequency is in the range from 40 MHz up to 150 MHz.

28.20.5.6 Time stamps and systematic corrections

The time stamps for the input signals *TRIGGER* and *STATE* have 24 bits each. These bits represent the value of the 24 bit free running counter running with a clock frequency selected by the configuration of the TBU. Using a typical frequency of 20 MHz the time stamp represents a relative value of time with a resolution of 50 ns.

The input signals have to be filtered. The filter is not part of the DPLL. The time stamps can have a delay caused by the filter algorithm used. There are delayed and undelayed filter algorithms available and the delay value can depend on a time or a position value.

Systematic deviations of *TRIGGER* inputs can be corrected by a profile, which also considers systematic missing *TRIGGER*s. The increments containing missing *TRIGGER*S are divided into the corresponding number of nominal increments whereas the duration of a nominal increment is the greatest divider of all increments duration.

For each increment this number of enclosed nominal increments is stored in a profile as NT value for *TRIGGER*. When the increment is a nominal increment the NT value is 1.

For the *TRIGGER* input the value NT is stored in the ADT_T field in RAM region 2c.

In the case of **AMT**[4] = 1 the **ADT_T[i]** values in the RAM region 2c must also contain the adapting information for the *TRIGGER* signal, which considers for each increment a systematic physical deviation **PD** from the perfect increment value with a resolution according to the chosen value of MLT+1, which describes the number of SUB_INC1 pulses for a nominal increment.

The value **PD** for the *TRIGGER* describes the amount of missing or surplus pulses with a sint13 value, to be added to MLT+1 directly. The correction value is in this way also applicable in the case of missing *TRIGGER* inputs for the synchronization gaps. In this case the amount of provided SUB_INC1 pulses for a nominal increment (MLT+1) is multiplied (MLT+1) + PD is multiplied by NT.

The NT value of the current increment is stored in the variable SYN_T (see **NUTC** register in **Section 28.20.12.14**).

In the case of **RMO**[4]= 1 for **SMC**[5]=0 (emergency mode) the time stamp of *STATE* is used to generate the output signal SUB_INC1.

More inaccuracy should be accepted in emergency mode because usually there are only fewer events available for FULL_SCALE according to the value SNU[5].

For the *STATE* signal the systematic deviations of the increments can be corrected in the same way as for *TRIGGER* by profile and adaptation information as described below.

Systematic deviations of *STATE* inputs can be corrected by a profile, which also considers systematic missing *STATE* events. The increments containing missing *STATEs* are divided into the corresponding number of nominal increments whereas the duration of a nominal increment is the greatest divider of all increments duration.

For each increment this number of enclosed nominal increments is stored in a profile as NS value for *STATE*. When the increment is a nominal increment the NS value is 1.

For the *STATE* input the value NS is stored in the ADT_S field in RAM region 1c3.

In the case of **AMS**[4] = 1 the **ADT_S[i]** values in the RAM region 1c3 must contain the adapting information for the STATE signal, which considers for each increment a systematic physical deviation **PD_S** from the perfect increment value with a resolution according to the chosen value of MLS1, which describes the number of SUB_INC1 pulses for a nominal increment (see below).

The number of pulses SUB_INC1 for a nominal *STATE* increment in emergency mode (for SMC=0) is given by the value of MLS1= (MLT + 1)* (TNU + 1) /(SNU + 1) in order to get the same number of pulses in FULL_SCALE for normal and emergency mode. This value has to be configured by the CPU.

The value **PD_S** for the *STATE* describes the amount of missing or surplus pulses with a sint16 value, to be added to MLS1 directly. The correction value is in this way also applicable in the case of missing *STATE* inputs for the synchronization gaps. In this case the amount of provided SUB_INC1 pulses for a nominal increment MLS1 is multiplied by NS first before the PD_S value is added.

The current NS value is stored in the variable SYN_S (see **NUSC** register in **Section 28.20.12.15**).

28.20.5.7 DPLL Architecture overview

As shown in **Figure 123** the DPLL can process different input signals. The signal *TRIGGER* is the normal input signal which gives the detailed information of the supervised process. It can be for instance the information of water or other liquid level representing the volume of the liquid, where each millimeter increasing results in a *TRIGGER* signal generation. In order to get a predefined filling level, without overflow also the inertia of the system must be taken into account. Hence, some delay for closing the inlet valve and also the remaining water amount in the pipe must be considered in order to start the closing action earlier as the filling level will be reached.

A second input signal *STATE* sends an additional (redundant) information for instance at some centimeters and because of intervals with different distances it gives also information about the system state with the direction of the water flow (in or out), while the *TRIGGER* signal must not contain information concerning the flow direction. In some applications the inactive slope of *TRIGGER* can be utilized to transmit a direction information. In the case of faults in the *TRIGGER* signal the *STATE* signal is to be processed in order to reach the desired value nevertheless, maybe with some loss of accuracy.

The measuring scale can have some systematic failures, because not all millimeter or centimeter distances measured mean the same value. This could be due to changes in the thickness of the measuring cylinder or the inaccurate position of the marks. These systematic failures are well known by the system and for improvement of the prediction the signals *ADT_T* and *ADT_S* for the correction of the systematic failures of *TRIGGER* and *STATE* respectively are stored in the internal RAM.

The input signals *TRIGGER* and *STATE* are represented as a time stamp signal each, which is stored in the 24 bit TS-part of the corresponding signal.

Information concerning the delay of this signal by filtering of disturbances is stored in the 24 bit FT-part of the signal.

In order to establish the relation of time stamps to the actual time the TBU_TS0[6] value is also provided showing the actual time value used for prediction of actions in the future.

After reaching the desired water level the water is filled in a bottle by draining. After that the water filling is repeated. The water level at draining is observed by the same sensor signals (the same number of *TRIGGER* pulses), but the duration of the draining could be different from the filling time. Both times together form the FULL_SCALE region, while one of them is a HALF_SCALE region, which can differ in time but not in the number of pulses, especially for *TRIGGER*.

For synchronization purposes some *TRIGGER* marks can be omitted in order to set the system to a proper synchronization value (maybe before the upper filling value is reached).

In emergency situations, when the *TRIGGER* signals are missed the *STATE* signal is used instead of.

The PMTR_i [4] signals announce the request for a position minus time calculation for up to 24 events.

All 24 events can be activated using the 24 AENi[1] (action enable) bits. Each of these enable bits are asked by the routing engine for a read access. The corresponding read request is generated by the AENi bit while CAIPx is zero. CAIP1 and CAIP2 are two bits of the DPLL_STATUS register for 12 actions each with the meaning “calculation of actions in progress”, controlled by the state machine (see **Section 28.20.2**) for scheduling the operations.

When such a request is serviced by the ARU (in the case CAIPx=0) the values for position and time are written in the corresponding RAM 1a region (0x0200… 0x025C for the position value and 0x0260… 0x02BC for the delay value), the control bits for the corresponding action are set accordingly. When a new PMTR value arrives, an old value is overwritten without notice and the shadow bit of ACT_N[i] is cleared while the ACT_N[i] (new action) bit in the DPLL_ACT_STA register is set. The ACT_N[i] is cleared, when the currently calculated action value is in the past. Overwriting of old information is possible without data inconsistency because the read request to ARU is suppressed during action calculations by the CAIP1,2 bits. In this way always the last possible PMTR value is used consistently.

28.20.5.8 DPLL Architecture description

The DPLL block diagram of **Figure 123** will now be explained in detail in combination with some example configurations of the control registers. There are different configuration bits available which can adopt the DPLL to the use case (see **Section 28.20.12**).

Let for example in HALF_SCALE the *TRIGGER* number TNU[4] be 0x3B (which is for TNU+1 = 60 decimal that does mean 120 events in FULL_SCALE) and the number of SUB_INC1 pulses between two *TRIGGER*s MLT[4] be 0x257 (this means 600 pulses per *TRIGGER* event). Than the FULL_SCALE region can be divided into 72000 parts each of them associated with its own SUB_INC1 pulse. For a run through FULL_SCALE all 72000 pulses should appear but maybe with a different pulse frequency between two *TRIGGER* events. For this example after each 600 pulses at the *SUB_INC1* output the next *TRIGGER* event is to be expected with the corresponding new time stamp.

Missing SUB_INC1 pulses due to acceleration have to be taken into account within the next increment. Not one pulse has to be missed or added because of calculation inaccuracy in average for a sufficient number of FULL_SCALE periods. This means that not one pulse is sent in addition and all missing pulses are to be caught up on afterwards.

For the systematic arrangement of *TRIGGER* inputs **the profile** (as already mentioned in **Section 28.20.5.6** is stored in the RAM region 2c (see **Section 28.20.14.3**). In this field the relative position of gaps can be stored in the NT value and also physical deviations in the PD value.

For the consideration of systematic missing *TRIGGER*s the actual NT value of the profile is stored in the SYN_T bits of the NUTC register (see **Section 28.20.12.14**).

In normal mode the physical deviation values PD in the ADT_T field could be used to balance the local systematic inaccuracy of the *TRIGGER* signal. The value of PD (see **Section 28.20.14.3**) is the pulse difference in the corresponding increment and does mean the number of sub pulses to be added to the nominal number of pulses. PD is a signed integer value using 13 bits: up to +/-4096 pulses can be added for each increment.

The NT value of the profile ADT_T has the value 1, when a nominal increment is assumed. An integer number greater than 1 shows the number of nominal increments to be considered for a gap. For the actual increment after synchronization the corresponding NT value is stored in SYN_T of the NUTC register.

Using the *STATE* input there are similar configuration bits available (see **Section 28.20.12**).

Let for example in HALF_SCALE the *STATE* number SNU[5] be 0xB (which is for SNU+1 =12 decimal and while SYSF[5] = 0 that does mean 24 events in FULL_SCALE). In order to get the same number of SUB_INC1 pulses for FULL_SCALE as above for *TRIGGER*s the value (MLT+1)=600 is divided by 2*(SNU+1)=24 and multiplied with 2*(TNU+1)=120. The result 3000 must be stored in MLS1 by the CPU (see **Section 28.20.12.75**).

For the systematic arrangement of *STATE* inputs **the profile** (as already mentioned in **Section 28.20.5.6** is stored in the RAM region 1c3 (see **Section 28.20.12.90**). In this field the relative position of gaps can be stored in the NS value and also physical deviations in the PD_S value.

For the consideration of systematic missing *TRIGGER*s the actual NS value of the profile is stored in the SYN_S bits of the NUSC register (see **Register DPLL_NUSC**).

In emergency mode the physical deviation values PD_S in the ADT_S field could be used to balance the local systematic inaccuracy of the *STATE* signal. The value of PD_S (see **Section 28.20.12.90**) is the pulse difference in the corresponding increment and does mean the number of sub pulses to be added to the nominal number of pulses per increment. PD_S is a signed integer value using 16 bits: up to +/-32768 pulses can be added for each increment.

In emergency mode the physical deviation values PD_S in the ADT_S field could be used to balance the local systematic inaccuracy of the STATE signal. The value of PD_S (see **Section 28.20.12.90**) is the pulse difference in the corresponding nominal increment and does mean the number of sub pulses to be added to the nominal number of pulses. . PD_S is a signed integer value using 16 bits: up to +/-32768 pulses can be added for each increment.

The NS value of the profile ADT_S has the value 1, when a nominal increment is assumed. An integer number greater than 1 shows the number of nominal increments to be considered for a gap. For the actual increment after synchronization the corresponding NS value is stored in SYN_S of the NUSC register.

28.20.5.9 Block diagrams of time stamp processing.

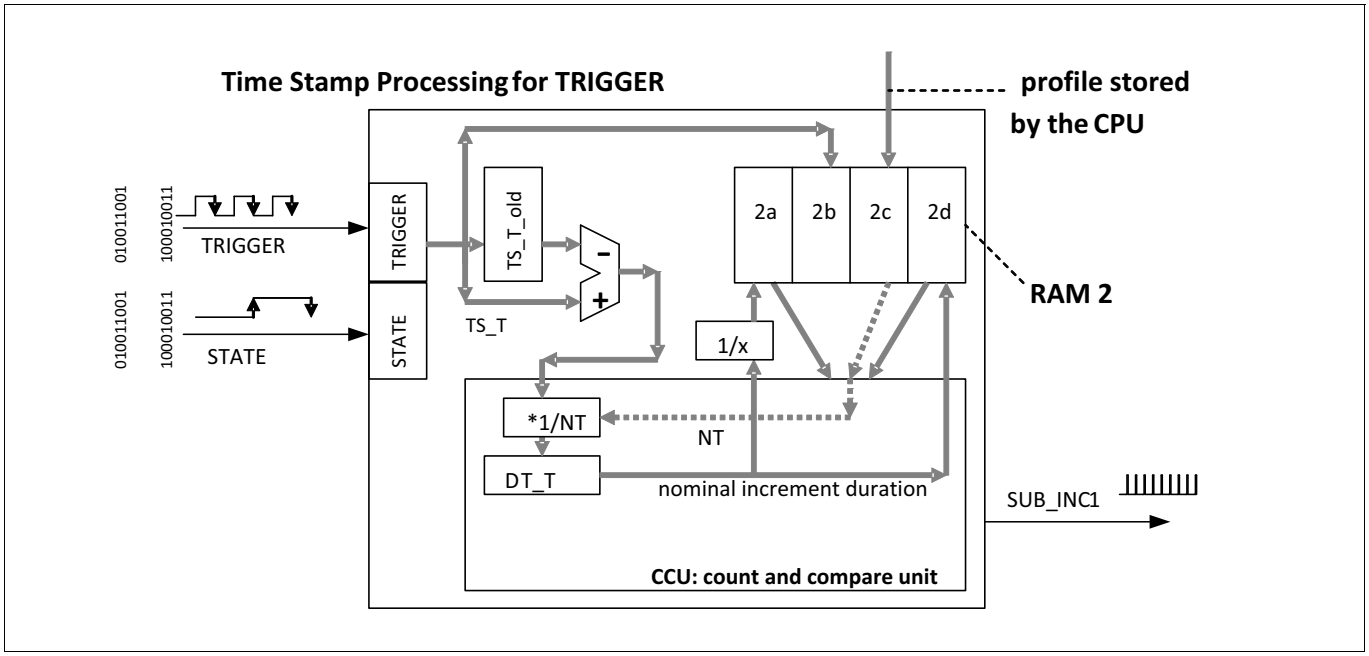


Figure 124 Time Stamp Processing Trigger

As shown in the block diagram above the time stamp difference of two succeeding input events is calculated. For the prediction of the current increment duration such values from the past are used. For this purpose the measured and calculated values of the last FULL_SCALE period are stored in the RAM. For the *TRIGGER* input there are 4 different RAM parts in the RAM region 2:

- 2a stores the reciprocals of each nominal increment duration RDT_T
- 2b stores the time stamps of each valid input event TSF_T
- 2c is used for the profile ADT_T and
- 2d for the nominal increment durations DT_T.

Because the prediction is based on the relations of increments in the past this relation can be calculated easily by the multiplication of increment duration values with the reciprocal value of another increment. In order not to be forced to distinguish between gaps and "normal" increments duration also for gaps only the nominal duration and the correspondent reciprocal values are stored in the RAM field. This is possible by consideration of the NT value in the profile: the measured increment duration is divided by NT.

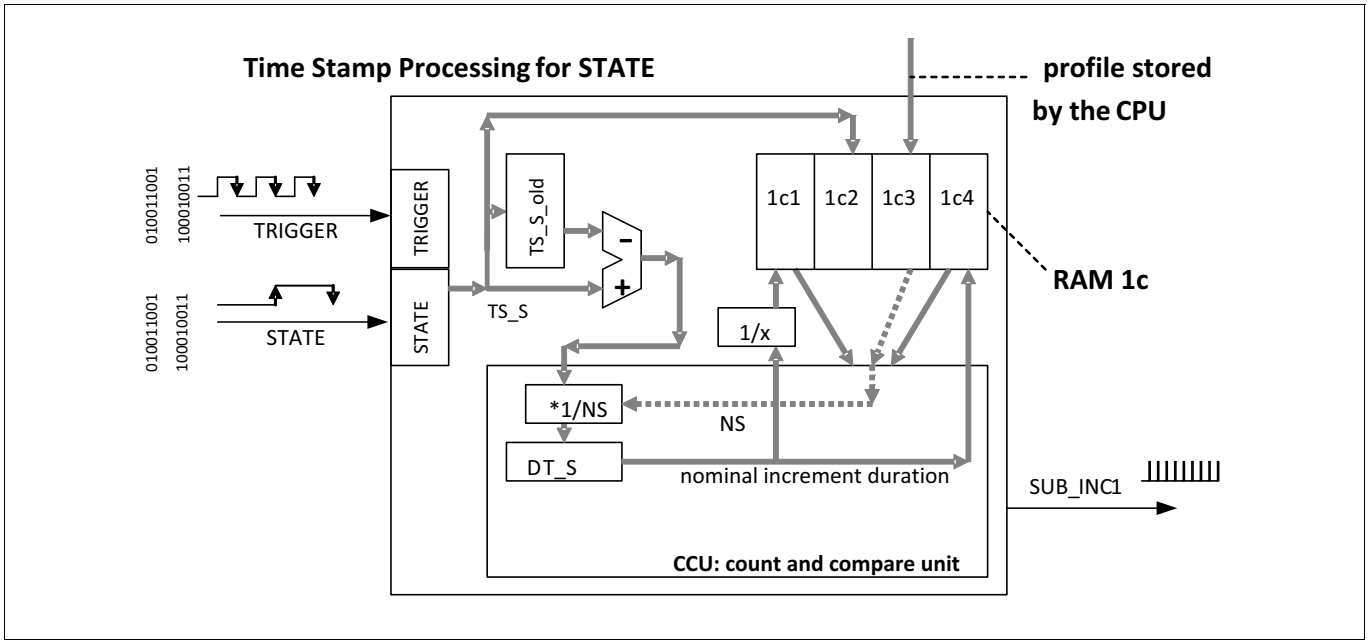


Figure 125 Time Stamp Processing State

For the *STATE* input there are also 4 different RAM parts in the RAM region 1c:

1c1 stores the reciprocals of each nominal increment duration RDT_S

1c2 stores the time stamps of each valid input event TSF_S

1c3 is used for the profile ADT_S and

1c4 for the nominal increment durations DT_S.

The calculations are performed similar as for the *TRIGGER* input. The NS value in the profile shows the appearance of a gap.

28.20.5.10Register and RAM address overview

The address map of the DPLL is divided into register and memory regions as defined in **Table 71**. The addresses from 0x0000 to 0x00FC are reserved for registers, from 0x0100 to 0x01FC is reserved for action registers to serve the ARU at immediately read request.

The RAM is divided into 3 independent accessible parts 1a, 1b+c and 2.

The part 1a from 0x0200 to 0x037C is used for PMTR values got from ARU and intermediate calculation values; there is no write access from the CPU possible, while the DPLL is enabled.

The RAM 1b part from 0x0400 to 0x05FC is reserved for RAM variables and the RAM part 1c from 0x0600 to 0x09FC is used for the *STATE* signal values.

The RAM region 2 from 0x4000 to 0x7FFC is reserved for the *TRIGGER* signal values. RAM region 1a has a size of 288 bytes, Ram 1b+c uses 1,125 Kbytes while RAM region 2 is configurable from 1,5 to 12 Kbytes, depending on the number of *TRIGGER* events in FULL_SCALE. The AOSV_2 register is used to determine the beginning of each part.

Table 71 gives the DPLL Address map overview.

Registers are used to control the DPLL and to show its status. Also parameters are stored in registers when useful. The table below shows the addresses for status and control registers as well as values stored in additional registers. The register meaning explained in the register overview (**Section 28.20.11**) while the bit positions of the status and control registers are described in detail in **Section 28.20.12**.

Time stamps for TRIGGER and *STATE* can have either the same resolution as the TBU_TS0 input or 8 times higher. This is configured in the DPLL_CTRL_1 register (see **Register DPLL_CTRL_1**). While the TBU_TS0 is used for

action predictions the higher resolution of *TRIGGER* and *STATE* inputs can be used for a more accurate pulse generation.

The time stamp fields of *TRIGGER* and *STATE* are stored in the corresponding RAM regions in such a way, that for a gap also entries for the virtual increments are provided. This is due to the necessity to calculate time differences between a given number of (real and virtual) input events independent of a gap. Therefore the gap is extended in the RAM fields 2b and 1c2.For all other RAM regions in RAM 2 and RAM 1c the gap is considered as one increment.

For the access to the RAM fields there must be address pointers. When the device starts all address pointers have a zero value and the first measured and calculated values are stored in the beginning of the corresponding RAM field.Because the position of the device is usually unknown at the beginning no profile information can be used. The profile regions must have their own address pointers each which are set by the CPU as soon as the position is known. By setting the appropriate value to the address pointer APT_2C of the *TRIGGER* profile or APS_1C3 of the *STATE* profile respectively the synchronization bits in the DPLL_STATUS register SYT or SYS are set respectively. In the following the gap information can be used.

Because the time stamp fields are extended at the gaps there must be additional address pointers for these regions: APT_2B for *TRIGGER* time stamps and APS_1C2 for *STATE* time stamps. These address pointers must be incremented by NT or NS respectively when a gap appears.

Table 71 Register and RAM address map

Addr. rangeStart	Addr. rangeEnd	Value number	Byte #	Content	Indication	Region	RAM size
0x0000	0x0FC	64	256	Register	used/reserved	0	no RAM
0x100	0x1FC	64	192	ACTION registers	direct read from ARU	0	no RAM
0x0200	0x03FC	128	384	PMTR values RAM 1a	CPU R/Pw access, when DPLL disabled; ARU has highest priority	1a with own ports	RAM part 1a:384 bytes
0x0400	0x05FC	128	384	Variables RAM 1b	R and monitored W access by the CPU	1b	RAM part 1b+c:1,125 Kbytes
0x0600	0x09FC	256	768	*STATE* data	R and monitored W access by the CPU	**1c**	
0x0600	0x06FC	64	192	RDT_S[i]	*STATE* reciprocal values	1c1	
0x0700	0x07FC	64	192	TSF_S[i]	*STATE* TS values	1c2	
0x0800	0x08FC	64	192	ADT_S[i]	adapted values of *STATE*	1c3	

Table 71 Register and RAM address map (cont'd)

Addr. rangeStart	Addr. rangeEnd	Value number	Byte #	Content	Indication	Region	RAM size
0x0900	0x09FC	64	192	DT_S[i]	nom. STATE inc.	1c4	
0x4000	0x47FC…0x7FFC	512 …4096	1536 …12288	*TRIGGER* data	R and monitored W access of CPU	**2**	RAM part 2: 1,5…12 Kbytes
0x4000	0x41FC…4FFC	128… 1024	384…3072	RDT_T[i]	*TRIGGER* reciprocal values	2a	
0x4200…5000	0x43FC…5FFC	128…1024	384…3072	TSF_T[i]	*TRIGGER* TS values	2b	
0x4400…6000	0x45FC…6FFC	128… 1024	384…3072	ADT_T[i]	adapted values of *TRIGGER*	2c	
0x4600…7000	0x47FC…7FFC	128… 1024	384…3072	DT_T[i]	nom. *TRIGGER* increments	2d	

28.20.5.10.1 RAM Region 1

RAM region 1 has a size of 1,5 Kbytes and is used to store variables and parameters as well as the measured and calculated values for increments of *STATE*. The RAM 1 region is divided into two independent accessible RAM parts (a and b+c) with own ports. The address information is shown in the table above and the detailed description is performed in the following chapters. The RAM 1a is used to store the PMTR values got from ARU and in addition some intermediate calculation results of actions. RAM region 1b is used for variables needed for the prediction of increments, while RAM 1c is used to store time stamps, profile and duration of all the *STATE* inputs of the last FULL_SCALE region. All variables and values of RAM 1b+c part use a data width of up to 24 bits.

The RAM is to be initialized by the DPLL after HW-reset. All RAM cells must have a zero value after performing the initialize procedure. This is performed when setting The Init_RAM bit in the DPLL_RAM_INI register. The DPLL is only available after finishing this procedure. The initialization progress is shown in the status bits of the same register.

- **RAM Region 1a**: used for storage of PMTR values got from ARU; read and write access by the CPU is only possible, when the DPLL is disabled.
  The CPU Address range: 0x0200 – 0x03FC
- **RAM Region 1b**: usable for intermediate calculations and auxiliary values, data width of 3 bytes used for 24 bit values; A write access to this region results in an interrupt to the CPU, when enabled.
  Address range: 0x0400 – 0x05FC
- **RAM Region 1c**: Values of all STATE increments in FULL_SCALE, data width of 3 bytes used for 24 bit values; A write access to this region results in an interrupt to the CPU, when enabled.
  Address range: 0x0600 – 0x09FC

In RAM region 1c there is a difference in the amount of data. While for the RAM regions 1c1, 1c3 and 1c4 there are **2*(SNU+1-SYN_NS)** entries for SYSF=0 or **2*(SNU+1) -SYN_NS** entries for SYSF=1, for the RAM region 1c2 there are **2*(SNU+1)** entries (see DPLL_CTRL_0 and _1 registers). For the latter also the virtual events are considered, that means the gap is divided into equidistant parts each having the same position share as increments without

a gap. For that reason the CPU must extend the stored TSF_S[i] values in the RAM region 1c2 before the APS_1C3 is written. The write access to APS_1C3 sets the SYS bit in the DPLL_Status register in order to show the end of the synchronization process. Only when the SYS bit is set the PMTR values can consider more than the last increment duration for the action prediction by setting NUSE to a corresponding value.

Note: RAM regions 1b and 1c have a common port.

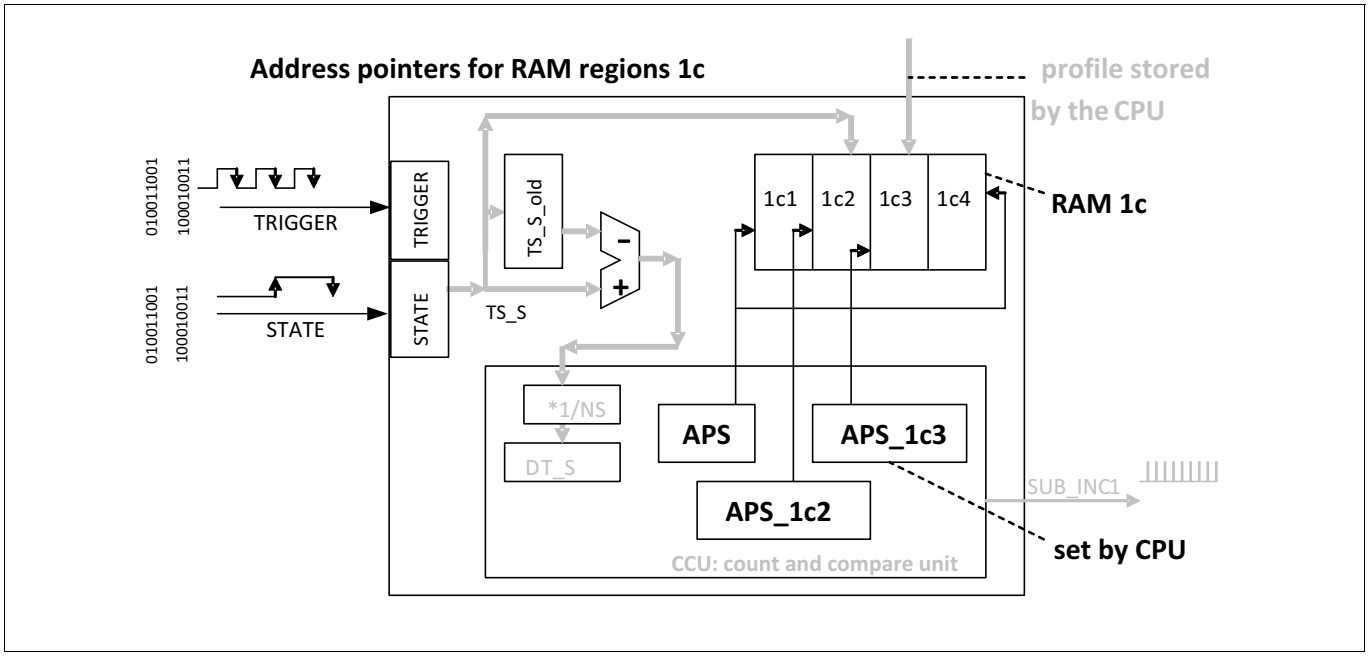


Figure 126 Address Pointer for RAM 1c

The address pointers for RAM region 1c are shown in the diagram above. While the address pointer APS points to the RAM regions 1c1 and 1c4, the address pointer APS_1C2 points to the time stamp field in the region 1c2. This is necessary, because in the time stamp field the gaps are extended to the number of nominal increments (see explanation above and also to the synchronization procedure explained in **Section 28.20.8.6.1**).The address pointer APS_1C3 is set by the CPU when the position is known and therefore the relation to the other address pointers is calculated. This setting of this profile address pointer synchronizes the RAM fields to one another. The synchronization is shown in the DPLL_STATUS register (see **Register DPLL_STATUS**) by the SYS bit.

28.20.5.10.2 RAM Region 2

The RAM region 2 has a configurable size of 1,5 to 12 Kilobytes and is used to store measured and calculated values for increments of *TRIGGER*. The address information is explained in **Section 28.20.11** while the meaning is explained in this chapter.

Because of up to 512 *TRIGGER* events in HALF_SCALE the fields 2a, b c and d must have up to 1024 storage places each. For 3 Bytes word size this does mean up to 12 k Byte of RAM region 2.

In order to save RAM size for configurations with less *TRIGGER* events the RAM is configurable by the offset switch Register OSW (0x001C) and the address offset value register of RAM region 2 AOSV_2 (0x0020). The RAM is to be initialized by the DPLL after HW-reset. All RAM cells must have a zero value after performing the initialize procedure. The DPLL is only available after finishing this procedure.

In RAM region 2 there is a difference in the amount of data. While for the RAM regions 2a, 2c and 2d there are **2*(TNU+1-SYN_NT)** entries, for the RAM region 2b there are **2*(TNU+1)** entries (see DPLL_CTRL_0 and _1 registers). For the latter also the virtual events are considered, that means the gap is divided into equidistant

parts each having the same position share as increments without a gap. For that reason the CPU must extend the stored TSF_T[i] values in the RAM region 2b before the APT_2C is written.

The write access to APT_2C sets the SYT bit in the DPLL_Status register in order to show the end of the synchronization process. Only when the SYT bit is set the PMTR values can consider more than the last increment duration for the action prediction by setting NUTE to a value greater than one.

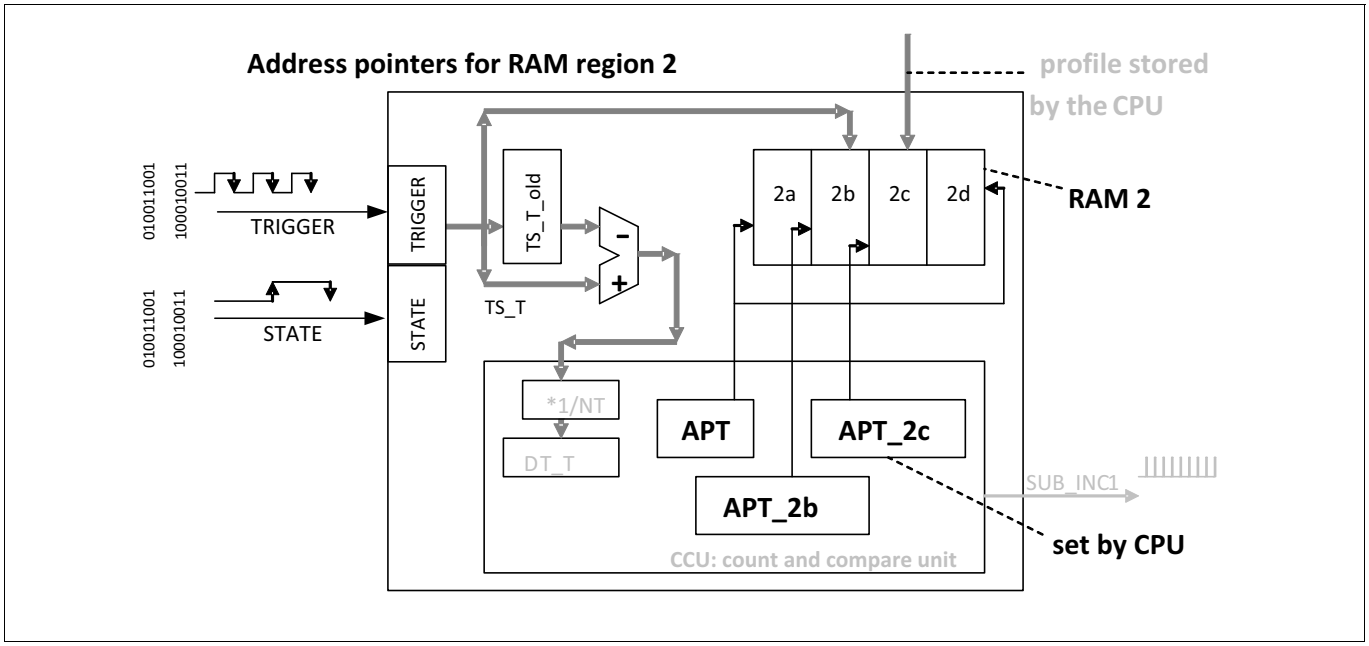


Figure 127 Address Pointer for RAM 2

The address pointers for RAM region 2 are shown in the diagram above. While the address pointer APT points to the RAM regions 2a and 2d, the address pointer APT_2B points to the time stamp field in the region 2b. This is necessary, because in the time stamp field the gaps are extended to the number of nominal increments (see explanation above and also to the synchronization procedure explained in **Section 28.20.8.6.1**).

The address pointer APT_2C is set by the CPU when the position is known and therefore the relation to the other address pointers is calculated. This setting of this profile address pointer synchronizes the RAM fields to one another. The synchronization is shown in the DPLL_STATUS register (see **Register DPLL_STATUS**) by the SYT bit.

28.20.5.11Software reset and DPLL deactivation

The DPLL module allows different options of deactivation and/or reset.To stop the operation of the DPLL module it is possible to deactivate the DPLL by setting of DPLL_CTRL_1.DEN = 0. This stops the calculations for the generation of the sub increments and the actions. Some control register areas are only configurable in this mode, some but not all register signals are set into an initial state. The RAM memory is not affected by DPLL deactivation at all. The behavior of the DPLL output signals and registers when deactivated is described in this document.

The deeper option to reconfigure the DPLL is the use of the software reset. When the DPLL module is deactivated setting DPLL_CTRL_1.SWR = 1 performs a reset of all DPLL registers and state controllers. The RAM memory is not affected by the software reset at all. After the software reset the DPLL module remains in deactivated state and the control registers must be configured again before operation (activation by DEN = 1) can start again.

The RAM modules can be reset (written to all zero) by activation of the memory init control bit (INIT_RAM) of the register DPLL_RAM_INI.If the RAM initialization is automatically done after power on reset or not depends on the GTM implementation.

A special case is the configuration of the control bit DPLL_CTRL_11.STATE_EXT. If this bit shall be modified during operation a software reset of the DPLL module is strongly recommended. A RAM initialisation should also be considered depending on the given application case.

28.20.6 Prediction of the current increment duration

28.20.6.1 The use of increments in the past

Past values to be considered for the prediction of TRIGGER

In order to take into account values of increments for *TRIGGER*s in the past, the NUTE value is configured to determine the number of past values. In addition the VTN has a value according to the number of virtual increments in the NUTE region. Because gaps come in to the NUTE region or leave it the VTN value must be updated by the CPU until NUTE is set to HALF_SCALE or FULL_SCALE. For the RAM regions 2a and 2d the value NUTE-VTN is to be considered while for the RAM region 2b only the NUTE value is to be considered. This is due to the fact that the time stamp entries in a gap are extended to the number of nominal increments, but duration entries not.

Past values to be considered for the prediction of STATE

In order to take into account values of increments for *STATE* in the past, the NUSE value is configured to determine the number of past values. In addition the VSN has a value according to the number of virtual increments in the NUSE region. Because gaps come in to the NUSE region or leave it the VSN value must be updated by the CPU until NUSE is set to HALF_SCALE or FULL_SCALE. For the RAM regions 1c1 and 1c4 in the past the value NUSE-VSN is to be considered while for the RAM region 1c2 only the NUSE value is to be considered. This is due to the fact that time stamp entries in a gap are extended to the number of nominal increments, but duration entries not.

28.20.6.2 Increment prediction in Normal Mode and for first PMSM forwards

For the prediction of increments and actions in normal mode the values are calculated as described in the following equations.

Please note, that the ascending order of calculation must be hold in order not to lose results still needed. It is important for *TRIGGER* values to calculate and store in the RAM region 2 all values according to equations up to DPLL-14 before DPLL-1a4…7, DPLL-1b1 and DPLL-1c1, while the last one overwrites DT_T[i] when NUTE (see **Register DPLL_NUTC**) is set to the FULL_SCALE range. Because the old value of DT_T[i] is also needed for equation DPLL-10 and DPLL-11 this value is stored temporarily at DT_T_ACT as shown by equation DPLL-1a or DPLL-1b respectively until all prediction calculations are done and after that equation DPLL-1a4…7, DPLL-1b1 and DPLL-1b1 updates DT_T[i]: update DT_T[i] after calculations of equation DPLL-14. For p=APT calculates in normal mode.

When using filter information of TRIGGER_FT, selected by IDT=1, it must be distinguished by IFP, if this filter information is time or position related.

In order to make possible to perform the automatic resolution corrections of equations DPLL-1a1a the filter unit in TIM module must operate using the time stamp clock.

28.20.6.2.1 Equations DPLL-1a to calculate TRIGGER time stamps

For calculation of time stamps use the filter delay information and an additional TRIGGER input delay value stored in register TIDEL (initial zero)

- TS_T_1=TRIGGER_TS - TIDEL **(DPLL-1a0)**
- TS_T=TS_T_1- FTV_Tx (for IDT=1 and IFP=0) **(DPLL-1a1)**

with

- FTV_Tx = FTV_T/8 (for LOW_RES = 1 and TS0_HRT = 0) **(DPLL-1a1a)**
- FTV_Tx = FTV_T (for LOW_RES = 0 or TS0_HRT = 1) **(DPLL-1a1b)**

and

- TS_T= TS_T_1- FTV_T *(CDT_TX/NMB_T)_old[1)] for (IDT=1 and IFP=1) **(DPLL-1a2)**

this can be also calculated using the value of ADD_IN_CALN:

- TS_T=TS_T_1 - FTV_T *(1/ADD_IN_CALN_old[1)]) for (IDT=1 and IFP=1) **(DPLL-1a3)**

1) Consider values, calculated for the last increment; position related filter values are only considered up to at least 1 ms time between two TRIGGER events. The reciprocal value is stored using a 32-bit fractional part, while only the 24 lower bits are used - for explanation see note 4) at DPLL_CTRL_0 register. The value of 1/ADD_IN_CALN_old or (CDT_TX/NMB_T)_old is set to 0xFFFFFF in the case of an overflow.

Note: CDT_TX is the predicted duration of the last TRIGGER increment and NMB_T the calculated number of SUB_INC1 events in the last increment, because the new calculations are done by equations DPLL-5 and DPLL-21 for the current increment after that. Therefore in equation DPLL-1a3 the value ADD_IN of the last increment is used (see equation DPLL-25). SYN_T_OLD is the number of TRIGGER events including missing TRIGGERs as specified in the NUTC register for the last increment, with the initial value of 1.

For storage of time stamps in the RAM see also equations DPLL-1a4 ff. after calculation of actions, **Section 28.20.7.5.1**.

28.20.6.2.2 Equation DPLL-1b to calculate DT_T_ACT (nominal value)

- DT_T_ACT= (TS_T - TS_T_OLD)/SYN_T_OLD **(DPLL-1b)**

For the case SYT=0 (still no synchronization to the profile) the values SYN_T and SYN_T_OLD are still assumed as having the value 1.

Correct the current increment duration value got by equation DPLL-1b in the case of physical deviations (ADT=1) by

- DT_T_ACT = DT_T_ACT * (1 - PDC_T+PDC_T^2+PDC_T^3) **(DPLL-1b1)**

with the relative correction value for the last increment (for SMC=0)

- PDC_T = PD_OLD/(MLT+1) **(DPLL-1b2)**

for SMC=1 use

- PDC_T = PD_OLD/(MLS1) (DPLL-1b3)

Note: The term (1 - PDC_T+PDC_T^2+PDC_T^3) is representing the third order Taylor series of the term 1/(1+PCT_T), which is chosen to reduce the additional time delay due to the more complex computation.

28.20.6.2.3 Equation DPLL-1c to calculate RDT_T_ACT (nominal value)

- RDT_T_ACT= 1/ DT_T_ACT **(DPLL-1C)**

28.20.6.2.4 Equation DPLL-2a1 to calculate QDT_T_ACT

Relation of the recent last two increment values for APT=p in forward direction (DIR1=0)

- QDT_T_ACT = DT_T_ACT * RDT_T[p-1] **(DPLL-2a1)**

QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

Note: QDT_T_ACT uses a 6 bit integer part and an 18 bit fractional part.

28.20.6.2.5 Equation DPLL-3 to calculate the error of last prediction

When q = NUTE-VTN considers for the error calculation only the last valid prediction values for DIR1=0:
Calculate the error of the last prediction when using only RDT_T_FS1, DT_T[p-q], DT_T[p-q-1] and DT_T[p-1] for the prediction of DT_T[p]:

- EDT_T=DT_T_ACT-(DT_T[p-1]*QDT_T[p-q]) **(DPLL-3)**

with

- QDT_T[p-q] = DT_T[p-q] * RDT_T[p-q-1] for FST=0 **(DPLL-2b1)**
- QDT_T[p-q] = DT_T[p-q] * RDT_T_FS1 for FST=1 **(DPLL-2b2)**

and FST has the meaning: NUTE=FULL_SCALE (see NUTC register)
while
QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and a 18 bit fractional part.

Note: QDT_T[p-q] uses a 6 bit integer part and a 18-bit fractional part.

28.20.6.2.6 Equation DPLL-4 to calculate the weighted average error

for SYT=1 calculate:

- MEDT_T := (EDT_T + MEDT_T)/2 **(DPLL-4)**

28.20.6.2.7 Equations DPLL-5 to calculate the current increment value

nominal increment value (for ADT=0):

- CDT_TX _nom = (DT_T_ACT + MEDT_T)* QDT_T[p-q+1] **(DPLL-5a1)**

nominal increment value (for ADT=1):

- CDT_TX _nom_corr = CDT_TX _nom*(1 + CDC_T) **(DPLL-5a2)**

with for SMC=0

- CDC_T = PD/ (MLT+1) **(DPLL-5a3)**

or for SMC=1 use

- CDC_T = PD/ (MLS1) **(DPLL-5a4)**

and with (for q>1):

- QDT_T[p-q+1] = DT_T[p-q+1] * RDT_T[p-q] **(DPLL-2c)**

and for q=1 use equation DPLL-2a1.

while
QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

The CDT_TX_nom value is limited by the relation

- CTN_MIN < CDT_TX_nom < CTN_MAX. **(DPLL-5c)**

When the calculated value exceeds one of the limit, it is replaced by the corresponding limit value.

The expected duration to the next TRIGGER event is (for ADT=0)

- CDT_TX = CDT_TX _nom * SYN_T **(DPLL-5b)**

The expected duration to the next TRIGGER event is (for ADT=1)

- CDT_TX = CDT_TX _nom_corr * SYN_T **(DPLL-5b)**

Note: QDT_T[p-q+1] uses a 6-bit integer part and a 18 bit fractional part.

Note: In the case of an overflow in equations DPLL-5a or b set the value to 0xFFFFFF and the corresponding CTO bit in the DPLL_STATUS register. In the case of negative values set CDT_TX to 0x0 without any effect to the CTO bit.

28.20.6.3 Increment prediction in Emergency Mode and for second PMSM forwards

Please note, that the ascending order of calculations for *STATE* and storage of the values in the RAM region 1c must be hold in order not to lose results still needed. The same considerations as done for DT_T_ACT are valid for DT_S_ACT (equation DPLL-6a4…7, DPLL-6b1 and DPLL-6b1): update TD_S[i] only after calculations of equation DPLL-14.

When using filter information of STATE_FT, selected by IDS=1, it must be distinguished by IFP, if this filter information is time or position related.

In order to make possible to perform the automatic resolution corrections of equations DPLL-6a1a the filter unit in TIM must operate using the time stamp clock.

28.20.6.3.1 Equations DPLL-6a to calculate STATE time stamps

For calculation of time stamps use the filter delay information, the additional STATE input delay value stored in the register SIDEL (initial zero) and use p=APS while DIR2=0:

- TS_S_1=STATE_TS - SIDEL **(DPLL-6a0)**
- TS_S=TS_S_1 - FTV_Sx (for IDS=1 and IFP=0) **(DPLL-6a1)**

with

- FTV_Sx = FTV_S/8 (for LOW_RES = 1 and TS0_HRS = 0) **(DPLL-6a1a)**
- FTV_Sx = FTV_S (for LOW_RES = 0 or TS0_HRS = 1) **(DPLL-6a1b)**

and

- TS_S=TS_S_1 - FTV_S *(CDT_SX/NMB_S)_old[1] (for IDS=1 and IFP=1) **(DPLL-6a2)**

this can be also calculated using the value of ADD_IN_CALE:

- TS_S=TS_S_1 - FTV_S *(1/ADD_IN_CALE)_old[1] (for IDS=1 and IFP=1) **(DPLL-6a3)**

with

see also equations DPLL-6a4 ff. at **Section 28.20.6.2** for TRIGGER.

1) Consider values, calculated for the last increment; position related filter values are only considered up to at least 1 ms time between two STATE events. The reciprocal value is stored using a 32 bit fractional part, while only the 24 lower bits are used - for explanation see note 4) at DPLL_CTRL_0 register. The value of 1/ADD_IN_CALE_old or (CDT_SX/NMB_S)_old is set to 0xFFFFFF in the case of an overflow.

Note: CDT_SX is the predicted duration of the last STATE increment and NMB_S the calculated number of SUB_INC1 events in the last increment, because the new calculations are done by equations DPLL-10 and DPLL-22 respectively for the current increment after that. Therefore in equation DPLL-6a3 the value ADD_IN of the last increment is used (see equation DPLL-26). SYN_S_OLD is the number of increments including missing STATEs as specified in the NUSC register for the last increment with the initial value of 1. The update to the RAM region 1c4 is done after all related calculations (see equation DPLL-6b1 for this reason).

28.20.6.3.2 Equation DPLL-6b to calculate DT_S_ACT (nominal value)

- DT_S_ACT= (TS_S - TS_S_OLD)/SYN_S_OLD **(DPLL-6b)**

For the case SYS=0 (still no synchronization to the profile) the values SYN_S and SYN_S_OLD are still assumed as having the value 1.

Correct the current increment duration value got by equation DPLL-6b in the case of physical deviations (ADS=1) by

- DT_S_ACT = DT_S_ACT * (1 - PDC_S+PDC_S2+PDC_S3) **(DPLL-6b1)**

with the relative correction value for the last increment (for SMC=0)

- PDC_S = PD_S_OLD/(MLS1) **(DPLL-6b2)**

for SMC=1 use

- PDC_S = PD_S_OLD/(MLS2) **(DPLL-6b3)**

Note: The term (1 - PDC_S+PDC_S2+PDC_S3) is representing the third order Taylor series of the term 1/(1+PDC_S), which is chosen to reduce the additional time delay due to the more complex computation.

28.20.6.3.3 Equation DPLL-6c to calculate RDT_S_ACT (nominal value)

- RDT_S_ACT = 1/ DT_S_ACT **(DPLL-6c)**

28.20.6.3.4 Equation DPLL-7a1 to calculate QDT_S_ACT

for APS=p in forward direction (DIR2=0)

- QDT_S_ACT = DT_S_ACT * RDT_S[p-1] **(DPLL-7a1)**

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and an 18 bit fractional part.

Note: QDT_S_ACT uses a 6 bit integer part and a 18 bit fractional part.

28.20.6.3.5 Equation DPLL-8 to calculate the error of last prediction

with q= NUSE-VSN when using QDT_S[p-q] and DT_S[p-1] for the prediction of DT_S[p]

- EDT_S = DT_S_ACT - (DT_S[p-1] * QDT_S[p-q]) **(DPLL-8)**

and with

- QDT_S[p-q] = DT_S[p-q] * RDT_S[p-q-1] for FSS=0 **(DPLL-7b1)**
- QDT_S[p-q] = DT_S[p-q] * RDT_S_FS1 for FSS=1 **(DPLL-7b2)**

and FSS has the meaning: NUSE=FULL_SCALE (see NUSC register)

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and a 18 bit fractional part.

Note: QDT_S[p-q] uses a 6 bit integer part and a 18 bit fractional part.

28.20.6.3.6 Equation DPLL-9 to calculate the weighted average error

for SYS=1 calculate:

- MEDT_S := (EDT_S + MEDT_S)/2 **(DPLL-9)**

28.20.6.3.7 Equations DPLL-10 to calculate the current increment (nominal value)

nominal increment value (for ADS=0):

- CDT_SX _nom= (DT_S_ACT + MEDT_S)* QDT_S[p-q+1] **(DPLL-10a1)**

or nominal increment value (for ADS=1):

- CDT_SX _nom_corr= CDT_SX _nom*(1+CDC_S) **(DPLL-10a2)**

with for SMC=0

- CDC_S = PD / (MLS1) **(DPLL-10a3)**

for SMC=1 use

- CDC_S = PD / (MLS2) **(DPLL-10a4)**

and with

- QDT_S[p-q+1] = DT_S[p-q+1] * RDT_S[p-q] (for q>1) **(DPLL-7c)**
  see equation DPLL-7a1 for q=1

while

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and a 18 bit fractional part.

The CDT_SX_nom value is limited by the relation

- CSN_MIN < CDT_SX_nom < CSN_MAX **(DPLL-10c)**

When the calculated value exceeds one of the limit, it is replaced by the corresponding limit value.

The expected duration to the next *STATE* event is (for ADT=0)

- CDT_SX = CDT_SX _nom * SYN_T **(DPLL-10b)**

The expected duration to the next *STATE* event is (for ADT=1)

- CDT_SX = CDT_SX _nom_corr * SYN_S **(DPLL-10b)**

Note: QDT_S[p-q+1] uses a 6 bit integer part and a 18 bit fractional part.

Note: In the case of an overflow in equations DPLL-10a or b set the value to 0xFFFFFF and the corresponding CSO bit in the DPLL_STATUS register. In the case of negative values set CDT_SX to 0x0 without any effect to the CSO bit. All 5 steps above (DPLL-6 to DPLL-10) are only needed in emergency mode. For the normal mode the calculations of equations DPLL-6 and DPLL-7 are done solely in order to get the values needed for a sudden switch to emergency mode.

28.20.6.4 Increment prediction in Normal Mode and for first PMSM backwards

28.20.6.4.1 Equations DPLL-2a2 to calculate QDT_T_ACT backwards

- QDT_T_ACT = DT_T_ACT * RDT_T[p+1] **(DPLL-2a2)**

QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and a 18 bit fractional part.

28.20.6.4.2 Equation DPLL-3a to calculate of the error of last prediction

When q = NUTE-VTN and DIR1=1 using only QT_T[p+q] and DT_T[p+1] for the prediction of DT_T[p]

- EDT_T=DT_T_ACT-(DT_T[p+1]*QDT_T[p+q] **(DPLL-3a)**

with

- QDT_T[p+q] = DT_T[p+q] * RDT_T[p+q+1] for FST=0 **(DPLL-2b3)**
- QDT_T[p+q] = DT_T[p+q] * RDT_T_FS1 for FST=1 **(DPLL-2b4)**

and FST has the meaning: NUTE=FULL_SCALE (see NUTC register)

QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and a 18 bit fractional part.

28.20.6.4.3 Equation DPLL-4 to calculate the weighted average error

For SYT=1 calculate: MEDT_T := (EDT_T + MEDT_T) / 2 (DPLL-4)

28.20.6.4.4 Equation DPLL-5 to calculate the current increment value

nominal increment value (for ADT=0):

- CDT_TX _nom = (DT_T_ACT + MEDT_T)* QDT_T[p+q-1] **(DPLL-5a5)**

nominal increment value (for ADT=1):

- CDT_TX _nom_corr = CDT_TX _nom * (1 + CDC_T) **(DPLL-5a6)**

with for SMC=0

- CDC_T = PD / (MLT + 1) **(DPLL-5a3)**

for SMC=1 use

- CDC_T = PD / (MLS1) **(DPLL-5a3)**

and with

- QDT_T[p+q-1] = DT_T[p+q-1] * RDT_T[p+q] (for q>1) **(DPLL-2c1**

for q=1 use equation DPLL-2a1.

while

QDT_T_ACT as well as QDT_T[i] have a 24 bit value using a 6 bit integer part and a 18 bit fractional part.

The CDT_TX_nom value is limited by the relation

- CTN_MIN < CDT_TX_nom < CTN_MAX. **(DPLL-5c)**

When the calculated value exceeds one of the limit, it is replaced by the corresponding limit value.
and the expected duration to the next TRIGGER event (for ADT=0)

- CDT_TX = CDT_TX _nom * SYN_T **(DPLL-5b)**

the expected duration to the next TRIGGER event (for ADT=1)

- CDT_TX = CDT_TX _nom_corr * SYN_T **(DPLL-5b)**

Note: In the case of an overflow in equations DPLL-5a1 or b set the value to 0xFFFFFF and the corresponding CTO bit in the DPLL_STATUS register. In the case of negative values set CDT_TX(_nom) to 0x0.

28.20.6.5 Increment prediction in Emergency Mode and for second PMSM backwards

28.20.6.5.1 Equation DPLL-7a2 to calculate QDT_S_ACT backwards

- QDT_S_ACT = DT_S_ACT * RDT_S[p+1] **(DPLL-7a2)**

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and a 18 bit fractional part.

28.20.6.5.2 Equation DPLL-8a to calculate the error of the last prediction

While q= NUSE-VSN, use only QDT_S[p+q] and DT_S[p+1] for the prediction of DT_S[p]

- EDT_S = DT_S_ACT - (DT_S[p+1] * QDT_S[p+q]) **(DPLL-8a)**

with

- QDT_S[p-q] = DT_S[p+q] * RDT_S[p+q+1] for FSS=0 **(DPLL-7b3)**
- QDT_S[p-q] = DT_T[p+q] * RDT_S_FS1 for FSS=1 **(DPLL-7b4)**

and FSS has the meaning: NUSE=FULL_SCALE (see NUSC register)

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and a 18 bit fractional part.

28.20.6.5.3 Equation DPLL-9 to calculate the weighted average error

For SYS=1 calculate:

- MEDT_S := (EDT_S + MEDT_S)/2 **(DPLL-9)**

28.20.6.5.4 Equations DPLL-10 to calculate the current increment value

nominal increment value (for ADS=0):

- CDT_SX _nom= (DT_S_ACT + MEDT_S)* QDT_S[p+q-1] **(DPLL-10a5)**

or nominal increment value (for ADS=1):

- CDT_SX _nom_corr= CDT_SX _nom*(1+CDC_S) **(DPLL-10a6)**

with for SMC=0

- CDC_S = PD / (MLS1) **(DPLL-10a3)**

for SMC=1 use

- CDC_S = PD / (MLS2) **(DPLL-10a4)**

and with

- QDT_S[p+q-1] = DT_S[p+q-1] * RDT_S[p+q] (for q>1) **(DPLL-7c1)**

  for q=1 use equation DPLL-7a.

while

QDT_S_ACT as well as QDT_S[i] have a 24 bit value using a 6 bit integer part and a 18 bit fractional part.

The CDT_SX_nom value is limited by the relation

- CSN_MIN < CDT_SX_nom < CSN_MAX. **(DPLL-10c)**

When the calculated value exceeds one of the limit, it is replaced by the corresponding limit value.

and calculate the expected duration to the next *STATE* event (for ADT=0)

- CDT_SX = CDT_SX _nom * SYN_T **(DPLL-10b)**

and calculate the expected duration to the next *STATE* event (for ADT=1)

- CDT_SX = CDT_SX _nom_corr * SYN_S **(DPLL-10b)**

Note: In the case of an overflow in equations DPLL-10a1 or b set the value to 0xFFFFFF and the corresponding CSO bit in the DPLL_STATUS register. In the case of negative values set CDT_SX(_nom) to 0x0.
All 5 steps above (DPLL-6 to DPLL-10) are only needed in emergency mode. For the normal mode the calculations of equations DPLL-6 and DPLL-7 are done solely in order to get the values needed for a sudden switch to emergency mode.

28.20.7 Calculations for actions

As already shown for the calculation of the current interval by equations DPLL-1 to DPLL-10 for the prediction of actions a similar calculation is to be done, as shown by the equations DPLL-11. to DPLL-14. The calculation of actions is also needed when the DPLL is used for synchronous motor control applications (SMC=1, see DPLL_CTRL_1 register). For action prediction purposes the measured time periods of the past (one FULL_SCALE back, when the corresponding NUTE or NUSE values are set properly by the CPU) are used. The calculation can be explained by the following assumptions, which are considerably simple:

Take the corresponding increments for prediction in the past and put the sum of it in relation to the increment (DT_T[k], DT_S[k], with k >= 0, which is represented by the time stamp difference) which is exactly one FULL_SCALE period in the past (DPLL-11 or DPLL-13 respectively). Make a prediction for the coming sum of increments using the current measured increment (DT_T_ACT or DT_S_ACT respectively, that means DPLL-1 or DPLL-6 respectively) and add a weighted average error (DPLL-3 and DPLL-4 or DPLL-8 and DPLL-9 respectively, calculated for one increment prediction) before multiplication with the relation of equation DPLL-11 or DPLL-13 respectively in order to get the result as described by equations DPLL-12 or DPLL-14 respectively.

In order to avoid division operations instead of the increment (DT_T[k], DT_S[k], with k > 0) in the past its reciprocal value (RDT_T[k], RDT_S[k], with k > 0) is used, which is stored also in RAM. For the calculation of actions perform always a new refined calculation as long as the resulting time stamp is not in the past. In the other case the TSAC/PSAC values (time/position stamp of action calculated) is set to the time/position stamp of the last input event (TRIGGER/STATE), the ACT_N[i] bit in the DPLL_ACT_STA register is reset, while the corresponding

ACT_N[i] bit in the DPLL_ACT_STA_shadow register is set. Each new PMTR_i value will set this ACT_N[i] bit again and reset the correspondent shadow bit until a new calculation is performed.

Please make sure that the prediction parameters are chosen such that under all conditions (acceleration/deceleration) the values of PDT_T, PDT_S and DTA respectively do not exceed the value 0xFFFFFF. This requirement can limit the predicted position range in the case of very low speed.

Action updates at highest speed

Up to 32 action values can be calculated. For the shortest increment duration (23,4 µs) not all of them can be updated with each active input event. Please notice the following conditions and parameters for an estimation of possible results.

All time estimation values are given for a system clock frequency of 100 MHz and the assumption, that the calculation of the DPLL is not impeded by a remote read or write access to the DPLL RAMs. Each RAM access is to be considered by an additional delay of about 40 ns ($t_{remote\_RAM\_acces}$, to be precised later). When using a different system clock frequency the calculation duration is extended accordingly.

1. Typical time needed for basic operations (RAM update, pointer calculation and SUB_INC generation for normal, emergency mode or one PMSM: t_{basic_0} = 9,9 µs.
2. Typical time needed basic operations (RAM update, pointer calculation and SUB_INC generation for two PMSMs): t_{basic_1} = 11,0 µs.
3. Typical time needed to calculate one action: t_{action_i} = 3,7 µs.

Please notice that the above mentioned values are observed worst case values, when the two state machines of TRIGGER and STATE are both in operation.

These values allow the calculation of at least 3 action values for each input event for all specified increments duration. The complete time needed for the basic operation, n action calculations and k remote RAM access operations can be calculated as follows:

$t_{complete} = t_{basic\_0/1} + n * t_{action\_}i + k * t_{remote\_RAM\_access}$.

Typical applications

Normal and emergency mode

For a typical application with the shortest increment duration of 100 µs in normal or emergency mode the calculation of up to 24 action values can be performed for each active input event.

One PMSM

For one PMSM and a typical shortest increment time of 39 µs there is the calculation of up to 7 action values possible for each input event.

Two PMSMs with restricted action calculations

When only one PMSM uses the action calculation service an the shortest increment duration is 39 µs, there can up to 7 actions served for each valid input event.

Two PMSMs with unrestricted action calculations

When 2 PMSMs are used and both use the action calculation service at a minimal increment duration of 39 µs there are up to 7 action calculations possible for each of the two engines - that means up to 14 action calculations per increment in average.

28.20.7.1 Action calculations for TRIGGER forwards

valid for RMO=0 or for SMC=1 with

p=APT_2B, t=APT, m=NA[i] (part w), mb=NA[i](part b)/1024, NUTE-VTN=q, NUTE=n

28.20.7.1.1 Equation DPLL-11a1 to calculate the time prediction for an action

For DIR1=0 and q>m calculate:

- PDT_T[i] = (TSF_T[p+m-n] - TSF_T[p-n] + mb* DT_Tx[t-q+1]) * RDT_T[t-q] **(DPLL-11a1)**

with

- DT_Tx[t-q+1] = DT_T[t-q+1] for TS0_HRT=0 **(DPLL-11b2)**

or

- DT_Tx[t-q+1] = DT_T[t-q+1]/8 for TS0_HRT=1 **(DPLL-11b3)**

and while the multiplication with mb does mean the fractional part of NA[i].

For SMC=0 and RMO=0 calculate for DIR1=0 all 32 actions in forward direction, if requested; in the case SMC=1 calculate up to 16 actions 0 to 15 in dependence of the *TRIGGER* input.

28.20.7.1.2 Equation DPLL-11a2 to calculate the time prediction for an action

For SYT=1, NUTE= 2*(TNU+1), q>m and DIR1=0 equation DPLL-11a2 is equal to

- PDT_T[i] = (TSF_T[p+m] - TSF_T[p]+ mb* DT_Tx[t-q+1]) * RDT_T[t] **(DPLL-11a2)**

with

- DT_Tx[t-q+1] = DT_T[t-q+1] for TS0_HRT=0 **(DPLL-11b2)**

or

- DT_Tx[t-q+1] = DT_T[t-q+1]/8 for TS0_HRT=1 **(DPLL-11b3)**

28.20.7.1.3 Equation DPLL-11b to calculate the time prediction for an action

for DIR1=0, NUTE-VTN=q, q (< or =) m, n>1 and t=APT:

- PDT_T[i] = (m+mb)*DT_Tx[t-q+1] * RDT_T[t-q] **(DPLL-11b)**

with

- DT_Tx[t-q+1] = DT_T[t-q+1] for TS0_HRT=0 **(DPLL-11b2)**

or

- DT_Tx[t-q+1] = DT_T[t-q+1]/8 for TS0_HRT=1 **(DPLL-11b3)**

Note: Make the calculations above before updating the TSF_T[i] values according to equations DPLL-1c3 ff.

28.20.7.1.4 Equation DPLL-11c to calculate the time prediction for an action

for n=1 (this is always valid for SYT=0)

- PDT_T[i] = (m+mb)* DT_T_ax* RDT_T[t-1] **(DPLL-11c)**

with

- DT_T_ax = DT_T_ACT for TS0_HRT=0 **(DPLL-1a4a)**

or

- DT_T_ax = DT_T_ACT/8 for TS0_HRT=1 **(DPLL-1a4b)**

Note: For the relevant last increment add the fractional part of DT_T_ACT as described in NA[i].

28.20.7.1.5 Equation DPLL-12 to calculate the duration value until action

DTA[i] = (DT_T_ACT + MEDT_T)* PDT_T[i] **(DPLL-12)**

Note: All 5 steps in equations DPLL-11 to DPLL-12 are only calculated in normal mode.

28.20.7.2 Action calculations for TRIGGER backwards

valid for RMO=0 or for SMC=1 with

p=APT_2B, t=APT, m=NA[i] (part w), mb=NA[i](part b)/1024, q= NUTE-VTN and n=NUTE

For SMC=0 and RMO=0 calculate for DIR1=1 all 32 actions in backward direction for special purposes; in the case SMC=1 calculate up to 16 actions 0 to 15 in dependence of the *TRIGGER* input.

28.20.7.2.1 Equation DPLL-11a3 to calculate the time prediction for an action

For DIR1=1 and q>m calculate:

- PDT_T[i] = (TSF_T[p-m+n] - TSF_T[p+n] + mb* DT_Tx[t+q-1]) * RDT_T[t+q] **(DPLL-11a3)**

with

- DT_Tx[t+q-1] = DT_T[t+q-1] for TS0_HRT=0 **(DPLL-11b4)**

or

- DT_Tx[t+q-1] = DT_T[t+q-1]/8 for TS0_HRT=1 **(DPLL-11b5)**

28.20.7.2.2 Equation DPLL-11a4 to calculate the time prediction for an action

For SYT=1 and NUTE = 2*(TNU+1), q>m, VTN=2*SYN_NT and hence **NUTE-VTN = 2*(TNU+1-SYN_NT)** for DIR1=1 this is equal to

- PDT_T[i] = (TSF_T[p-m] - TSF_T[p]+ mb* DT_Tx[t+q-1]) * RDT_T[t] **(DPLL-11a4)**

with

- DT_Tx[t+q-1] = DT_T[t+q-1] for TS0_HRT=0 **(DPLL-11b4)**

or

- DT_Tx[t+q-1] = DT_T[t+q-1]/8 for TS0_HRT=1 **(DPLL-11b5)**

Note: Make the calculations above before updating the TSF_T[i] values according to equations DPLL-1c3 ff.

28.20.7.2.3 Equation DPLL-11b1 to calculate the time prediction for an action

For NUTE-VTN =q, q (< or =) m the following equation is valid for n>1 and t=APT:

- PDT_T[i] = (m+mb)*DT_Tx[t+q-1] * RDT_T[t+q] **(DPLL-11b1)**

with

- DT_Tx[t+q-1] = DT_T[t+q-1] for TS0_HRT=0 **(DPLL-11b4)**

or

- DT_Tx[t+q-1] = DT_T[t+q-1]/8 for TS0_HRT=1 **(DPLL-11b5)**

28.20.7.2.4 Equation DPLL-11c1 to calculate the time prediction for an action

for n=1 (this is always valid for SYT=0)

- PDT_T[i] = (m+mb)* DT_T_ax * RDT_T[t+1] **(DPLL-11c1)**

with

- DT_T_ax = DT_T_ACT for TS0_HRT=0 **(DPLL-1a4a)**

or

- DT_T_ax = DT_T_ACT/8 for TS0_HRT=1 **(DPLL-1a4b)**

Note: For the relevant last increment add the fractional part of DT_T_ACT as described in NA[i].

28.20.7.2.5 Equation DPLL-12 to calculate the duration value for an action

DTA[i] = (DT_T_ACT + MEDT_T)* PDT_T[i] **(DPLL-12)**

Use the results of equations DPLL-1a, b, DPLL-3 and DPLL-4 for the above calculation

28.20.7.3 Action calculations for STATE forwards

valid for RMO=1 with

p=APS_1C2, t=APS, m=NA[i](part w) mb=NA[i](part b)/1024, NUSE-VSN = q and NUSE=n>m

For SMC=0 and RMO=1 calculate for DIR2=0 all 32 actions in forward direction, if requested; in the case SMC=1 and RMO=1 calculate up to 16 actions 16 to 31 in dependence of the *STATE* input.

Note: All 5 steps of equations DPLL-13 to DPLL-14 are only calculated in emergency mode or for SMC=1 in combination with RMO=1.

28.20.7.3.1 Equation DPLL-13a1 to calculate the time prediction for an action

For DIR2=0 and q>m calculate:

PDT_S[i] = (TSF_S[p+m-n] - TSF_S[p-n] + mb* DT_Sx[t-q+1] * RDT_S[t-q] **(DPLL-13a1)**

with

- DT_Sx[t-q+1]= DT_S[t-q+1] for TS0_HRS=0 **(DPLL-13b2)**

or

- DT_Sx[t-q+1] = DT_S[t-q+1]/8 for TS0_HRS=1 **(DPLL-13b3)**

28.20.7.3.2 Equation DPLL-13a2 to calculate the time prediction for an action

For **SYS=1 and NUSE=2*(SNU+1)**, q>m, SYSF=0, VSN=2*SYN_NS and hence NUSE-VSN = 2*(SNU+1-SYN_NS) equation DPLL-13a1 is equal to

- PDT_S[i] = (TSF_S[p+m] - TSF_S[p]+ mb*DT_Sx[t-q+1]) * RDT_S[t] **(DPLL-13a2)**

with

- DT_Sx[t-q+1] = DT_S[t-q+1] for TS0_HRS=0 **(DPLL-13b2)**

or

- DT_Sx[t-q+1] = DT_S[t-q+1]/8 for TS0_HRS=1 **(DPLL-13b3)**

28.20.7.3.3 Equation DPLL-13b to calculate the time prediction for an action

For NUSE -VTN=q, q (< or =) m and n>1:

- PDT_S[i] = (m+mb)*DT_Sx[t-q+1] * RDT_S[t-q] **(DPLL-13b)**

with

- DT_Sx[t-q+1] = DT_S[t-q+1] for TS0_HRS=0 **(DPLL-13b2)**

or

- DT_Sx[t-q+1] = DT_S[t-q+1]/8 for TS0_HRS=1 **(DPLL-13b3)**

28.20.7.3.4 Equation DPLL-13c to calculate the time prediction for an action

for n=1

- PDT_S[i] = (m+mb)* DT_S_ax * RDT_S[t-1] **(DPLL-13c)**

with

- DT_S_ax = DT_S_ACT for TS0_HRS=0 **(DPLL-6a4a)**

or

- DT_S_ax = DT_S_ACT/8 for TS0_HRS=1 **(DPLL-6a4b)**

28.20.7.3.5 Equation DPLL-14 to calculate the duration value for an action

- DTA[i] = (DT_S_ACT + MEDT_S)* PDT_S[i] **(DPLL-14)**

Use the results of DPLL-7, DPLL-8 and DPLL-9 for the above calculation

28.20.7.4 Action calculations for STATE backwards

valid for RMO=1 with

p=APS_1C2, t=APS, m=NA[i](part w) mb=NA[i](part b)/1024, NUSE-VSN = q and NUSE=n

For SMC=0 and RMO=1 calculate for DIR1=1 all 32 actions in backwards mode for special purposes; in the case SMC=1 and RMO=1 calculate up to 16 actions 16 to 31 in dependence of the *STATE* input.

28.20.7.4.1 Equation DPLL-13a3 to calculate the time prediction for an action

For (DIR2= 1 (SMC=1) or DIR1=1 (SMC=0)) and q>m calculate

- PDT_S[i] = (TSF_S[p-m+n] - TSF_S[p+n] + mb* DT_Sx[t+q-1]) * RDT_S[t+q] **(DPLL-13a3)**

with

- DT_Sx[t+q-1]= DT_S[t+q-1] for TS0_HRS=0 **(DPLL-13b4)**

or

- DT_Sx[t+q-1] = DT_S[t+q-1]/8 for TS0_HRS=1 **(DPLL-13b5)**

28.20.7.4.2 Equation DPLL-13a4 to calculate the time prediction for an action

For SYS=1, **NUSE=2*(SNU+1)**, q>m, SYSF=0, VSN=2*SYN_NS and hence NUSE-VSN = 2*(SNU+1-SYN_NS) equation DPLL-13a3 is equal to

- PDT_S[i] = (TSF_S[p-m] - TSF_S[p]+ mb* DT_Sx[t+q-1]) * RDT_S[t] **(DPLL-13a4)**

with

- DT_Sx[t+q-1]= DT_S[t+q-1] for TS0_HRS=0 **(DPLL-13b4)**

or

- DT_Sx[t+q-1] = DT_S[t+q-1]/8 for TS0_HRS=1 **(DPLL-13b5)**

28.20.7.4.3 Equation DPLL-13b1 to calculate the time prediction for an action

For NUSE-VSN =q, q (< or =) m, NUSE=n and n>1:

- PDT_S[i] = m*DT_Sx[t+q-1] * RDT_S[t+q] **(DPLL-13b1)**

with

- DT_Sx[t+q-1] = DT_S[t+q-1] for TS0_HRS=0 **(DPLL-13b4)**

or

- DT_Sx[t+q-1] = DT_S[t+q-1]/8 for TS0_HRS=1 **(DPLL-13b5)**

28.20.7.4.4 Equation DPLL-13c1 to calculate the time prediction for an action

for n=1

- PDT_S[i] = (m+mb)* DT_S_ax * RDT_S[t+1] **(DPLL-13c1)**

with

- DT_S_ax = DT_S_ACT for TS0_HRS=0 **(DPLL-6a4a)**

or

- DT_S_ax = DT_S_ACT/8 for TS0_HRS=1 **(DPLL-6a4b)**

28.20.7.4.5 Equation DPLL-14 to calculate the duration value until action

DTA[i] = (DT_S_ACT + MEDT_S)* PDT_S[i] (DPLL-14)

Use the results of DPLL-7, DPLL-8 and DPLL-9 for the above calculation

28.20.7.5 Update of RAM in Normal and Emergency Mode

After considering the calculations for up to all 24 actions according to equations (DPLL-11, DPLL-12), only when going back to state 1 or 21 (because of a new TRIGGER or STATE event, that means when no further PMTR values are to be considered) set time stamp values and duration of increments in the RAM.

28.20.7.5.1 Equation DPLL-1a4 to update the time stamp values for TRIGGER

- TSF_T[s]=TS_Tx **(DPLL-1a4)**

using the following equations for the determination of TS_Tx
For TS0_HRT=0:

- TS_Tx=TS_T **(DPLL-1a4w)**
- DT_T_ax = DT_T_ACT **(DPLL-1a4a)**

For TS0_HRT=1:

- TS_Tx(20:0)=TS_T/8 **(DPLL-1a4x)**
- TS_Tx(23:21)=TBU_TS0_T(23:21) **(DPLL-1a4y)**
  for TBU_TS0_T(20:0) > or = TS_Tx(20:0)
- TS_Tx(23:21)=TBU_TS0_T(23:21) -1 **(DPLL-1a4z)**
  for TBU_TS0_T(20:0) < TS_Tx(20:0)
- DT_T_ax = DT_T_ACT/8 **(DPLL-1a4b)**

Note: the combination of values LOW_RES=0 and TS0_HRT=1 is not possible.

Store the time stamp values in the time stamp field according to the address pointer APT_2B=s, but make this update only after the calculations of actions (**Section 28.20.7**) because the old TSF_T[i] values are still needed for these calculations. Please note that the address pointer after a gap is still incremented by SYN_T_OLD in that case (see state machine step 1 in **Section 28.20.8.6**).

28.20.7.5.2 Equation DPLL-1a5-7 to extend the time stamp values for TRIGGER in forward direction

when SYT=1 and SYN_T_OLD=r>1 and DIR1=0

- TSF_T[s-1] = TSF_T[s] -DT_T_ax **(DPLL-1a5)**
- TSF_T[s-2] = TSF_T[s-1] -DT_T_ax **(DPLL-1a6)**

until

- TSF_T[s- r+1] = TSF_T[s- r+2] -DT_T_ax **(DPLL-1a7)**

after the incrementation of the pointer APT_2B by SYN_T_OLD

28.20.7.5.3 Equations DPLL-1a5-7 for backward direction

when SYT=1 and SYN_T_OLD=r>1 and DIR1=0

- TSF_T[s+1] = TSF_T[s] -DT_T_ax **(DPLL-1a5)**
- TSF_T[s+2] = TSF_T[s+1] -DT_T_ax **(DPLL-1a6)**

until

- TSF_T[s- r+1] = TSF_T[s- r+2] -DT_T_ax **(DPLL-1a7)**

after the incrementation of the pointer APT_2B by SYN_T_OLD

28.20.7.5.4 Equations DPLL-1b1 and DPLL-1c1 to update the RAM after calculation

- DT_T[p] = DT_T_ACT **(DPLL-1b1)**

save old reciprocal value from RAM before overwriting:

- RDT_T_FS1= RDT_T[p] **(DPLL-1c1)**

after that store new value in RAM

- RDT_T[p]= RDT_T_ACT **(DPLL-1c2)**

Store increment duration and reciprocal value in RAM region 2 in normal mode after calculation of actions only when a new valid TRIGGER slope is detected and in emergency mode directly after calculation of DT_T_ACT or RDT_T_ACT respectively.

28.20.7.5.5 Equation DPLL-6a4 to update the time stamp values for STATE

- TSF_S[s]=TS_Sx **(DPLL-6a4)**

using the following equations for the determination of TS_Sx

For TS0_HRS=0:

- TS_Sx=TS_S **(DPLL-6a4)**
- DT_S_ax = DT_S_ACT **(DPLL-6a4a)**

For TS0_HRS=1:

- TS_Sx(20:0)=TS_S/8 **(DPLL-6a4x)**
- TS_Sx(23:21)=TBU_TS0_S(23:21) for TBU_TS0_S(20:0) > or = TS_Sx(20:0) **(DPLL-6a4y)**
- TS_Sx(23:21)=TBU_TS0_S(23:21) -1 for TBU_TS0_S(20:0) < TS_Sx(20:0) **(DPLL-6a4z)**
- DT_S_ax = DT_S_ACT/8 **(DPLL-6a4b)**

Note: the combination of values LOW_RES=0 and TS0_HRS=1 is not possible.

Store the time stamp value in the time stamp field according to the address pointer APS_1C2=s, but make this update only after the calculation of actions (equations DPLL-13a2, **Section 28.20.7.3.2** or DPLL-13a4 **Section 28.20.7.4.2**, if applicable) because the old TSF_S[i] values are still needed for these calculations. Please note, that the address pointer after a gap is still incremented by SYN_S_OLD in that case (see state machine step 21 in **Section 28.20.8.6**).

28.20.7.5.6 Equations DPLL-6a5-7 to extend the time stamp values for STATE

When SYS=1 and SYN_S_OLD=r>1 and DIR2=0 or DIR1=0 respectively calculate

- TSF_S[s-1] = TSF_S[s] - DT_S_ax **(DPLL-6a5)**
- TSF_S[s-2] = TSF_S[s-1] - DT_S_ax **(DPLL-6a6)**

until

- TSF_S[s- r+1] = TSF_S[s- r+2] - DT_S_ax **(DPLL-6a7)**

after incrementation of the pointer APS_2b by SYN_S_OLD

28.20.7.5.7 Equations DPLL-6a5-7 for backward direction

When SYS=1 and SYN_S_OLD=r>1 and DIR2=1 or DIR1=1 respectively calculate

- TSF_S[s+1] = TSF_S[s] - DT_S_ax **(DPLL-6a5)**
- TSF_S[s+2] = TSF_S[s+1] - DT_S_ax **(DPLL-6a6)**

until

- TSF_S[s+r-1] = TSF_S[s+r-2] - DT_S_ax **(DPLL-6a7)**

after the incrementation of the pointer APS_1C2 by SYN_S_OLD

28.20.7.5.8 Equations DPLL-6b1 and DPLL-6c2 to update the RAM after calculation

- DT_S[p] = DT_S_ACT **(DPLL-6b1)**

save old reciprocal value from RAM before overwriting:

- RDT_S_FS1= RDT_S[p] **(DPLL-6c1)**

after that store new value in RAM

- RDT_S[p] = RDT_S_ACT **(DPLL-6c2)**

when a new active STATE slope is detected in emergency mode or in normal mode (SMC=RMO=0) directly after calculation of the values above.

Store increment duration and reciprocal value in RAM region 1c in emergency mode after calculation of actions only when a new active STATE slope is detected and in normal mode directly after calculation of DT_S_ACT or RDT_S_ACT respectively.

28.20.7.6 Time and position stamps for actions in Normal Mode

28.20.7.6.1 Equation DPLL-15 to calculate the action time stamp

- TSAC[i]= DTA[i] - DLA[i] + TS_Tx (for DTA[i] > DLA[i] and DTA[i] - DLA[i] < 0x800000) **(DPLL-15a)**
- TSAC[i]= TS_Tx (for DTA[i] < DLA[i]) **(DPLL-15b)**
- TSAC[i]= 0x7FFFFF + TS_Tx (for DTA[i] > DLA[i] and DTA[i] - DLA[i] > 0x7FFFFF) **(DPLL-15c)**

*Note: For TS_Tx see equations (DPLL-1a4 and following and following), **Section 28.20.7.5.1***

The calculation is done after the calculation of the current expected duration value according to equation DPLL-12 at **Section 28.20.7.2.5**. The time stamp of the action can be calculated as shown above in equation DPLL-15 using the delay value of the action and the current time stamp

28.20.7.6.2 Equations DPLL-17 to calculate the position stamp forwards

for **DIR1**=0 and TS0_HRT=0:

- PSAC[i] = PSA[i] - (DLA[i]*RCDT_TX_NOM)*(MLT+1) **(DPLL-17)**

with

- RCDT_TX_NOM= (1/CDT_TX_NOM) * SYN_T **(DPLL-17a)**

and

- RCDT_TX= 1/CDT_TX **(DPLL-17b)**

for **DIR1**=0 and TS0_HRT=1:

- PSAC[i] = PSA[i] - (8*DLA[i]*RCDT_TX_NOM)*(MLT+1) **(DPLL-17d)**

with

- RCDT_TX_NOM= (1/CDT_TX_NOM) * SYN_T **(DPLL-17a)**

and

- RCDT_TX= 1/CDT_TX **(DPLL-17b)**

replace (MLT+1) in equations (DPLL-17) and (DPLL-17d) by MLS1 for SMC=1
use the calculated value of (DPLL-17b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_D_i

The action is to be updated for each new *TRIGGER* event until the calculated time stamp is in the past.
In this case the values of TSAC[i] and PSAC[i] depend on the DPLL_CTRL_11.ACBU signal. When DPLL_CTRL_11.ACBU = '0': Use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.
When DPLL_CTRL_11.ACBU = '1':
- for ACB[z][1]='1':
is used as input signal to control if "action in the past" shall be checked based on position information. If the position has reached "past" use the calculated position stamp of the actual increment as target position value.
- for ACB[z][1]='0':
In this case the PSAC[i] is used as calculated by the DPLL.
- for ACB[z][0]='1': is used as input signal to control if "action in past" shall be checked based on time information. If the time has reached "past" use the time stamp of the last input event instead of the calculated TSAC[i] value.
- for ACB[z][0]='0':
In this case the TSAC[i] is used as calculated by the DPLL.

Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

28.20.7.6.3 Equations DPLL-17 to calculate the position stamp backwards

For **DIR1**=1 and TS0_HRT=0:

- PSAC[i] = PSA[i] + (DLA[i]*RCDT_TX_NOM)*(MLT+1) **(DPLL-17c)**

with

- RCDT_TX_NOM= (1/CDT_TX_NOM) * SYN_T **(DPLL-17a)**

and

- RCDT_TX= 1/CDT_TX **(DPLL-17b)**

For **DIR1**=1 and TS0_HRT=1:

- PSAC[i] = PSA[i] + (8*DLA[i]*RCDT_TX_NOM)*(MLT+1) **(DPLL-17e)**

with

- RCDT_TX_NOM= (1/CDT_TX_NOM) * SYN_T **(DPLL-17a)**

and

- RCDT_TX= 1/CDT_TX **(DPLL-17b)**

replace (MLT+1) in equations (DPLL-17c) and (DPLL-17e) by MLS1 for SMC=1
use the calculated value of (DPLL-17b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_D_i

The action is to be updated for each new *TRIGGER* event until the calculated time stamp is in the past.
Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.
In this case the values of TSAC[i] and PSAC[i] depend on the DPLL_CTRL_11.ACBU signal. When DPLL_CTRL_11.ACBU = '0': Use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.
When DPLL_CTRL_11.ACBU = '1':
- for ACB[z][1]='1':
is used as input signal to control if "action in the past" shall be checked based on position information. If the position has reached "past" use the calculated position stamp of the actual increment as target position value.
- for ACB[z][1]='0':
In this case the PSAC[i] is used as calculated by the DPLL.
- for ACB[z][0]='1': is used as input signal to control if "action in past" shall be checked based on time information. If the time has reached "past" use the time stamp of the last input event instead of the calculated TSAC[i] value.
- for ACB[z][0]='0':
In this case the TSAC[i] is used as calculated by the DPLL.

Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

28.20.7.7 The use of the RAM

The RAM is used to store the data of the last FULL_SCALE period. The use of single port RAMs is recommended. The data width of the RAM is usual 3 bytes, but could be extended to 4 bytes in future applications. There are 3 different RAMs, each with separate access ports. The RAM 1a is used to store the position minus time requests, got from the ARU. No CPU access is possible to this RAM during operation (when the DPLL is enabled).

Ram 1b is used for configuration parameters and variables needed for calculations. Within RAM 1c the values of the *STATE* events are stored. RAM 1b and RAM 1c do have a common access port and are also marked as RAM 1bc in order to clarify this fact.

RAM 2 is used for values of the *TRIGGER* events.

Because of the access of the DPLL internal state machine at the one side and the CPU at the other side the access priority has to be controlled for both RAMs 1bc and 2. The access priority is defined as stated below. The CPU access procedure via AE-interface goes in a wait state (waiting for data valid) while it needs a colliding RAM access during serving a corresponding state machine RAM access. In order not to provoke unexpected behavior of the algorithms the writing of the CPU to the RAM regions 1b, 1c or 2 will be monitored and results in interrupt requests when enabled.

CPU access is specified at follows:

1. CPU has highest priority for a single read/write access. The DPLL algorithm is stalled during external bus RAM accesses.
2. After serving the CPU access to the RAM the DPLL gets the highest RAM access priority for 8 clock cycles. Afterwards continue with 1.

The RAM address space has to be implemented in the address space of the CPU.

28.20.7.8 Time and position stamps for actions in Emergency Mode

28.20.7.8.1 Equation DPLL-18 to calculate the action time stamp

- TSAC[i]= DTA[i] - DLA[i] + TS_Sx (for DTA[i] > DLA[i] and DTA[i] - DLA[i] < 0x800000) **(DPLL-18a)**
- TSAC[i]= TS_Sx (for DTA[i] < DLA[i]) **(DPLL-18b)**
- TSAC[i]= 0x7FFFFF + TS_Sx (for DTA[i] > DLA[i] and DTA[i] - DLA[i] > 0x7FFFFF) **(DPLL-18c)**

Note: For TS_Sx see equations (DPLL-6a4 and following), ***Section 28.20.7.5.5***

The calculation is done after the calculation of the current expected duration value according to equation DPLL-14 at **Section 28.20.7.3.5**. The time stamp of the action can be calculated as shown in equation DPLL-18 using the delay value of the action and the current time stamp.

28.20.7.8.2 Equations DPLL-20 to calculate the position stamp forwards

for **DIR2**=0 or DIR1=0 respectively and TS0_HRS=0:

- PSAC[i] = PSA[i] - (DLA[i]*RCDT_SX_NOM)*MLS1 **(DPLL-20)**

with

- RCDT_SX_NOM= (1/CDT_SX_NOM) * SYN_S **(DPLL-20a)**

and

- RCDT_SX= 1/CDT_SX **(DPLL-20b)**

for **DIR2**=0 or DIR1=0 respectively and TS0_HRS=1:

- PSAC[i] = PSA[i] - (8*DLA[i]*RCDT_SX_NOM)*MLS1 **(DPLL-20d)**

with

- RCDT_SX_NOM= (1/CDT_SX_NOM) * SYN_S **(DPLL-20a)**

and

- RCDT_SX= 1/CDT_SX **(DPLL-20b)**

replace MLS1 in equations (DPLL-20) and (DPLL-20d) by MLS2 for (SMC=1 and RMO=1)
use the calculated value of (DPLL-17b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_Di.

The action is to be updated for each new *STATE* event until the calculated time stamp is in the past.
In this case the values of TSAC[i] and PSAC[i] depend on the DPLL_CTRL_11.ACBU signal.

When DPLL_CTRL_11.ACBU = ‘0’: Use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.
When DPLL_CTRL_11.ACBU = ‘1’:
- for ACB[z][1]=’1’:
is used as input signal to control if “action in the past” shall be checked based on position information. If the position has reached “past” use the calculated position stamp of the actual increment as target position value.
- for ACB[z][1]=’0’:
In this case the PSAC[i] is used as calculated by the DPLL.
- for ACB[z][0]=’1’: is used as input signal to control if “action in past” shall be checked based on time information. If the time has reached “past” use the time stamp of the last input event instead of the calculated TSAC[i] value.
- for ACB[z][0]=’0’:
In this case the TSAC[i] is used as calculated by the DPLL.
the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.
Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

se the calculated value of (DPLL-17b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_D_i.

The action is to be updated for each new *STATE* event until the calculated time stamp is in the past.
In this case the values of TSAC[i] and PSAC[i] depend on the DPLL_CTRL_11.ACBU signal.

When DPLL_CTRL_11.ACBU = ‘0’: Use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.
When DPLL_CTRL_11.ACBU = ‘1’:
- for ACB[z][1]=’1’:
is used as input signal to control if “action in the past” shall be checked based on position information. If the position has reached “past” use the calculated position stamp of the actual increment as target position value.
- for ACB[z][1]=’0’:
In this case the PSAC[i] is used as calculated by the DPLL.
- for ACB[z][0]=’1’: is used as input signal to control if “action in past” shall be checked based on time information. If the time has reached “past” use the time stamp of the last input event instead of the calculated TSAC[i] value.
- for ACB[z][0]=’0’:
In this case the TSAC[i] is used as calculated by the DPLL.
the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value.
Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

28.20.7.8.3 Equations DPLL-20 to calculate the position stamp backwards

For **DIR2**=1 or DIR1=1 respectively and TS0_HRS=0:

- PSAC[i] = PSA[i] + (DLA[i]*RCDT_SX_NOM)*MLS1 **(DPLL-20c)**

with

- RCDT_SX_NOM= (1/CDT_SX_NOM) * SYN_S **(DPLL-20a)**

and

- RCDT_SX= 1/CDT_SX **(DPLL-20b)**

For **DIR2**=1 or DIR1=1 respectively and TS0_HRS=1:

- PSAC[i] = PSA[i] + (8*DLA[i]*RCDT_SX_NOM)*MLS1 **(DPLL-20e)**

with

- RCDT_SX_NOM= (1/CDT_SX_NOM) * SYN_S **(DPLL-20a)**

and

- RCDT_SX= 1/CDT_SX **(DPLL-20b)**

replace MLS1 in equations (DPLL-20c) and (DPLL-20e) by MLS2 for (SMC=1 and RMO=1)
use the calculated value of (DPLL-20b) also for the generation of SUB_INCi and serve the action by transmission of TSAC[i] and PSAC[i] to ACT_D.

The action is to be updated for each new *STATE* event until the event is in the past. In this case use the time stamp of the last input event instead of the calculated value and the calculated position stamp of the actual increment as target position value. Set the corresponding shadow bit in the DPLL_ACT_STA register. Because of the blocking read operation the ACT_D values can be read only once.

28.20.8 Signal processing

28.20.8.1 Time stamp processing

Signal processing does mean the computation of the time stamps in order to calculate at which time the outputs have to appear. For such purposes the time stamp values have to be stored in the RAM and by calculating the difference between old and new values the duration of the last time interval is determined simply. This difference should be also stored in the RAM in order to see the changes between the intervals by changing the conditions and the speed of the observed process.

28.20.8.2 Count and compare unit

The count and compare unit processes all input signals taking into account the configuration values. It uses a state machine and provides the output signals as described above.

28.20.8.3 Sub pulse generation for SMC=0

28.20.8.3.1 Adder for generation of SUB_INCx by the carry c_{out}

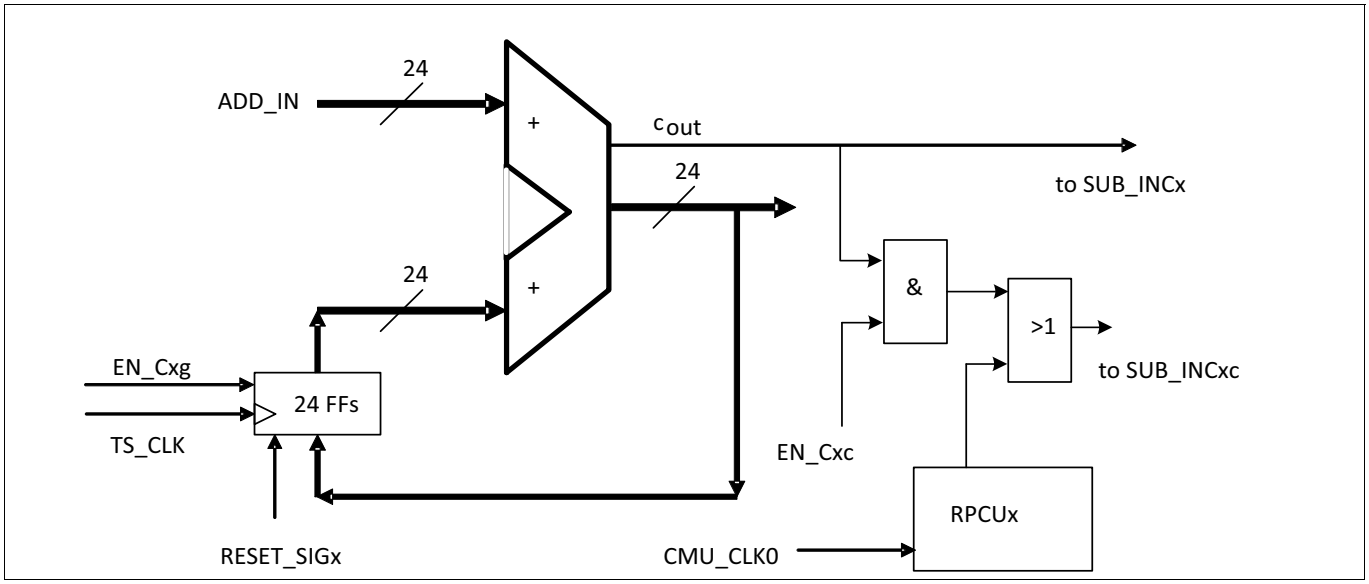


Figure 128 Adder for generation of SUB_INCx by the carry cout

Note: The SUB_INC generation by the circuit above has the advantage, that the resolution for higher speed values is better as for a simple down counter.

After RESET and after EN_Cxg=0 the flip-flops (FF s) should have a zero value. EN_Cxg has to be zero until reliable ADD_IN values are available and the pulse generation starts. This is controlled by the configuration bits SGE1,2 in the DPLL_CONTROL_1 register. The calculated values for the increment prediction using equations DPLL-2c **Chapter 28.20.6.2.7**, DPLL-2c1 **Chapter 28.20.6.4.4**, DPLL-7c **Chapter 28.20.6.3.7** or DPLL-7c1 **Chapter 28.20.6.4.4** respectively are valid only when at least NUTE>1 TRIGGER values or at least NUSE>1 STATE values are available. For NUTE =1 or NUSE=1 respectively the equations DPLL-25 **Chapter 28.20.8.3.4** and DPLL-26 **Chapter 28.20.8.3.6** use the actual increment value subtracted by the weighted average error.

The generation of SUB_INC1 pulses depends on the configuration of the DPLL. In automatic end mode the counter INC_CNT1 resets the enable signal EN_C1 when the number of pulses desired is reached. In this case only the uncompensated output SUB_INC1 remains active in order to provide pulses for the input filter unit. In the case of acceleration missing pulses can be determined at the next TRIGGER/STATE event in normal/emergency mode easily. For the correction strategy COA = 0 those missing pulses are sent out with CMU_CLK0 frequency as soon they are determined. During this time period the EN_Cxg remains cleared. After calculation or providing of a new ADD_IN value the FFs are enabled by EN_Cxg. In this way no pulse is lost. The new pulses are sent out afterwards, when INC_CNT1 is set to the desired value, maybe by adding MLT+1 or MLS1 respectively for the new TRIGGER/STATE event.

Because the used DIV procedure of the algorithms results only in integer values, a systematic failure could appear. The pulse generation at SUB_INC1 will stop in automatic end mode when the INC_CNT1 register reaches zero or all remaining pulses at a new increment will be considered in the next calculation. In this way the loss of pulses can be avoided. When a new TRIGGER/STATE appears the value of SYN_T*(MLT+1) or SYN_S*MLS1 respectively is added to INC_CNT1, when SGE1=1. Therefore for FULL_SCALE 2*(TNU+1)*(MLT+1) pulses SUB_INC1 generated, when INC_CNT1 reaches the zero value. The generation of SUB_INC1 pulses has to be done as fast as possible. The calculations for the ADD_IN value must be done first. Therefore all values needed for calculation are to be fetched in a forecast.

28.20.8.3.2 Equation DPLL-21 to calculate the number of pulses to be sent in normal mode using the automatic end mode condition

For RMO=0, SMC=0 and DMO=0

- NMB_T = (MLT+1)*SYN_T + MP + PD_store + MPVAL1 **(DPLL-21)**

with

PD_store = ADT_T[12:0], prefetched during last increment
SYN_T = ADT_T[18:16], prefetched during last increment
MPVAL1 = pulse correction value for PCM1_SHADOW_TRIGGER=1

while the value for PD_store is zero for AMT=0
and
the value of MP is zero for COA=0

In order to get a higher resolution for higher speed a generator for the sub-pulses is chosen using an adder. All missing pulses MP are considered using equation DPLL-21 and are determined by counting the number of pulses of the last increment. The value SYN_T is stored from the last increment using NT of the ADT_T[i] value at RAM region 2c.

28.20.8.3.3 Equations DPLL-22-24 to calculate the number of pulses to be sent in emergency mode using the automatic end mode condition for SMC=0

For RMO=1, SMC=0 and DMO=0; the value for PD_S_store is zero for AMS=0

- NMB_S = (MLS1 + PD_S_store) *SYN_S+ MP **(DPLL-22)**

with

- MLS1= (MLT+1) * (TNU+1) / (SNU+1) **(DPLL-23)**

PD_S_store = ADT_S[15:0], prefetched during last increment
SYN_S = ADT_S[21:16], prefetched during last increment
MPVAL1 = pulse correction value for PCM1_SHADOW_STATE=1

while the value for PD_S_store is zero for AMS=0
and
the value of MP is zero for COA=0

Please note, that these calculations above in equations DPLL-21 and DPLL-22 are only valid for an automatic end mode (DMO = 0).
For calculation of the number of generated pulses a value of 0.5 is added as shown in equations DPLL-25 or DPLL-26 respectively in order to compensate rounding down errors at the succeeding arithmetic operations. Because in automatic end mode the number of pulses is limited by **INC_CNT1** it is guaranteed, that not more pulses as needed are generated and in the same way missing pulses are caught up for the next increment.

28.20.8.3.4 Equation DPLL-25 to calculate ADD_IN in normal mode for SMC=0

In normal mode (for RMO=0) calculate in the case LOW_RES=TS0_HRT

- ADD_IN_CALN= (NMB_T+0.5) *RCDT_TX **(DPLL-25)**

with

RCDT_TX is the 2^{32} time value of the quotient in equation DPLL-17b, see **Section 28.20.7.6.3**

In normal mode (for RMO=0) calculate in the case LOW_RES=1 and TS0_HRT=0

- ADD_IN_CALN= (NMB_T+0.5) *(RCDT_TX /8) **(DPLL-25a)**

with

RCDT_TX is the 2^{32} time value of the quotient in equation DPLL-17b, see **Section 28.20.7.6.3**

For RMO=0 and SMC=0:

- ADD_IN_CAL1 = ADD_IN_CALN **(DPLL-25b)**

LOW_RES=0 and TS0_HRT=1 is not possible. For such a configuration the RCT bit in the DPLL_STATUS register is set together with the ERR bit.

In the automatic end mode (DMO=0) missing pulses should be sent to the input RPCUx (rapid pulse catch up on) in **Figure 128**, to be caught up on with CMU_CLK0 (for COA=0).

When normal and rapid pulses are generated simultaneously, the SUB_INCx frequency is doubled at this moment in order to count two pulses at the TBU_CHx_BASE register. In order to make the frequency doubling possible, the CMU_CLK0 should be having a frequency which does not exceed half the frequency of TS_CLK. In addition the ADD_IN value should never exceed the value 0x800000. This limitation is only necessary for DMO=0 and COA=0 (see DPLL_CTRL_1 register).

For the normal mode replace ADD_IN of the ADDER (see **Figure 128**) by ADD_IN_CAL1 (when calculated, DLM=0) or ADD_IN_LD1 (when provided by the CPU, DLM=1).

The sub-pulse generation in this case is done by the following calculations using a 24 bit adder with a carry out c_{out} and the following inputs:

- ADD_IN
- the second input is the output of the adder, stored one time stamp clock before

In order not to complicate the calculation procedure use a Multiplier with a sufficient bit width at the output and use the corresponding shifted output bits.

28.20.8.3.5 Enabling of the compensated output for pulses

The c_{out} of the adder influences directly the *SUB_INC1* output of the DPLL (see **Figure 128**). The compensated output SUB_INCxc is in automatic end mode only enabled by EN_Cxc when **INC_CNTx** >0.

28.20.8.3.6 Equation DPLL-26 to calculate ADD_IN in emergency mode for SMC=0

In emergency mode (RMO=1) calculate in the case LOW_RES=TS0_HRS

- ADD_IN_CALE= (NMB_S+0.5)* RCDT_SX **(DPLL-26)**

while

RCDT_SX is the 2^{32} time value of the quotient in equation DPLL-20b, see **Section 28.20.7.8.2**

In emergency mode (RMO=1) calculate in the case LOW_RES=1 and TS0_HRS=0

- ADD_IN_CALE= (NMB_S+0.5)* RCDT_SX /8 **(DPLL-26a)**

while

RCDT_SX is the 2^{32} time value of the quotient in equation DPLL-20b, see **Section 28.20.7.8.2**

For RMO=1 and SMC=0:

- ADD_IN_CAL1 = ADD_IN_CALE **(DPLL-26b)**

LOW_RES=0 and TS0_HRS=1 is not possible. For such a configuration the RCS bit in the DPLL_STATUS register is set together with the ERR bit.

In the automatic end mode (DMO=0) missing pulses should be sent to the input RPCUx (rapid pulse catch up on) in **Figure 128**, to be caught up on with CMU_CLK0 (for COA=0).

When normal and rapid pulses are generated simultaneously, the SUB_INCx frequency is doubled at this moment in order to count two pulses at the TBU_CHx_BASE register. In order to make the frequency doubling possible, the CMU_CLK0 should be having a frequency which does not exceed half the frequency of the system clock. In addition the ADD_IN value should never exceed the value 0x800000 when the TS_CLK frequency exceeds half the frequency of the system clock. This limitation is only necessary for DMO=0 and COA=0 (see DPLL_CTRL_1 register).

For the emergency mode replace ADD_IN of the ADDER (see **Figure 128**) by ADD_IN_CAL1 (when calculated, DLM=0) or ADD_IN_LD1 (when provided by the CPU, DLM=1).

The sub-pulse generation in this case is done by the following calculations using a 24 bit adder with a carry out c_{out} and the following inputs:

- ADD_IN
- the second input is the output of the adder, stored one time stamp clock before

In order not to complicate the calculation procedure use a Multiplier with a sufficient bit width at the output and use the corresponding shifted output bits.

28.20.8.4 Sub pulse generation for SMC=1

28.20.8.4.1 Necessity of two pulse generators

The Adder of picture **Figure 128** must be implemented twice in the case of SMC=1: one for SUB_INC1 controlled by the *TRIGGER* input and (while RMO=1) one for SUB_INC2, controlled by the *STATE* input. In the case described in the chapter above for SMC=0 only one Adder is used to generate SUB_INC1 controlled by the *TRIGGER* in normal mode or by *STATE* in emergency mode.

28.20.8.4.2 Equation DPLL-27 to calculate the number of pulses to be sent for the first device using the automatic end mode condition

For SMC=1 and DMO=0

- NMB_T = (MLS1 + PD_store) *SYN_T + MP + MPVAL1 **(DPLL-27)**

with
PD_store = ADT_T[12:0], prefetched during last increment
SYN_T = ADT_T[18:16], prefetched during last increment
MPVAL1 = pulse correction value for PCM1_SHADOW_TRIGGER=1

while the value for PD_store is zero for AMT=0
and
for COA=0 use zero instead of the value of MP

28.20.8.4.3 Equation DPLL-28 to calculate the number of pulses to be sent for the second device using the automatic end mode condition

for RMO=1, SMC=1 and DMO=0

- NMB_S = MLS2 *SYN_S+ MP + PD_S_store + MPVAL2 **(DPLL-28)**

with
PD_S_store = ADT_S[15:0], prefetched during last increment
SYN_S = ADT_S[21:16], prefetched during last increment
MPVAL2 = pulse correction value for PCM2_SHADOW_STATE=1

while the value for PD_S_store is zero for AMS=0
and
for COA=0 use zero instead of the value of MP

Please note, that these calculations above in equations DPLL-27 and DPLL-28 are only valid for an automatic end mode (DMO = 0). In addition the number of generated pulses is added by 0.5 as shown in equations DPLL-30 or DPLL-31 respectively in order to compensate rounding down errors at the succeeding division operation. Because in automatic end mode the number of pulses is limited by **INC_CNTx** it is guaranteed, that not more pulses as needed are generated and in the same way missing pulses are made up for the next increment.

28.20.8.4.4 Equation DPLL-30 to calculate ADD_IN for the first device for SMC=1

The sub-pulse generation in this case is done by the following calculations using a 24 bit adder with a carry out c_{out} and the following inputs:

- ADD_IN
- the second input is the (delayed) output of the adder, stored with each time stamp clock.

Replace ADD_IN by ADD_IN_CAL1 (when calculated, DLM1=0) or ADD_IN_LD1 (when provided by the CPU, DLM1=1) respectively while:

For SMC=1 and LOW_RES=TS0_HRT

- ADD_IN_CAL1= (NMB_T+0.5) * RCDT_TX **(DPLL-30)**

When RCDT_TX is the 2^{32} time value of the quotient in equation DPLL-17b, see **Section 28.20.7.6.3**

For SMC=1,LOW_RES= 1 and TS0_HRT=0

- ADD_IN_CAL1= (NMB_T+0.5) * (RCDT_TX /8) **(DPLL-30a)**

When RCDT_TX is the 2^{32} time value of the quotient in equation DPLL-17b, see **Section 28.20.7.6.3**

In order not to complicate the calculation procedure use a Multiplier with a sufficient bit width at the output and use the corresponding shifted output bits.

ADD_IN_CAL1 is a 24 bit integer value. The CDT_TX is the expected duration of current *TRIGGER* increment.

The c_{out} of the adder influences directly the *SUB_INC1* output of the DPLL (see **Figure 128**). The SUB_INC1 output is in automatic end mode only enabled by EN_C1 when **INC_CNT1** >0.

28.20.8.4.5 Equation DPLL-31 to calculate ADD_IN for the second device for SMC=1

Replace ADD_IN by ADD_IN_CAL2 (when calculated, DLM2=0) or ADD_IN_LD2 (when provided by the CPU, DLM2=1) respectively while:

for SMC=1, RMO=1 and LOW_RES=TS0_HRS:

- ADD_IN_CAL2= (NMB_S+0.5)* RCDT_SX **(DPLL-31)**

When RCDT_SX is the 2^{32} time value of the quotient in equation DPLL-20b, see **Section 28.20.7.8.2**

for SMC=1, RMO=1, LOW_RES=1 and TS0_HRS=0:

- ADD_IN_CAL2= (NMB_S+0.5)* (RCDT_SX /8) **(DPLL-31a)**

When RCDT_SX is the 2^{32} time value of the quotient in equation DPLL-20b, see **Section 28.20.7.8.2**

In order not to complicate the calculation procedure use a Multiplier with a sufficient bit width at the output and use the corresponding shifted output bits.

The c_{out} of the adder2 influences directly the *SUB_INC2* output of the DPLL (see **Figure 128**).

The SUB_INC2 output is in automatic end mode only enabled by EN_C2 when **INC_CNT2** >0.

Note: Please note, that after RESET and after EN_Cxc=0 (after stopping in automatic end mode) the flip-flops (FFs) have a zero value and also EN_Cxg has to be zero until reliable ADD_IN values are available and the pulse generation starts. The calculated values for the increment prediction using equations DPLL-2c **Section 28.20.6.2.7**, DPLL-2c1 **Section 28.20.6.4.4**, DPLL-7c **Section 28.20.6.3.7** or DPLL-7c1

Section 28.20.6.4.4 respectively are valid only when NUTE>1 or NUSE>1 respectively. For NUTE=1 or NUSE=1 respectively the equations DPLL-30 (see **Section 28.20.8.4.4**) and DPLL-31 (see **Section 28.20.8.4.5**) use the actual increment value subtracted by the weighted average error.

The generation of *SUB_INCx* pulses depends on the configuration of the DPLL.In automatic end mode the counter **INC_CNTx** resets the enable signal EN_Cxcu when the number of pulses desired is reached. In this case only the uncompensated outputs SUB_INCx remain active in order to provide pulses for the input filter units. A new *TRIGGER* or *STATE* input respectively can reset the FFs and also ADD_IN, especially when EN_Cxc was zero before. In the case of acceleration missing pulses can be determined at the next *TRIGGER/STATE* event easily. For the correction strategy COA = 0 those missing pulses are sent out with CMU_CLK0 frequency as soon they are determined. After that the pulse counter **INC_CNTx** should be always zero and the new pulses are sent out afterwards, when **INC_CNTx** is set to the desired value by adding MLS1 or MLS2 for the new *TRIGGER* or *STATE* event respectively.

Because the used DIV procedure of the algorithms results only in integer values, a systematic failure could appear. The pulse generation will stop when the **INC_CNTx** register reaches zero or all remaining pulses at a new increment will be considered in the next calculation. In this way the loss of pulses can be avoided.

When a new *TRIGGER* appears the value of SYN_T*MLS1 is added to **INC_CNT1**. Therefore for FULL_SCALE 2*(TNU+1)*MLS1 pulses *SUB_INC1* generated, when **INC_CNT1** reaches the zero value. The generation of SUB_INC1 pulses has to be done as fast as possible.

When a new *STATE* appears the value of SYN_S*MLS2 is added to **INC_CNT2**. Therefore for FULL_SCALE 2*(SNU+1)*MLS2 pulses *SUB_INC2* generated, when **INC_CNT2** reaches the zero value. The generation of SUB_INC2 pulses has to be done as fast as possible.

28.20.8.5 Calculation of the Accurate Position Values

All appearing *TRIGGER* and *STATE* signals do have a time stamp and a position stamp assigned after the input filter procedure. For the calculation of the exact time stamp the filter values are considered in the calculations of equations DPLL-1a **Section 28.20.6.2.1** or DPLL-6a **Section 28.20.6.3.1** respectively. A corresponding calculation is to be performed for the calculation of position values. The PSTC and PSSC values can be corrected by the CPU, when needed.

The PSTC and PSSC values can be corrected by the CPU, when needed.
After reset, while FTD=0 and no active TRIGGER slope is detected:

- PSTC = 0 **(DPLL-32a)**

Calculate the new Position value for each valid TRIGGER event:

- PSTC= PSTC_old + NMB_T_TAR_OLD **(DPLL-32b)**

when FTD=1 and SGE1=1
with
PSTC_old is the last PSTC value and
NMB_T_old is the number of pulses which are calculated
and provided for sending out in the last increment.

After reset, while FSD=0 and no active STATE slope is detected:

- PSSC= 0 **(DPLL-33a)**

Calculate the new Position value for each STATE event:

- PSSC= PSSC_old + NMB_S_TAR_OLD **(DPLL-33b)**

when FSD=1 and SGE1=1 (SMC=0) or SGE2=1 (SMC=1 respectively with
PSSC_old is the last PSSC value and
NMB_S_old is the number of pulses which are calculated and provided for sending out in the last increment.

28.20.8.6 Scheduling of the Calculation

After enabling the DPLL with each active *TRIGGER* or *STATE* event respectively a cycle of operations is performed to calculate all the results shown in detail in the table below **Section 28.20.2**. A state machine controls this procedure and consists of two parts, the first is triggered by an active slope of the signal *TRIGGER*, begins at step 1 and ends at step 20 (in normal mode and for SMC=1). The second state machine is controlled by an active slope of the signal *STATE*, begins at step 21 and ends at step 40 (in emergency mode and also for SMC=RMO=1). Depending on the mode used all 20 steps are executed or already after 2 steps the jump into the initial state is performed, as shown in the state machine descriptions below. For each new extended cycle (without this jump) all prediction values for actions in the case SMC=0 are calculated once more (with maybe improved accuracy because of better parameters) and all pending decisions are made using these new values when transmitted to the decision device.

In **Section 28.20.8.6.6** the steps of the state machine are described. Please note, that the elaboration of the steps depends on the configuration bits described in the comments. The steps 4 to 17 are only calculated in normal mode (in the state machine explanation below marked yellow in **Section 28.20.2**), but steps 24 to 37 are only calculated in emergency mode (in the state machine explanation below marked cyan in **Section 28.20.2**) when SMC=0.

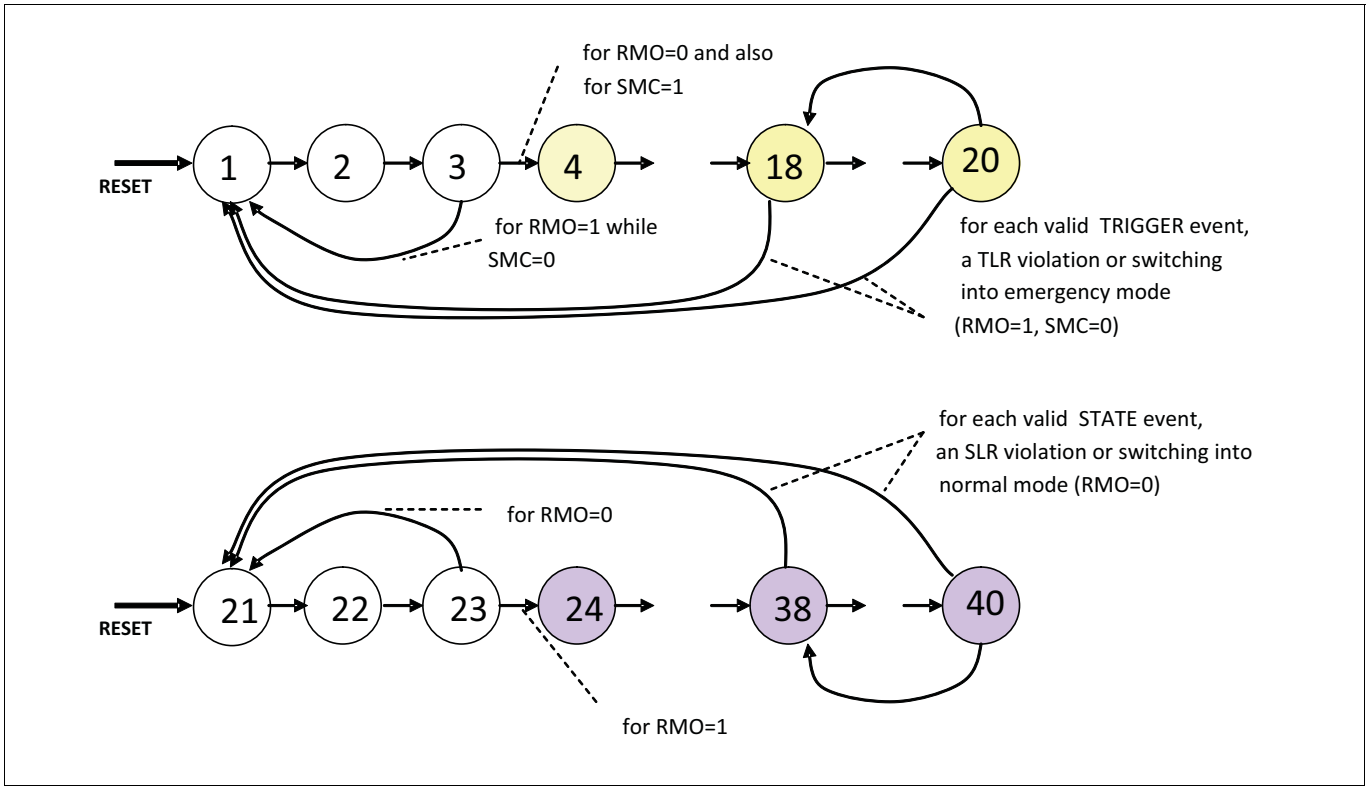


Figure 129 State machine partitioning for normal and emergency mode.

28.20.8.6.1 Synchronization description

TRIGGER

The APT (address pointer for duration and reciprocal duration values of *TRIGGER* increments) is initially set to zero and incremented with each active *TRIGGER* event. Therefore data are stored in the RAM beginning from the first available value. The actual duration of the last increment is stored at DT_T_ACT. For the prediction of the next increment it is assumed, that the same value is valid as long as NUTE is one.

A missing *TRIGGER* is assumed, when at least after TOV* DT_T_ACT no active *TRIGGER* event appears.

The data of equations DPLL-1b1 and DPLL-1c2 **Section 28.20.7.5.4** are written in the corresponding RAM regions and APT is incremented accordingly up to 2*TNU-2*SYN_NT+1.

The APT_2B (address pointer for the time stamp field of *TRIGGER*) is initially set to zero and incremented with each active *TRIGGER* event. When no gap is detected because of the incomplete synchronization process at the beginning, for all *TRIGGER* events the time stamp values are written in the RAM up to 2*(TNU+1) entries, although only 2*(TNU+1-SYN_NT) events in FULL_SCALE appear. When the current position is detected, the synchronization procedure can be performed as described below:

Before the CPU sets the APT_2C address pointer in order to synchronize to the profile, it writes the corresponding increment value for the necessary extension of the RAM region 2b value APT_2B_EXT into the register APT_2B_sync and sets the status bit APT_2C_status. This value can be e.g. 2*SYN_NT, when all gaps in FULL_SCALE already passed the input data stream of *TRIGGER*, or less then this value, when up to now e.g. only a single gap is to be considered in the data stream stored already in the RAM region 2b. The number of virtual increments to be considered depends on the number of inputs already got. After writing APT_2C by the CPU, with the next *TRIGGER* event the APT_2B address pointer is incremented (as usual) and then the additional offset value APT_2B_EXT is added to it once (while APT_2B_STATUS=1 and for forward direction). For that reason the APT_2B_STATUS bit is reset after it. The old APT_2B value before adding the offset is stored in the APT_2B_OLD register as information for the CPU where to start the extension procedure. In the following the CPU fills in the time stamp field around the APT_2B_OLD position taking into account the corresponding number of virtual entries stored in the APT_2B_EXT value and the corresponding NT values in the profile. The extension procedure ends when all gaps considered in the APT_2B_EXT value are treated once. In the consequence all storage locations of RAM region 2b up to now do have the corresponding entries. Future gaps are treated by the DPLL. For a backward direction the APT_2C_ext value is subtracted accordingly.

When the CPU writes the APT_2C address pointer the SYT bit is set simultaneously. For SYT=1 in normal mode (SMC=0) the LOCK1 bit is set with the system clock, when the right number of increments between two synchronization gaps is detected by the DPLL. An unexpected missing *TRIGGER* or an additional *TRIGGER* between two synchronization gaps does reset the LOCK1 bit in normal mode. In that case the CPU must correct the SUB_INC pulse number and maybe correct the APT_2C pointer. For this purpose the LL1I interrupt can be used.

When SYT is set the calculations of equations DPLL-1 to DPLL-5 are performed accordingly and the values are stored in (and distributed to) the right RAM positions.This includes the multiple time stamp storage by the DPLL for a gap according to equations DPLL-1a5 to 7 forwards **Section 28.20.7.5.2** or backwards **Section 28.20.7.5.3**. The APT_2B pointer is for that reason incremented or decremented before this operation considering the virtual increments in addition.

Please note, that for the APT and APT_2C pointers the gap is considered as a single increment.

STATE:

The APS (address pointer for duration and reciprocal duration values of *STATE*) is initially set to zero and incremented with each active *STATE* event. Therefore data are stored in the RAM field beginning at the first

location. The actual duration of the last increment is stored at DT_S_ACT. For the prediction of the next increment it is assumed, that the same value is valid as long as NUSE is one.

A missing *STATE* is assumed, when at least after SOV* DT_S_ACT no active *STATE* event appears.

The data of equations DPLL-6b1 and DPLL-6c2 **Section 28.20.7.5.8** is written in the corresponding RAM regions and APS is incremented accordingly up to 2*SNU-2*SYN_NS+1 (for SYSF=0).

The APS_1C2 (address pointer for the time stamp field of *STATE*) is initially set to zero and incremented with each active *STATE* event. When no gap is detected because of the incomplete synchronization process at the beginning, for all *STATE* events the time stamp values are written in the RAM up to 2*(SNU+1) entries, although (e.g. for SYSF=0) only 2*(SNU+1-SYN_NS) events in FULL_SCALE appear. When the current position is detected, the synchronization procedure can be performed as described below:

Before the CPU sets the APS_1C3 address pointer in order to synchronize to the profile, it writes the corresponding increment value APS_1C2_EXT for the necessary extension of the RAM region 1c2 into the register DPLL_APS_SYNC and sets the APS_1C2_STATUS bit there. This value can be e.g. 2*SYN_NS (for SYSF=0) or SYN_NS (for SYSF=1), when all gaps in FULL_SCALE already passed the input data stream of *STATE*. Also less then this value can be considered, when up to now only a single gap is to be considered in the data stream stored already in the RAM region 1c2. The number of increments to be considered depends on the number of inputs already got. After writing APS_1C3 by the CPU, with the next active *STATE* slope the APS_1C2 address pointer is incremented (as usual) and then the additional offset value APS_1C2_EXT is added to it once (while APS_1C2_STATUS=1 and forward direction). For that reason the APS_1C2_STATUS bit is reset after it. The old APS_1C2 value is stored in the APS_1C2_OLD register as information for the CPU where to start the extension procedure. In the following the CPU extends the time stamp field beginning from the APS_1C2_OLD position taking into account the corresponding number of virtual entries according to the APS_1C2_EXT value and also the correspondent NS values in the profile. The extension procedure ends when all gaps considered in the APS_1C2_EXT value are treated once. In the consequence all storage locations of RAM region 1c2 up to now do have the corresponding entries. Future gaps are treated by the DPLL.

For a backward direction the APS_1C2_EXT value is subtracted accordingly.

When the CPU writes the APS_1C3 address pointer the SYS bit is set simultaneously. For SYS=1 in emergency mode (SMC=0 and DMO=1) the LOCK1 bit is set with the system clock, when the right number of increments between two synchronization gaps is detected by the DPLL. An unexpected missing *STATE* or an additional *STATE* between two synchronization gaps does reset the LOCK1 bit in emergency mode. In that case the CPU must correct the SUB_INC1 pulse number and maybe correct the APS_1C3 pointer. For this purpose the LL1I interrupt can be used.

When SYS is set the calculations of equations DPLL-5 to DPLL-10 are performed accordingly and the values are stored in (and distributed to) the right RAM positions.This includes the multiple time stamp storage by the DPLL for a gap according to equations DPLL-6a5 to 7 forwards **Section 28.20.7.5.6** or backwards **Section 28.20.7.5.7**. The APS_1C2 pointer is for that reason incremented or decremented before this operation considering the virtual increments in addition.Please note, that for the APS and APS_2c pointers the gap is considered as a single increment.

SMC=1:

For SMC=1 it is assumed, that the starting position is known by measuring the characteristic of the device. In this way the APT and APT_2C as well the APS and APS_1C3 values are set properly, maybe with an unknown repetition rate. When no gap is to be considered for *TRIGGER* or *STATE* signals the APT_2B and APS_1C2 address pointers are set equal to APT or APS respectively. It is assumed, that all missing *TRIGGER*s and missing *STATE*s can be also considered from the beginning, when a valid profile with the corresponding adapted values is written in the RAM regions 1c3 and 2c respectively. In that case the TSF_T[i] and TSF_S[i] must be extended by the DPLL according to the profile. Thus the SYT and SYS bits could be set from the beginning and the LOCK1 and LOCK2 bits are set after recognition of the corresponding gaps accordingly. When no gap exists (SYN_NT=0 or SYN_NS=0), the LOCK

bits are set immediately. The CPU can correct the APT_2C and APS_1C3 pointer according to the recognized repetition rate later once more without the loss of Lock1,2.

28.20.8.6.2 Operation for direction change in normal and emergency mode (SMC=0)

When for SMC=0 in normal mode a backwards condition is detected for the TRIGGER input signal (e.g. when THMI is not violated), the LOCK1 bit in the DPLL_STATUS register is reset, the NUTE value in NUTC register is set to 1 (the same for NUSE in NUSC). The address pointers APT_2C as described below (and after that decremented for each following active slope of *TRIGGER* as long as the DIR1 bit shows the backward direction).

Please notice, that in the case of the change of the direction the ITN and ISN bit in the DPLL_STATUS register are reset.

For this transition to the backward direction no change of address pointer APT and APT_2B is necessary.

profile update for TRIGGER when changing direction

The profile address pointer APT_2C is changed step by step in order to update the profile information in SYN_T, SYN_T old and PD_store:

- decrement APT_2C, load SYN_T
- decrement APT_2C, load SYN_T
- decrement APT_2C, load SYN_T, PD_store, update SYN_T_OLD
- decrement APT_2C, **make calculations**, load SYN_T and PD_store, update SYN_T_OLD and PD_store_old and wait for a new *TRIGGER* event

Note: The update of SYN_T_OLD and the loading of PD_store can be performed in all steps above. The value of APT_2B needs not to be corrected. For a direction change from backwards to forwards make the same corrections by incrementing APT_2C.

Make calculations does mean: the operation of the state machine starts with the calculations of NMB_T and INC_CNT1 using the actual APT_2C address pointer value, see **Section 28.20.2**.

The TBU_TB1 value is to be corrected by the number of pulses sent out in the wrong direction mode during the last and current increment. This correction is done by sending out SUB_INC1 pulses for decrementing TBU_TB1 (while DIR1=1).

Save inc_cnt1 value at direction change to inc_cnt1_save.

Calculate the new inc_cnt1 value as follows:

1. Stop sending pulses and save inc_cnt1 at the moment of direction change as inc_cnt1_save.
2. Set inc_cnt1 to the target value of the last increment
   nmb_t_tar_old
3. Add the target number of trigger which were calculated for the current increment when this value was already added to inc_cnt1 before the direction change is detected
   + nmb_t_tar
4. Subtract the value of still not sent pulses (remaining value at inc_cnt1_save)
   - inc_cnt1_save
5. Calculate the new target pulses to be sent considering the new values of SYN_T and PD_store and add them:
   + nmb_t_tar_new

This does mean the following equation:

inc_cnt1 = nmb_t_tar_old + nmb_t_tar
- inc_cnt1_save + nmb_t_tar_new

All pulses summarized at inc_cnt1 are sent out by the maximum possible frequency, because no speed information is available for the first increment after changing the direction. Please notice that no pulse correction using PCM1 of DPLL_CTRL1 is possible during direction change.

When PSTC was incremented/decremented at the active slope and after that the direction change was detected at the same input event, correct **PSTC** once by

- nmb_t_tar_old when changed to backwards

+ nmb_t_tar_old when changed to forwards

in order to compensate the former operation. When the direction information is known before an intended change of PSTC, do not change them.

Store the new calculated value **nmb_t_tar_new** at **nmb_t_tar** for the correct calculation of PSTC at the next input event.

Consequences for STATE

With the next active *STATE* event the direction information is already given. The profile pointer APS_1C3 is to be corrected by a two times decrement in order to point to the profile of the next following increment. In the following it is decremented with each *STATE* event while DIR1=1. The SYN_S and PD_S_store values must be updated accordingly, including SYN_S_OLD and PD_S_store_old.

Because the right direction is already known when an input event appears, make the following corrections:

- decrement APS_1C3, load SYN_S and PD_S_store, update SYN_S_OLD and PD_S_store_old
- decrement APS_1C3, **make calculations**, load SYN_S and PD_S_store, update SYN_S_OLD and PD_S_store_old and wait for a new *STATE* event.

The update of SYN_S_OLD and the loading of PD_S_store can be performed in all steps above. The value of APS_1C2 needs not to be corrected.

When a new *STATE* event occurs, all address pointers are decremented accordingly as long as DIR1=1.

In **emergency mode** the pulses are corrected as follows:

Save inc_cnt1 value at direction change to inc_cnt1_save.

Calculate the new inc_cnt1 value as follows:

1. Stop sending pulses and save inc_cnt1 at the moment of direction change as inc_cnt1_save.
2. Set inc_cnt1 to the target value of the current increment **nmb_s_tar**
   Please notice, that in difference to the normal mode, nmb_s_tar is to be used instead of nmb_s_tar_old, because direction information in emergency mode is only given from the TRIGGER input and occurs of a STATE event independently. That means: The calculations at the last STATE event were done for the correct former direction. In addition still no pulse calculations are performed for the current increment, because the direction change is known at the moment of the recent STATE event.
   Later direction changes are considered at the next STATE event:
3. Do not add the calculated number of state pulses because no new STATE event occurred.
4. Subtract the value of still not sent target pulses (remaining value at inc_cnt1_save)
   - inc_cnt1_save
5. Add the new calculated target pulses for the current increment
   + nmb_s_tar_new

when for the calculation all new conditions of PD_S_store and SYN_S are considered.

inc_cnt1 = nmb_s_tar_old - inc_cnt1_save + nmb_s_tar_new

All pulses summarized at inc_cnt1 are sent out by the maximum possible frequency, because no speed information is available for the first increment after changing the direction. Please notice that no pulse correction using PCM1 of DPLL_CTRL1 is possible during direction change.

Do not change PSSC and suppress incrementing/decrementing of PSSC at the event directly following to the direction change information.

Store the new calculated value **nmb_s_tar_new** at **nmb_s_tar** for the correct calculation of PSTC at the next input event.

Repeated change to forward direction for TRIGGER

The DIR1 bit remains set as long as the THMI value remains none violated for the following *TRIGGER* events and is reset when for an inactive TRIGGER slope the THMI is violated.

Resetting the DIR1 to 0 results (after repeated reset of LOCK1, ITN, ISN) the opposite correction of the profile address pointer considered.

This does mean two increment operations of the address pointer APS_1C3 including the update of SYN_S and PD_S_store with the automatic update of SYN_S_OLD and PD_S_store_old for STATE and four increment operations of the address pointer APT_2C including the update of SYN_T and PD_store with the automatic update of SYN_T_OLD and PD_store_old for TRIGGER.

The correction of TBU_CH1 is done by sending out the correction pulses with the highest possible frequency at SUB_INC1 while DIR1=0. The number of pulses is calculated as shown above.

Consequences for STATE

see corrections above. After that the address pointers are incremented again with each following active *STATE* event as long as DIR1=0.

28.20.8.6.3 Operation for direction change for TRIGGER (SMC=1)

When for SMC=1 a backwards condition is detected for the *TRIGGER* input signal (TDIR=1, resulting in DIR1=1), the LOCK1 bit in the DPLL_STATUS register is reset, the NUTE value in NUTC register is set to 1. The address pointers APT and APT_2C as well as APT_2B are decremented for each active slope of *TRIGGER* as long as the DIR1 bit shows the backward direction.

Please notice, that in the case of the change of the direction the ITN bit in the DPLL_STATUS register is reset.

Profile update for TRIGGER

Make the same update steps for the profile address pointer as shown in **Section 28.20.8.6.2**: Decrement APT_2C for 2 times with the update of the SYN_T and PD_store values at each step with an automatic update of SYN_T_OLD and PD_store_old:

- decrement APT_2C, load SYN_T, PD_store, update SYN_T_OLD
- decrement APT_2C, **make calculations**, load SYN_T and PD_store, update SYN_T_OLD and PD_store_old and wait for a new *TRIGGER* event.

In the normal case no correction of wrong pulses sent is necessary, because the direction change is detected by the pattern immediately.

Nevertheless a correction is necessary as shown below. In the other case: see treatment of pulses TBU_CH1_BASE in normal mode at **Section 28.20.8.6.2**.

Save inc_cntx value at direction change to inc_cnt1_save.

Calculate the new inc_cnt1 value as follows:

1. Clear inc_cnt1.
2. Set inc_cnt1 to the target value of the last increment
   nmb_t_tar
   Please notice, that in difference to the normal mode, nmb_t_tar is to be used instead of nmb_t_tar_old, because the direction information is known before the calculation takes place.
3. Do not add the calculated number of trigger pulses because it is not calculated yet before the direction change information is known.

4. Subtract the value of still not sent pulses (remaining value at inc_cnt1_save)
   - inc_cnt1_save
5. Add the new calculated target pulses for the current increment
   + nmb_t_tar_new

when for the calculation all new conditions of PD_S_store and SYN_S are considered.

inc_cnt1 = nmb_t_tar_old - inc_cnt1_save + nmb_t_tar_new

All pulses summarized at inc_cnt1 are sent out by the maximum possible frequency, because no speed information is available for the first increment after changing the direction. Please notice that no pulse correction using PCM1 of DPLL_CTRL1 is possible during direction change.

Suppress changing of PSTC for the TRIGGER event when a direction change is detected.

Store the new calculated value **nmb_t_tar_new** at **nmb_t_tar** for the correct calculation of PSTC at the next input event.

Repeated change to forward direction for TRIGGER

The DIR1 bit remains set as long as the TDIR bit is set for the following *TRIGGER* events and is reset when for an active *TRIGGER* slope the TDIR is zero.

Resetting the DIR1 to 0 results (after repeated reset of LOCK1 and ITN) the opposite correction of the address pointer use.

This does mean two increment operations of the address pointer including the update of SYN_T and PD_store.

A complex correction of TBU_CH1_BASE and INC_CNT1 is in the normal case not necessary, when all increments are equal (SYN_NT=0) and no adapt information is used. In this case only the MLS1 value is added to INC_CNT1 in order to back count the value for the last increment. In the other case: see treatment of pulses TBU_CH1_BASE and ICN_CNT1 in normal mode at **Section 28.20.8.6.2**.

28.20.8.6.4 Operation for direction change for STATE (SMC=1)

When for SMC=1 a backwards condition is detected for the *STATE* input signal (SDIR=1, resulting in DIR2=1), the LOCK2 bit in the DPLL_STATUS register is reset, the NUSE value in NUSC register is set to 1 and the address pointers APS and APS_1C3_f and APS_1C2 are decremented for each active slope of *STATE* as long as the DIR2 bit shows the backward direction.

Please notice, that in the case of the change of the direction the ISN bit in the DPLL_STATUS register is reset.

For this transition to the backward direction no change of address pointer APS and APS_1C2 is necessary.

Profile update for STATE

Make the same update steps for the profile address pointer as shown in **Section 28.20.8.6.2**: Decrement APS_1C3 for 2 times with the update of the SYN_S, SYN_S_OLD, PD_S_store and PD_S_store_old values at each step:

- decrement APT_1c3, load SYN_S, PD_S_store, update SYN_S_OLD
- decrement APT_1c3, **make calculations**, load SYN_S and PD_S_store, update SYN_S_OLD and PD_S_store_old and wait for a new *STATE* event.

A complex correction of TBU_CH2_BASE and INC_CNT2 is in the normal case not necessary, when all increments are equal (SYN_NS=0) and no adapt information is used. In this case only the MLS2 value is added to INC_CNT2 in order to back count the value for the last increment. In the other case: see treatment of pulses TBU_CH1_BASE and ICN_CNT1 in normal mode at **Section 28.20.8.6.2**.

For the second PMSM the pulses are corrected as follows:

Save inc_cnt2 value at direction change to inc_cnt2_save.

Calculate the new inc_cnt2 value as follows:

1. Clear inc_cnt2.
2. Set inc_cnt2 to the target value of the last increment
   nmb_s_tar
   Please notice, that in difference to the normal mode, nmb_s_tar is to be used instead of nmb_s_tar_old, because no new calculation is performed so far.
3. Do not add the calculated number of state pulses because it is not calculated yet before the direction change information is known.
4. Subtract the value of still not sent pulses (remaining value at inc_cnt2_save)
   - inc_cnt2_save
5. Add the new calculated target pulses for the current increment
   + nmb_s_tar_new

when for the calculation all new conditions of PD_S_store and SYN_S are considered.

inc_cnt2 = nmb_s_tar_old - inc_cnt2_save + nmb_s_tar_new

All pulses summarized at inc_cnt2 are sent out by the maximum possible frequency, because no speed information is available for the first increment after changing the direction. Please notice that no pulse correction using PCM2 of DPLL_CTRL1 is possible during direction change.

Do not change PSSC for a STATE event when a direction change is detected.

Store the new calculated value **nmb_s_tar_new** at **nmb_s_tar** for the correct calculation of PSTC at the next input event.

Repeated change to forward direction for STATE

The DIR2 bit remains set as long as the SDIR bit is set for the following *STATE* events and is reset when for an active *STATE* slope SDIR is zero.

Resetting the DIR2 to 0 results (after repeated reset of LOCK2 and FSD) in the opposite correction of the address pointer use.

After a last decrementing of all address pointers the APS_1C3 is incremented 2 times with a repeated update of SYN_S, SYN_S_OLD and PD_S_store after each increment.

28.20.8.6.5 DPLL reaction in the case of non plausible input signals

When the DPLL is synchronized concerning the *TRIGGER* signal by setting the FTD, SYT and LOCK1 bits in the DPLL_STATUS register, the number of active *TRIGGER* events between the gaps is to be checked continuously.

When additional events appear while a gap is expected, the LOCK1 bit is reset and the ITN bit in the DPLL_STATUS register is set.

When an unexpected gap appears (missing *TRIGGER*S), the NUTE value in the NUTC register is set to 1, the LOCK1 bit is reset and the ITN bit in the DPLL_STATUS register is set. The address pointers are incremented with the next active *TRIGGER* slope accordingly.

The TOR Bit in the DPLL_STATUS register is set, when the time to the next active *TRIGGER* slope exceeds the value of the last nominal *TRIGGER* duration multiplied with the value of the TLR register (see chapter **Section 28.20.12.72**). In this case also the TORI interrupt is generated, when enabled.

When in the following the direction DIR1 changes as described in the chapters above the ITN bit in the DPLL_STATUS register is reset, the use of the address pointers APT_2C is switched and the pulse correction takes place as described above.

In all other cases the CPU can interact to leave the instable state. This can be done by setting the APT_2C address pointer which results in a reset of the ITN bit. In the following NUTE can also be set to higher values.

When the DPLL is synchronized concerning the *STATE* signal by setting the FSD, SYS and LOCK1 (for SMC=0) or LOCK2 (for SMC=1) bits in the DPLL_STATUS register, the number of active *STATE* events between the gaps is to be checked continuously.

When additional events appear while a gap is expected or while an unexpected missing *STATE* event appears, the LOCK1,2 bit is reset and the ISN bit in the DPLL_STATUS register is set.

When an unexpected gap appears for RMO=SMC=1 (missing *STATE*s for synchronous motor control), the NUSE value in the NUSC register is set to 1, the LOCK2 bit is reset and the ISN bit in the DPLL_STATUS register is set. The address pointers are incremented with the next active *STATE* slope accordingly.

When the *STATE* locking range SLR is violated[7)], the state machine 2 will remain in state 21 and the address pointer APS, APS_1C2 and APS_1C3 will remain unchanged until the CPU sets the APS_1C3 accordingly. In this case also the NUSE value in the NUSC register is set to 1. The DPLL stops the generation of the SUB_INC1,2 pulses respectively and will perform no other actions - remaining in step21 of the second state machine (see **Section 28.20.2**).

[7)] The SOR Bit in the DPLL_STATUS register is set, when the time to the next active *STATE* slope exceeds the value of the last nominal *STATE* duration multiplied with the value of the SLR register (see **Section 28.20.12.73**).In this case also the SORI interrupt is generated, when enabled.

When in the following the direction DIR2 changes as described in the chapters above the ISN bit in the DPLL_STATUS register is reset, the use of the address pointers APS_1C3 is switched and the pulse correction takes place as described above. In all other cases the CPU must interact to leave the instable state. This can be done by setting the APS_1C3 address pointers which results in a reset of the ISN bit. In the following NUSE can also be set to higher values.

28.20.8.6.6 State description of the State Machine

Table 72 State description of the State Machine.

Step	Description	Comments

always for DEN=1	**for each inactive TRIGGER slope with TEN=1:**
	check, if the last active TRIGGER slope was passing the PVT check; only in this case perform the following tasks:
	calculate the time stamp difference ΔT to the last active event, store this value at THVAL;
	when THMI >0 is violated (ΔT < THMI):
	generate TINI interrupt,
	set DIR1=0 (forwards)
	set BWD1=0
	(see DPLL_STATUS register)
	else (only for THMI >0):
	set DIR1= 1 (backwards);
	set BWD1=1 (see DPLL_STATUS register)
	after changing the direction correct the pulses WP sent with wrong direction information and send the pulses for the actual increment in addition with highest possible frequency: WP=NMB_T-DPLL_INC_CNT1;
	correct INC_CNT1 by addition of 2*WP before sending the correction pulses;
	generate the TISI Interrupt.
	check THMA, when THMA is violated, generate the TAXI interrupt; go to step 1
	for each inactive STATE slope with SEN=1:
	set DIR2=DIR1
		for SMC=0;
		set DIR1 always after inc./ decr. the address pointers APT, APT_x;
		go to step 1;
		stop output of SUB_INC1 and correct pulses after changing DIR1 after incr./ decr. of APS_x set DIR2 always after incr./decr. the address pointers APS, APS_x;
		go to step 1

always for DEN=1 and (TEN=1 or SEN=1, respectively)	set DIR1=BWD1=TDIR, set DIR2=BWD2=SDIR; for each change of TDIR go to step 1 after performing the following calculations: correct INC_CNT1 correct the pulses (WP, see above) sent with wrong direction information and send the pulses for the actual increment in addition with highest possible frequency. For each change of SDIR go to step 21 after performing the following calculations: update of SYN_S, PD_S_store according to **Section 28.20.8.6.2** correct INC_CNT1,2 correct the pulses sent with wrong direction information and send the pulses for the actual increment in addition with highest possible frequency.	**for SMC=1**; set the direction bits always after incr./decr. the corresponding address pointers;

Table 72 State description of the State Machine. (cont'd)

Step	Description	Comments
1	When DEN = 0 or TEN=0: stay in step 1 until DEN=1, TEN=1 and at least one active *TRIGGER* has been detected (FTD=1); the following steps are performed always (not necessarily in step 1, but also in steps 18 to 20 (when waiting for new PMTR values to be calculated): compare TRIGGER_S with TSL (active slope); **When no active TRIGGER slope appears** and when TS_T_CHECK time is reached: • send missing *TRIGGER* INT, also when a gap is expected according to the profile; set MT=1 (missing *TRIGGER* bit) in the DPLL_STATUS register; do not leave the active step, until a valid active *TRIGGER* appears. **When an active TRIGGER slope appears check PVT** **- when the PVT value is violated:** generate the PWI interrupt, ignore the *TRIGGER* input and wait for the next active TRIGGER slope (ignore each inactive slope); do not store any value **- When the PVT value is fulfilled:** store the actual position stamp at PSTM (value at the TRIGGER event); update the RAM region 2 by equation DPLL-1a-c (see **Section 28.20.7.5**) **store the actual INC_CNT1 value at MP1 as missing pulses***(instead of calculation in step 5)*: store all relevant configuration bits **X** of the DPLL_CTRL(0,1) Registers in **shadow** registers and consider them for all corresponding calculations of steps 2 to 20 accordingly; the relevant bits are explained in the registers itself generate the TASI interrupt; for FTD=0: • set PSTC=PSTM • set FTD (first *TRIGGER* detected) • do not change PSTC,APT, APT_2B • for (RMO=0 or SMC=1) **and SGE1=1**: increment INC_CNT1 by (MLT+1)*) +MPVAL1***) • send SUB_INC1 pulses with highest possible frequency **when SGE1=1** and DPLL_CTRL11_SIP1=0, for SYT=0 and FTD =1: • **dir_crement** APT and APT_2B by one; • **dir_crement** for **SGE1_delay******)=1: PSTC by NMB_T_TAR**) • for (RMO=0 or SMC=1) **and SGE1=1**: increment INC_CNT1 by (MLT+1)*) +MPVAL1***)	Depending on TSL, TEN, DEN step one is leaving with the next *TRIGGER* input; **Note:** Step 1 is also left in emergency mode when an active *TRIGGER* event appears in order to make a switch back to normal mode possible; _old - values are values valid at the last but one active *TRIGGER* event; for the whole table: use always MLS1 instead of (MLT+1) for the case SMC=1; **dir_crement** does mean: increment for DIR1=0 decrement for DIR1=1 *)replace (MLT+1) by MLS1 for SMC=1 **) NMB_T_TAR is the target value of NMB_T of the last increment (see step 5 ff.) ***) add MPVAL1 once to INC_CNT1, that means only when PCM1=1 ****) **SGE1_delay** is the value of SGE1 delayed by one active TRIGGER event *****) PD_store = 0 for AMT=0 (see DPLL_CTRL_0 register)

Table 72 State description of the State Machine. (cont'd)

Step	Description	Comments
1 (cont'd)	for SYT=1 and TOR=0: • **dir_crement** APT, APT_2C, **dir_crement** APT_2B by SYN_T_OLD • **dir_crement** for SGE1_delay[****)]<b>=1 PSTC by NMB_T_TAR[**)] • for (RMO=0 or SMC=1) **and SGE1=1**: increment INC_CNT1 by SYN_T*((MLT+1)[*)]+ PD_store[*****)] + MPVAL1[***)] PD_store is 0 for AMT=0 within the DPLL_STATUS register: • set LOCK1 bit accordingly;	
2	calculate TS_T according to equations DPLL-1a; calculate DT_T_ACT = TS_T - TS_T_OLD calculate RDT_T_ACT calculate QDT_TX according to equation DPLL-2	
3	send CDTI interrupt when NTI_CNT is zero or decrement NTI_CNT when not zero; calculate EDT_T and MEDT_T according to equations DPLL-3 and DPLL-4 for (RMO=1 and SMC=0): update SYN_T, PD_store and go back to step 1	Note: There are different behaviors of RM and HW-IP: For the HW-IP the values of SYN_T and PD_store are not updated until a new active TRIGGER slope occurs.
4	calculate CDT_TX according to equation DPLL-5a and b;	for RMO=0 or SMC=1;

5	calculate missing pulses:
	MP1 = INC_CNT1(at the moment of an active TRIGGER slope)
	
	calculate target pulses:
	NMB_T_TAR = (MLT+1)[*)]*SYN_T + PD_store + MPVAL1
	(instead of PD_store use zero in the case AMT=0)
		for RMO=0 or SMC=1;
		
		[*)]replace (MLT+1) by MLS1 for SMC=1;
		
		add MPVAL1 only for PCM=1 and reset PCM1 after that;

6	sent MP with highest possible frequency and set NMB_T = NMB_T_TAR	for RMO=0 or SMC=1, DMO=0 and COA=0
7	calculate the number of pulses to be sent NMB_T = NMB_T_TAR + MP (see equations DPLL-21 or DPLL-27 respectively)	for RMO=0 or SMC=1, DMO=0 and COA=1
8	NMB_T = SYN_T*CNT_NUM_1	for RMO=0 or SMC=1, DMO=1
9	update SYN_T and PD_store;	Note: There are different behaviors of RM and HW-IP: For the HW-IP the values of SYN_T and PD_store are not updated until a new active TRIGGER slope occurs.

Table 72 State description of the State Machine. (cont'd)

Step	Description	Comments
10	calculate ADD_IN_CAL1 according to equation DPLL-25 and DPLL-25b or DPLL-31 and store this value in RAM use ADD_IN_CAL1 as ADD_IN value for the case DLM=0 use ADD_IN_LD1 as ADD_IN for the case DLM=1, but do this update immediately (without waiting for this step 10); for DMO=DLM=0 and EN_C1u=0: reset the flip-flops in the SUB_INC1 generator; start sending SUB_INC1;	for RMO=0 or SMC=1 for DLM=0 for DLM=1
11	calculate TS_T_CHECK = TS_T + DT_T_ACT *(TOV) ;	for RMO=0 or SMC=1;
12	automatic setting of actions masking bits in the DPLL_STATUS register: for SMC=0: set CAIP1=CAIP2=1 for SMC=1: set only CAIP1=1	steps 12 to 16 are not valid for the combination: (SMC=0 and RMO=1)
13	for all correspondent actions with ACT_N[i]=1 calculate: NA[i] = (PSA[i] - PSTC)/(MLT+1)*)for forward direction with w= integer part and b = remainder of the division (fractional part); for backward direction use NA[i] = (PSTC - PSA[i])/(MLT+1)*) and consider in both cases the time base overflow in order to get a positive difference	actions 0…11 for SMC=1 actions 0…23 for SMC=0 depending on ACT_N[i] in **DPLL_ACT_STA** register; replace MLT+1 by MLS1 for SMC=1
14	calculate PDT_T[i] and DTA[i] for up to 24 action values according to equations DPLL-11 and DPLL-12;	actions 0…11 for SMC=1 actions 0…23 for SMC=0
15	calculate TSAC[i] according to equation DPLL-15 and PSAC[i] according to equation DPLL-17	actions 0…11 for SMC=1 actions 0…23 for SMC=0
16	automatic resetting of actions masking bits in the DPLL_STATUS register: for SMC=0: set CAIP1=CAIP2=0 for SMC=1: set only CAIP1=0; set the corresponding ACT_N[i] bits in the DPLL_ACT_STA register	Set ACT_N[i] for all enabled actions concerned: 0…11 for SMC=1 0…23 for SMC=0
17	check the relation of the last increment to its predecessor according to the profile and taking into account TOV: set the ITN status bit and reset the corresponding LOCK bit, when not plausible; go to step 18, when no active *TRIGGER* appears **for all following steps 18 to 20:go immediately back to step 1, when an active TRIGGER event occurs, interrupt all calculations there and reset all CAIP in that case; when going back to step 1:** store TS_T in RAM 2b according to APT_2B; update RAM 2a and RAM 2d	for all conditions

Table 72 State description of the State Machine. (cont'd)

Step	Description	Comments
18	wait for a new PMTR value; set the corresponding CAIPx values and go to step 19 in that case	go immediately to step 1 and update the RAM according to step 17 when an active *TRIGGER* event occurs
19	make the requested action calculation according to new PMTR values	go immediately to step 1 and update the RAM according to step 17 when an active *TRIGGER* event occurs
20	reset CAIPx and go back to step 18	go immediately to step 1 and update the RAM according to step 17 when an active *TRIGGER* event occurs
21	When DEN = 0 or SEN=0: make sure that the first active slope of STATE is detected; stay in step 1 until DEN=1, SEN=1 and at least one active *STATE* has been detected (FSD=1); the following steps are performed always (not necessarily in step 21, but also in steps 38 to 40 (when waiting for new PMTR values to be calculated): compare STATE_S with SSL (active slope); for each inactive slope: generate a SISI interrupt; • send missing *STATE* INT when TS_S_CHECK time is reached and set MS=1 (missing *STATE* bits) in that case; do not leave step 21 while no active *STATE* appears. **When an active STATE slope appears:** store the actual position stamp at PSSM (value at the STATE event) update RAM by equation DPLL-6a-c (see **Section 28.20.7.5**); **store the actual INC_CNT1/2 at MP1/MP2 respectively as missing pulses** (instead of calculations in step 25) store all relevant configuration bits **X** of the DPLL_CTRL(0,1) Registers in **shadow** registers and consider them for all corresponding calculations of steps 22 to 37 accordingly; the relevant bits are explained in the registers itself for FSD=0: • set PSSC=PSSM • set FSD (first *STATE* detected) • do not increment PSSC • for (RMO=1 and SMC=0) **and SGE1=1**>: increment INC_CNT1 by MLS1+MPVAL1 **) • for (RMO=1 and SMC=1) **nd SGE2=1**: increment INC_CNT2 by MLS2+MPVAL2 **)	Depending on SSL, SEN, DEN step 21 is leaving with the next *STATE* input; for the steps 22-37: for SMC=1 replace: MLS1 by MLS2, LOCK1 by LOCK2; SUB_INC1 by SUB_INC2; CNT_NUM_1 by CNT_NUM_2; MPVAL1 by MPVAL2; EN_C1u by EN_C2u; **dir_crement** does mean: increment for DIR2=0 decrement for DIR2=1 or DIR1 respectively *) target number of pulses of the last increment (see step 25 ff.) **) add MPVAL1 or MPVAL2 only once, that means as long as PCM1 or PCM2 is set respectively ***) **SGE1_delay** is the value of SGE1 delayed by one active STATE event ****) **SGE2_delay** is the value of SGE2 delayed by one active STATE event *****) PD_S_store = 0 for AMS=0 (see DPLL_CTRL_0 register)

Table 72 State description of the State Machine. (cont'd)

Step	Description	Comments
21 cont'd	for SYS=0, FSD =1: • **dir_crement** PSSC by NMB_S_TAR*) for (SMC=0**and SGE1_delay*****)**=1**) or (SMC=1 and **SGE2_delay******)**=1**) • increment INC_CNT1 by MLS1+MPVAL1**) (for SMC=0, **SGE1=1** and RMO=1); • increment INC_CNT2 by MLS2+MPVAL2**)(for SMC=1, **SGE2=1** and RMO=1); • **dir_crement** APS and APS_1C2 for SYS=1: • **dir_crement** APS and APS_1C3 • **dir_crement** APS_1C2 by SYN_S_OLD • for RMO=1 and SMC=0: for **SGE1_delay*****)**=1dir_crement** PSSC by NMB_S_TAR*); for **SGE1=1** increment INC_CNT1 by SYN_S*MLS1 + PD_S_store + MPVAL1**) • for RMO=1 and SMC=1: for**SGE2_delay******)**=1dir_crement** PSSC by NMB_S_TAR*); for **SGE2=1** increment INC_CNT2 by SYN_S*(MLS2 + PD_S_store)*****) +MPVAL2**) • within the DPLL_STATUS register: set LOCK1 or 2 bit accordingly;	
22	calculate TS_S according to equations DPLL-6a; calculate DT_S_ACT = TS_S - TS_S_OLD calculate RDT_S_ACT calculate QDT_SX	
23	send CDSI interrupt; calculate EDT_S and MEDT_S according to equations DPLL-8 and DPLL-9 for RMO=0: go back to step 21 for RMO=0 and update SYN_S and PD_S_store using the current ADT_S[i] values in that case;	Note: There are different behaviors of RM and HW-IP: For the HW-IP the values of SYN_S and PD_S_store are not updated until a new active STATE slope occurs.
24	calculate CDT_SX according to equation DPLL-10a and b;	only for RMO=1
25	calculate missing pulses - for TBU_CH1: MP1 = INC_CNT1(active STATE slope) - for TBU_CH2: MP2 = INC_CNT2(active STATE slope) calculate target number of pulses: NMB_S_TAR = (MLS1 + PD_S_store)*SYN_S + PD_S_store+MPVAL1 (for SMC=0) NMB_S_TAR = MLS2*(SYN_S + PD_S_store) + MPVAL2 (for SMC=1) (instead of PD_S_store use zero in the case AMS=0)	only for RMO=1 for SMC=0 instead of MPVAL1 use zero for PCM1=0f or SMC=1 instead of MPVAL2 use zero for PCM2=0; add MPVAL1/2 once to INC_CNT1/2 and reset PCM1/2 after that

Table 72 State description of the State Machine. (cont'd)

Step	Description	Comments
26	sent MPx with highest possible frequency and set NMB_S = NMB_S_TAR	only for RMO=1, DMO=0 and COA=0
27	calculate number of pulses to be sent according to DPLL-22 or NMB_S = NMB_S_TAR + MPx	only for RMO=1, DMO=0 and COA=1
28	NMB_S = SYN_S*CNT_NUM_1 (SMC=0) NMB_S = SYN_S*CNT_NUM_2 (SMC=1)	only for RMO=1, DMO=1
29	update SYN_S and PD_S_store;	Note: There are different behaviors of RM and HW-IP: For the HW-IP the values of SYN_S and PD_S_store are not updated until a new active STATE slope occurs.
30	calculate ADD_IN_CAL2 according to equation DPLL-26 and DPLL-26b or DPLL-31 respectively and store this value in RAM use ADD_IN_CAL2 as ADD_IN value for the case DLM=0 use ADD_IN_LD2 as ADD_IN for the case DLM=1, but do this update immediately (without waiting for this step 30); for RMO=1, DMO=DLM=0 and EN_C1u=0 (EN_C1u=0): reset the flip-flops in the SUB_INC1 or SUB_INC2 generator respectively; start sending SUB_INC1 / SUB_INC2;	only for RMO=1 for DLM=0 for DLM=1
31	calculate TS_S_CHECK = TS_S + DT_S _ACT * (SOV);	only for RMO=1;
32	automatic setting of actions masking bits in the DPLL_STATUS register: CAIP1 and CAIP2 for SMC=0 only CAIP2 for SMC=1	for RMO=1
33	for all actions with ACT_N[i]=0 calculate: NA[i] = (PSA[i] - PSSC)/MLS1 for forward direction with w = integer part and b = remainder of the division (fractional part) for backward direction use NA[i] = (PSSC - PSA[i])/(MLS1) and consider in both cases the time base overflow in order to get a positive difference use MLS2 as divider in the case of SMC=1	for SMC=0: 24 actions, for SMC=1: 12 actions; depending on ACT_N[i] in **DPLL_ACT_STA** register
34	calculate PDT_S[i] and DTA[i] for up to 24 action values according to equations DPLL-13 and DPLL-14;	only for RMO=1; for SMC=0 actions 0…23 for SMC=1 actions 12…23
35	calculate TSAC[i] according to equation DPLL-18 and PSAC[i] according to equation DPLL-20	for the relevant actions (see above) and RMO=1

Table 72 State description of the State Machine. (cont'd)

Step	Description	Comments
36	automatic reset of the actions masking bit CAIP in the DPLL_STATUS register: CAIP1=CAIP2=0 for SMC=0 and only CAIP2=0 for SMC=1 set the corresponding ACT_N[i] bits in the DPLL_ACT_STA register	for the relevant actions (see above) and RMO=1 Set ACT_N[i] and reset ACT_WRi for all enabled actions
37	check the duration of the last increment to its predecessor according to the profile and taking into account SOV: set the ISN status bit and reset the corresponding LOCK bit, when not plausible; go to step 38, when no active *STATE* appears **for all following steps 38 to 40:** **go immediately back to step 21, when an active STATE event occurs, interrupt all calculations there and reset all CAIPx in that case;** **when going back to step 21:** store TS_S in RAM 1c2 according to APS_1C2; update RAM 1c1 and RAM 1c4	for all conditions
38	wait for a new PMTR value; set the corresponding CAIPx values and go to step 39 in that case	go immediately to step 21 and update the RAM according to step 37 when an active *STATE* event occurs
39	make the requested action calculation according to new PMTR values	go immediately to step 21 and update the RAM according to step 37 when an active *STATE* event occurs
40	reset CAIP and go back to step 38	go immediately to step 21 and update the RAM according to step 37 when an active *STATE* event occurs

28.20.9 DPLL Interrupt signals

The DPLL provides 27 interrupt lines. These interrupts are shown below.

Table 73 DPLL Interrupt signals

Signal	Description
DPLL_DCGI_IRQ	Direction change
DPLL_SORI_IRQ	STATE is out of range
DPLL_TORI_IRQ	TRIGGER is out of range
DPLL_CDSI_IRQ	STATE duration calculated for last increment
DPLL_CDTI_IRQ	TRIGGER duration calculated for last increment
DPLL_TE4_IRQ	TRIGGER event interrupt 4 request3)
DPLL_TE3_IRQ	TRIGGER event interrupt 3 request3)

Table 73 DPLL Interrupt signals (cont'd)

Signal	Description
DPLL_TE2_IRQ	TRIGGER event interrupt 2 request3)
DPLL_TE1_IRQ	TRIGGER event interrupt 1 request3)
DPLL_TE0_IRQ	TRIGGER event interrupt 0 request3)
DPLL_LL2_IRQ	Loss of lock interrupt for SUB_INC2 request
DPLL_GL2_IRQ	Get of lock interrupt for SUB_INC2 request
DPLL_E_IRQ	Error interrupt request
DPLL_LL1_IRQ	Loss of lock interrupt for SUB_INC1 request
DPLL_GL1_IRQ	Get of lock interrupt for SUB_INC1 request
DPLL_W1_IRQ	Write access to RAM region 1b or 1c interrupt request
DPLL_W2_IRQ	Write access to RAM region 2 interrupt request
DPLL_PW_IRQ	Plausibility window violation interrupt of TRIGGER request
DPLL_TAS_IRQ	TRIGGER active slope while NTI_CNT is zero interrupt request
DPLL_SAS_IRQ	STATE active slope interrupt request
DPLL_MT_IRQ	Missing TRIGGER interrupt request
DPLL_MS_IRQ	Missing STATE interrupt request
DPLL_TIS_IRQ	TRIGGER inactive slope interrupt request
DPLL_SIS_IRQ	STATE inactive slope interrupt request
DPLL_TAX_IRQ	TRIGGER maximum hold time violation interrupt request
DPLL_TIN_IRQ	TRIGGER minimum hold time violation interrupt request
DPLL_PE_IRQ	DPLL enable interrupt request
DPLL_PD_IRQ	DPLL disable interrupt request

Note: TEi_IRQ depends on the TINT value in ADT_T[i]1) and is only active when SYT2) =1.

1) see RAM region 2 explanations; see **Section 28.20.14**

2) see DPLL STATUS register; see **Register DPLL_STATUS**

3) see TINT value in the corresponding ADT_T[i] section of RAM region 2; see **Section 28.20.14.3**

28.20.10 MCS to DPLL interface

A reduced AEI interface is implemented in the DPLL, which can only be accessed by the MCS Bus Master interface in the same cluster. The purpose of this interface is to enable a faster interchange of data between the MCS and the DPLL, while enabling a certain control over the DPLL internal state machine.

28.20.10.1Architecture and organization

The implemented interface has an address width of 4 bits, while the size of the data interface is 24 bits.

The following table shows the implemented AEI addresses from the MCS side. Label RD refers to the label used for the address when reading from the MCS or writing from the DPLL, whereas Label WR refers to the label used for the address when writing from the MCS or reading from the DPLL.

28.20.10.2General functionality

In order to have a better understanding of the implications when this interface is used, the following working concepts are informally defined here. They refer to the STATE engine operation when DPLL_CTRL_11.STATE_EXT is set.

Update of ram:

Operation which stores TSF_S, DT_S_ACTUAL and RDT_S_ACTUAL back to the RAM and reads the profile.

Calculation of sub-increments:

Calculation of DT_S_ACTUAL, RDT_S_ACTUAL, NMB_S and ADD_IN.

Change of direction:

Update of profile and its increment or decrement (only in the STATE processor).

Calculation of actions (PMT):

Where the calculation of PSAC and TSAC is performed (only in state processor).

If **DPLL_CTRL_11.STATE_EXT is not set**, the DPLL will ignore the data written to this interface from the MCS. The DPLL will not update the interface either and a read done to this interface from the MCS can obtain out-of-date information.

If **DPLL_CTRL_11.STATE_EXT is set**, some modifications are done to the way that the DPLL module works when using the STATE engine.

Up to 128 STATE events can be handled.

RAM1c is not used anymore. Instead, the data needed to perform each of the already described operations is fetched from registers in this interface. The data that would have to be written back to RAM1c is also written to this interface.

In each of the procedures described above, the DPLL will enter in one or more stalled states, in which it will wait for one or more words to be written to MCS2DPLL_DEB15 (STATUS_INFO)

Table 74 Correspondence between STA_S values and their unlocking keywords

Operation	STA_S value	First Keyword	Second Keyword
Update of ram	0b00001_001	0xE	0x1
Calculation of sub-increments	0b00010_000	0xD	0x2
Calculation of actions	0b01110_000	0xC	0x3
Change of direction	0b00000_100	0xB	0x4
Change of direction	0b00000_110	0xB	0x4

The stalled STATE in the DPLL is freed by writing the upper keywords to MCS2DPLL_DEB15. Please note the requirements on DPLL level (described in **Section 28.20.15**) regarding the data on the interface that has to be written by the MCS program. If this data is incorrectly delivered or the STATE state machine is unlocked before delivery, the proper signal processing of the DPLL cannot be assured.

For the particular case of an update of ram after a virtual increment, the data field TSF_S is not calculated completely by the DPLL STATE processing unit. Instead, the values needed in order to fill this data field are provided (**Section 28.20.7.5.6** and **Section 28.20.7.5.7**)

28.20.10.3MCS to DPLL Register overview

The following registers of the MCS2DPLL interface are exclusively accessible by the MCS instance of cluster 0 and cannot be accessed directly via CPU.

Table 75 MCS to DPLL Register overview

Address offset	Common Label	Label RD	Label WR	see Page
0x0	MCS2DPLL_DEB0	DT_S_P	DT_S_P1	650
0x4	MCS2DPLL_DEB1	not used	RDT_S_P1	650
0x8	MCS2DPLL_DEB2	TS_SX	RDT_S_PQ1	651
0xC	MCS2DPLL_DEB3	DT_SX	DT_S_PQ	652
0x10	MCS2DPLL_DEB4	SYN_S_OLD	RDT_S_PQ	652
0x14	MCS2DPLL_DEB5	M_DW	DT_S_PQ1	653
0x18	MCS2DPLL_DEB6	not used	ADT_S_P	654
0x1C	MCS2DPLL_DEB7	RDT_S_P_RD	S_P_RD	655
0x20	MCS2DPLL_DEB8	not used	TSF_S_P	655
0x24	MCS2DPLL_DEB9	not used	TSF_S_P_MQ	656
0x28	MCS2DPLL_DEB10	not used	TSF_S_P_PM_MQ	657
0x2C	MCS2DPLL_DEB11	not used	TSF_S_P_PM	657
0x30	MCS2DPLL_DEB12	not used	ADT_S_P1	658
0x34	MCS2DPLL_DEB13	not used	not used	658
0x38	MCS2DPLL_DEB14	not used	not used	659
0x3C	MCS2DPLL_DEB15	not used	STATUS_INFO	659

28.20.11 DPLL Register Memory overview

The available registers and the size of the RAM area 2 depends on the chosen device.

Please refer to device specific appendix.

28.20.11.1Available DPLL register overview

Table 76 Available DPLL register overview

Register name	Description	see Page
DPLL_CTRL_0	DPLL Control Register 0	511
DPLL_CTRL_1	DPLL Control Register 1	514
DPLL_CTRL_2	DPLL Control Register 2 (actions 0-7 enable)	520
DPLL_CTRL_3	DPLL Control Register 3 (actions 8-15 enable)	521
DPLL_CTRL_4	DPLL Control Register 4 (actions 16-23 enable)	522
DPLL_CTRL_5[1)]	DPLL Control Register 5 (actions 24-31 enable)	523
DPLL_ACT_STA	DPLL ACTION Status Register with connected shadow register	524
DPLL_OSW	DPLL Offset and switch old/new address register	525
DPLL_AOSV_2	DPLL Address offset register for APT in RAM region 2	527

Table 76 Available DPLL register overview (cont'd)

Register name	Description	see Page
DPLL_APT	DPLL Actual RAM pointer to RAM regions 2a, b and d	528
DPLL_APS	DPLL Actual RAM pointer to regions 1c1, 1c2 and 1c4	530
DPLL_APT_2C	DPLL Actual RAM pointer to RAM region 2c	531
DPLL_APS_1C3	DPLL Actual RAM pointer to RAM region 1c3	532
DPLL_NUTC	DPLL Number of recent TRIGGER events used for calculations (mod 2*(TNU +1-SYN_NT))	533
DPLL_NUSC	DPLL Number of recent STATE events used for calculations (e.g. mod 2*(SNU +1-SYN_NS) for SYSF=0)	535
DPLL_NTI_CNT	DPLL Number of active TRIGGER events to interrupt	537
DPLL_IRQ_NOTIFY	DPLL Interrupt notification register	538
DPLL_IRQ_EN	DPLL Interrupt enable register	541
DPLL_IRQ_FORCINT	DPLL Interrupt force register	544
DPLL_IRQ_MODE	DPLL Interrupt mode register	546
DPLL_EIRQ_EN	DPLL Error interrupt enable register	546
DPLL_INC_CNT1	DPLL Counter for pulses for TBU_CH1_BASE to be sent in automatic end mode	549
DPLL_INC_CNT2	DPLL Counter for pulses for TBU_CH2_BASE to be sent in automatic end mode when SMC=RMO=1	549
DPLL_APT_SYNC	DPLL old RAM pointer and offset value for TRIGGER	550
DPLL_APS_SYNC	DPLL old RAM pointer and offset value for STATE	551
DPLL_TBU_TS0_T	DPLL TBU_CH0_BASE value at last TRIGGER event	553
DPLL_TBU_TS0_S	DPLL TBU_CH0_BASE value at last STATE event	553
DPLL_ADD_IN_LD1	DPLL direct load input value for SUB_INC1	554
DPLL_ADD_IN_LD2	DPLL direct load input value for SUB_INC2	555
DPLL_STATUS	DPLL Status Register	556
DPLL_ID_PMTR_[z] (z:0…31)[2]	DPLL 9 bit ID information for input signals PMT z 3)	562
DPLL_CTRL_0_SHADOW_TRIGGER	DPLL shadow register of DPLL_CTRL_0	562
DPLL_CTRL_0_SHADOW_STATE	DPLL shadow register of DPLL_CTRL_0	563
DPLL_CTRL_1_SHADOW_TRIGGER	DPLL shadow register of DPLL_CTRL_1	564
DPLL_CTRL_1_SHADOW_STATE	DPLL shadow register of DPLL_CTRL_1	565
DPLL_RAM_INI	DPLL initialization control and status for RAMs	566

1) This register is only available for device 4.

2) The registers DPLL_ID_PMTR 24-31 are not available for all devices.

28.20.11.2RAM Region 1a map description

Table 77 RAM Region 1a map description

Memory name	Description	Details on Page
PSA[i] (i:0…NOAC-1)	Position/Value request for action i	643
DLA[i] (i:0…NOAC-1)	Time to react before PSAi	643
NA[i] (i:0…NOAC-1)	Number of TRIGGER/STATE increments to ACTION i	644
DTA[i] (i:0…NOAC-1)	Calculated relative time to ACTION i	645

[1)] The values PSA24-31, DLA24-31, NA24-31 and DTA24-31 in RAM 1a are not available for all devices. Please refer to device specific appendix.

28.20.11.3RAM Region 1b map description

Table 78 RAM Region 1b map description

Memory name	Description	see Page
TS_T	Actual signal TRIGGER time stamp register TRIGGER_TS	567
TS_T_OLD	Previous signal TRIGGER time stamp register TRIGGER_TS_OLD	567
FTV_T	Actual signal TRIGGER filter value	568
TS_S	Actual signal STATE time stamp register STATE_TS	569
TS_S_OLD	Previous signal STATE time stamp register STATE_TS_OLD	569
FTV_S	Actual signal STATE filter value	570
THMI	TRIGGER hold time min. value	571
THMA	TRIGGER hold time max. value	572
THVAL	measured last pulse time from active to inactive TRIGGER slope	572
TOV	Time out value of TRIGGER, according to the last nominal increment for a missing TRIGGER	573
TOV_S	Time out value of STATE, according to the last nominal increment for a missing STATE	574
ADD_IN_CAL1	calculated ADD_IN value for SUB_INC1 generation	575
ADD_IN_CAL2	calculated ADD_IN value for SUB_INC2 generation	576
MPVAL1	missing pulses to be added/subtracted directly to SUB_INC1 and INC_CNT1 once	577
MPVAL2	missing pulses to be added/subtracted directly to SUB_INC2 and INC_CNT2 once	578
NMB_T_TAR	target number of TRIGGER pulses	579
NMB_T_TAR_OLD	target number of TRIGGER pulses	580
NMB_S_TAR	target number of STATE pulses	581
NMB_S_TAR_OLD	target number of STATE pulses	582
RCDT_TX	reciprocal value of expected increment duration (T)	583

Table 78 RAM Region 1b map description (cont'd)

Memory name	Description	see Page
RCDT_SX	reciprocal value of expected increment duration (S)	583
RCDT_TX_NOM	reciprocal value of the expected nominal increment duration (T)	584
RCDT_SX_NOM	reciprocal value of the expected nominal increment duration (S)	585
RDT_T_ACT	actual reciprocal value of TRIGGER	586
RDT_S_ACT	actual reciprocal value of STATE	586
DT_T_ACT	Duration of last TRIGGER increment	587
DT_S_ACT	Duration of last STATE increment	588
EDT_T	Absolute error of prediction for last TRIGGER increment	589
MEDT_T	Average absolute error of prediction up to the last TRIGGER increment	589
EDT_S	absolute error of prediction for last STATE increment	590
MEDT_S	Average absolute error of prediction up to the last STATE increment	591
CDT_TX	Expected duration of current TRIGGER increment	592
CDT_SX	Expected duration of current STATE increment	592
CDT_TX_NOM	Expected nominal duration of current TRIGGER increment (without consideration of missing events)	593
CDT_SX_NOM	Expected nominal duration of current STATE increment (without consideration of missing events)	594
TLR	TRIGGER locking range value; the TOR bit in the DPLL_STATUS register is set when violated	594
SLR	STATE locking range value; the SOR bit is set when violated	595
PDT_[i] (i:0…NOAC-1)[1)]	predicted time to ACTION i	596
MLS1	Calculated number of sub-pulses between two STATE events (to be set by CPU)	597
MLS2	Calculated number of sub-pulses between two STATE events (to be set by CPU) for the use when SMC=RMO=1	598
CNT_NUM_1	number of sub-pulses of SUB_INC1 in continuous mode, updated by the host only	598
CNT_NUM_2	number of sub-pulses of SUB_INC2 in continuous mode, updated by the host only	599
PVT	Plausibility value of next active TRIGGER slope	600
PSTC	Accurate calculated position stamp of last TRIGGER input;	601
PSSC	Accurate calculated position stamp of last STATE input;	602
PSTM	Measured position stamp at last active TRIGGER input	602
PSTM_OLD	Measured position stamp at last but one active TRIGGER input	602
PSSM	Measured position stamp at last active STATE input	604
PSSM_OLD	Measured position stamp at last but one active STATE input	604
NMB_T	Number of pulses of current increment in normal mode for SUB_INC1 (see equation DPLL-21 or for SMC=1 equation DPLL-27 respectively)	606
NMB_S	Number of pulses of current increment in emergency mod for SUB_INC1 (see equation DPLL-22) or in the case SMC=1 for SUB_INC2 (see equation DPLL-28)	606

1) The values PDT_24 to PDT_31 in RAM1b are not available for all devices.

28.20.11.4RAM Region 1c map description

Table 79 RAM Region 1c map description

Memory name	Description	see Page
RDT_S[i] (i:0…63)	Part of RAM1c1. Reciprocal value of the corresponding successive increment i, for each true nominal increment.	607
TSF_S[i] (i:0…63)	Part of RAM1c2. Time stamp field for state events, for each true nominal increment plus each virtual increment.	608
ADT_S[i] (i:0…63)	Part of RAM1c3. Adapt values for the current STATE increment, for each true nominal increment.	609
DT_S[i] (i:0…63)	Part of RAM1c4. Uncorrected last increment value of STATE for full scale, for each true nominal increment.	610

28.20.11.5Register Region EXT description

Table 80 Register Region EXT description

Register name	Description	see Page
DPLL_TSAC[z] (z:0…NOAC-1)	DPLL calculated action time stamps for action z	611
DPLL_PSAC[z] (z:0…NOAC-1)	DPLL calculated action position stamps for action z	611
DPLL_ACB_[z] (z:0…(NOAC/4)-1)	DPLL control bits for actions ((4*z)…(4*z)+3)	612
DPLL_CTRL_11	DPLL control register	614
DPLL_THVAL2	DPLL immediate THVAL value	622
DPLL_TIDEL	DPLL additional TRIGGER input delay	623
DPLL_SIDEL	DPLL additional STATE input delay	623
DPLL_CTN_MIN	CDT_T_NOM minimum value	624
DPLL_CTN_MAX	CDT_T_NOM maximum value	624
DPLL_CSN_MIN	CDT_S_NOM minimum value	625
DPLL_CSN_MAX	CDT_S_NOM maximum value	625
DPLL_STA	DPLL state machine status information	626
DPLL_INCF1_OFFSET	DPLL ADD_IN_ADDER1 offset for fast pulse generation	630
DPLL_INCF2_OFFSET	DPLL ADD_IN_ADDER2 offset for fast pulse generation	630
DPLL_DT_T_START	DPLL first value of DPLL_DT_T_ACT for the first increment after setting SIP1 from 0 to 1.	631
DPLL_DT_S_START	DPLL first value of DPLL_DT_S_ACT for the first increment after setting SIP2 from 0 to 1.	632
DPLL_STA_MASK	DPLL trigger masks for signals DPLL_STA_T and DPLL_STA_S	632
DPLL_STA_FLAG	DPLL STA_T/S and INC_CNT1/2 flags	633
DPLL_INC_CNT1_MASK	DPLL INC_CNT1 trigger mask	634

Table 80 Register Region EXT description (cont'd)

Register name	Description	see Page
DPLL_INC_CNT2_MASK	DPLL INC_CNT2 trigger mask	635
DPLL_NUSC_EXT1	Extension register number 1 for DPLL_NUSC[1)]	635
DPLL_NUSC_EXT2	Extension register number 2 for DPLL_NUSC[1)]	636
DPLL_APS_EXT	Extension register for DPLL_APS[1)]	637
DPLL_APS_1C3_EXT	Extension register for DPLL_APS_1C3[1)]	639
DPLL_APS_SYNC_EXT	Extension register for DPLL_APS_SYNC[1)]	640
DPLL_CTRL_EXT	Extension register for DPLL_CTRL[1)]	641

1) These registers will return AEI_STATUS = b#10 if DPLL_CTRL_11.STATE_EXT is not set.

28.20.11.6RAM Region 2 map description

Table 81 RAM Region 2 map description

Memory name	Description	see Page
RDT_T[i] (i:0…AOSV_2B/4-1)	Region 2a. Reciprocal value of the corresponding successive increment i, for each true nominal increment.	646
TSF_T[i] (i:0…AOSV_2B/4-1)	Region 2b. Time Stamp Field for TRIGGER event i, for each true nominal increment plus each virtual increment.	647
ADT_T[i] (i:0…AOSV_2B/4-1)	Region 2c. Adapt values for the current TRIGGER increment i, for each true nominal increment.	647
DT_T[i] (i:0…AOSV_2B/4-1)	Region 2d. Uncorrected last increment value of TRIGGER i, for each true nominal increment.	649

Note: For each of the regions, the maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.

The description of registers is beginning at the register DPLL_CTRL_0.

The description of RAM regions is beginning at RAM 1a (see below): Bits 31 down to 24 in each RAM region are not implemented and therefore always read as zero (reserved). Other bits which are declared as reserved are not protected against writing. Reserved address regions are not protected against writing.

The description of the memory region RAM 1a begins with memory element PSA[i]: The RAM region 1a is writable only for DEN=0 (see DPLL_CTRL_1 register).

The description of memory region RAM1b begins with memory element TS_T.

The description of memory region RAM1c begins with memory element RDT_S.

The description of register region EXT begins with the register DPLL_TSAC[z]: This is an extension of the normal register region above in order to allow up to 32 action calculations and later specification modifications.

The description of the memory region RAM 2 begins with memory element RDT_T.

28.20.12 DPLL Register and Memory description

28.20.12.1Register DPLL_CTRL_0

DPLL Control Register 0

DPLL_CTRL_0
DPLL Control Register 0 **(028000$_H$)** **Application Reset Value: 003B BA57$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMO	TEN	SEN	IDT	IDS	AMT	AMS	TNU								
rw	rw	rw	rw	rw	rw	rw					rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SNU					IFP	MLT									
		rw			rw						rw				

Field	Bits	Type	Description
MLT	9:0	rw	**Multiplier for TRIGGER** 1) MLT+1 is number of *SUB_INC1* pulses between two *TRIGGER* events in normal mode (1...1024); *Note: For emergency mode the number of SUB_INC1 pulses between two STATE events is calculated by the CPU using the formula MLS1=(MLT+1)* (TNU+1) / (SNU+1) in order to get the same number of SUB_INC1 pulses for FULL_SCALE. This value is stored in RAM at 0x05C0. Change of MLT by the CPU must result in the corresponding change of MLS1 by the CPU for SMC=0.* *Note: The number of MLT events is the binary value plus 1. The value MLT+1 is replaced by MLS1 in the case of SMC=1 (see DPLL_CTRL_1 register) for all relevant calculations.*
IFP	10	rw	**Input filter position** 1) 2) 3) Value contains position or time related information. 0_B TRIGGER_FT and STATE_FT mean time related values, that means the number of time stamp clocks 1_B TRIGGER_FT and STATE_FT mean position related values, that means the number of SUB_INC1 (or SUB_INC2 in the case SMC=1) pulses, respectively

Field	Bits	Type	Description
SNU	15:11	rw	**STATE number** 4) SNU+1 is number of nominal STATE events in HALF_SCALE (1...32). *Note: The number of nominal STATE events is the decimal value plus 1. This value can only be written when (RMO=0 and SMC=0) or DEN=0. To make sure that this signal is not changed during a mode change, RMO=0 means that the status of RMO=0 must be given before and during writing to the register. Set SSL=00 before changing this value and set RMO=1 only after FULL_SCALE with SSL>0.* *Note: This register can only be written when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set, the signal cannot be written, the read value is zero.*
TNU	24:16	rw	**TRIGGER number** 4) TNU+1 is number of nominal *TRIGGER* events in HALF_SCALE (1...512). *Note: The number of nominal TRIGGER events is the decimal value plus 1.This value can only be written when (RMO=1 and SMC=0) or DEN=0.* To make sure that this signal is not changed during a mode change RMO=0 means that the status of RMO=0 must be given before and during writing to the register. Set TSL=00 before changing this value and set RMO=0 only after FULL_SCALE with TSL>0.
AMS	25	rw	**Adapt mode STATE** 2) Use of adaptation information of STATE. 0_B No adaptation information is used for STATE 1_B Immediate adapting mode; the values for physical deviation PD_S of ADT_S[i] are considered to calculate SUB_INC1 pulses in emergency mode (SMC=0), or SUB_INC2 pulses for SMC=1
AMT	26	rw	**Adapt mode TRIGGER** 1) Use of adaptation information of TRIGGER. 0_B No adaptation information for TRIGGER is used 1_B Immediate adapting mode; the values for physical deviation PD of ADT_T[i] are considered to calculate the SUB_INC1 pulses in normal mode and for SMC=1
IDS	27	rw	**Input delay STATE** 2) Use of input delay information transmitted in FT part of the STATE signal. 0_B Delay information is not used 1_B Up to 24 bits of the FT part contain the delay value of the input signal, concerning the corresponding edge

Field	Bits	Type	Description
IDT	28	rw	**Input delay TRIGGER** 1) Use of input delay information transmitted in FT part of the TRIGGER signal. 0_B Delay information is not used 1_B Up to 24 bits of the FT part contain the delay value of the input signal, concerning the corresponding edge
SEN	29	rw	**STATE enable** 0_B STATE signal is not enabled (no signal considered) 1_B STATE signal is enabled
TEN	30	rw	**TRIGGER enable** 0_B TRIGGER signal is not enabled (no signal considered) 1_B TRIGGER signal is enabled
RMO	31	rw	**Reference mode** 1)2) Selection of the relevant input signal for generation of SUB_INC1. Double synchronous mode for SMC=1: Signal *TRIGGER*is used to generate the *SUB_INC1* signals, and *STATE* is used to generate the *SUB_INC2* signals. *Note: For SMC=0: TRIGGER and STATE are prepared to calculate SUB_INC1. The RMO bit gives a decision only, which of them is used. For changing from normal mode to emergency mode at the following TRIGGER slope (according to the RMO value in the shadow register)[1], the PSSC value is calculated by PSSC = PSSM + correction value (forward direction) or PSSC = PSSM - correction value (backward direction) with the correction value = inc_cnt1 - nmb_t. For changing from emergency mode to normal mode at the following STATE slope (according to the RMO value in the shadow register)[2], the PSTC value is calculated by PSTC = PSTM + correction value (forward direction) or PSTC = PSTM - correction value (backward direction) with the correction value = inc_cnt1 - nmb_s. In case no further TRIGGER or STATE events the CPU has to perform the above corrections.* 0_B Normal mode; the signal TRIGGER is used to generate the SUB_INC1 signals 1_B Emergency mode for SMC=0; signal STATE is used to generate the SUB_INC1 signals

1) stored in an independent shadow register for an active TRIGGER event and for DEN = 1.
2) stored in an independent shadow register for an active STATE event and for DEN = 1. T
3) he time between two active STATE or TRIGGER events must be always greater then 23.4 µs; in addition, the TS_CLK and the resolution must be chosen such that for each nominal increment, the time stamps at the beginning and the end of the increment differ at least in the value of 257.
4) For IFP=1, the time between two active TRIGGER or STATE events must be always greater then 2.34 ms, and the value x of MLT, MLS1 or MLS2 must be chosen such that the number of time stamp pulses between two SUB_INC events must be less than 65536. This is fulfilled when x is greater than 256.

28.20.12.2Register DPLL_CTRL_1

DPLL Control Register 1

DPLL_CTRL_1
DPLL Control Register 1 **(028004$_H$)** **Application Reset Value: B000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSL		SSL		SMC	TS0_HRT	TS0_HRS	SYSF	SWR	LCD	SYN_NT					
rw		rw		rw	rw	rw	rw	rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYN_NS					PCM2	DLM2	SGE2	PCM1	DLM1	SGE1	PIT	COA	IDDS	DEN	DMO
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
DMO	0	rw	**DPLL mode select** 1) 2) 0_B Automatic end mode; if the number of pulses for an increment is reached, no further pulse is generated until the next active TRIGGER/STATE is received; in the case of getting a new active TRIGGER/STATE before the defined number of pulses is reached, the pulse frequency is changed according to the conditions described below (COA) 1_B Continuous mode; in this mode, a difference between the predefined number of pulses and the actual number of generated pulses can influence the pulse frequency by writing a corresponding pulse number into CNT_NUM_1 or CNT_NUM_2, respectively, in RAM region 1b
DEN	1	rw	**DPLL enable** *Note: The bits 31 down to 0 of the DPLL_STATUS register are cleared, when the DPLL is disabled. Some bits of the control registers can be set only when DEN=0. The protected bits in the DPLL_CTRL_1 register cannot be written when simultaneously DEN is set to 1.* 0_B The DPLL is not enabled; disabling the DPLL will result in a reset state of the DPLL_STATUS register, which remains in this state until DEN=1. No DPLL related interrupt will be generated in that case. 1_B The DPLL is enabled

Field	Bits	Type	Description
IDDS	2	rw	**Input direction detection strategy in the case of SMC=0** *Note: This bit can only be written when the DPLL is disabled and fixed to zero, when not needed for an implementation. Independent of the value of IDDS is the direction information for TRIGGER in the case SMC=0 always considered at the moment when the inactive slope appears.* 0_B The input direction is detected comparing the THMI value with the duration between active and inactive slope of TRIGGER 1_B The input direction is detected using TDIR input signal also in the case SMC=0
COA	3	rw	**Correction strategy in automatic end mode (DMO=0)** 1) 2) For SMC=RMO=1: **COA** is used for SUB_INC1 **and** SUB_INC2. 0_B The pulse frequency of the CMU_CLK0 will be used to make up for missing pulses from last increment; the output of the calculated new pulses will start after resetting the FFs in the pulse generation unit. The frequency of CMU_CLK0 should not exceed half the frequency of the system clock (see Adder for generation SUB_INCx by the carry cout with SMC=0). (see also figure Adder for generation of SUB_INCx by the carry cout, **Figure 128**). 1_B Missing pulses of the last increment are distributed evenly to the next increment, calculations are done when the next active input event appears. The number of missing sub-pulses will be determined by the pulse counter difference between the last two active TRIGGER/STATE events, respectively; the FFs in the pulse generation unit are not reset before sending new pulses.
PIT	4	rw	**Plausibility value PVT to next active TRIGGER is time related** 1) 0_B The plausibility value is position related (PVT contains the number of SUB_INC1 pulses) 1_B The plausibility value is time related (the PVT value is to be multiplied with the duration of the last increment DT_T_ACT and divided by 1024)
SGE1	5	rw	**SUB_INC1 generator enable** 1) 2) 0_B The SUB_INC1 generator is not enabled 1_B The SUB_INC1 generator is enabled

Field	Bits	Type	Description
DLM1	6	rw	**Direct Load Mode for SUB_INC1 generation** 1) 2) 0_B The DPLL uses the calculated ADD_IN_CAL value for the SUB_INC1 generation 1_B The ADD_IN_LD value is used for the SUB_INC1 generation and is provided by the CPU; the value remains valid until the CPU writes a new one; the calculated ADD_IN values are stored as ADD_IN_CAL in the RAM at different locations for normal and emergency mode
PCM1	7	rw	**Pulse Correction Mode for SUB_INC1 generation** 1) 2) 3) 0_B The DPLL does not use the correction value stored in MPVAL1 1_B The DPLL uses the correction value stored in MPVAL1 in normal and emergency mode
SGE2	8	rw	**SUB_INC2 generator enable** 2) 0_B The SUB_INC2 generator is not enabled 1_B The SUB_INC2 generator is enabled
DLM2	9	rw	**Direct Load Mode for SUB_INC2 generation** 2) 0_B The DPLL uses the calculated ADD_IN_CAL value for the SUB_INC2 generation 1_B The ADD_IN_LD value is used for the SUB_INC2 generation and is provided by the CPU; the value remains valid until the CPU writes a new one; the calculated ADD_IN values are stored as ADD_IN_CAL in the RAM at different locations for normal and emergency mode
PCM2	10	rw	**Pulse Correction Mode for SUB_INC2 generation** 2) 3) 0_B The DPLL does not use the correction value stored in MPVAL2 1_B The DPLL uses the correction value stored in MPVAL2

Field	Bits	Type	Description
SYN_NS	15:11	rw	**Synchronization number of STATE** Summarized number of virtual increments in HALF_SCALE. Sum of all systematic missing *STATE* events in HALF_SCALE (for SYSF=0) or FULL SCALE (for SYSF=1) ; the SYN_NS missing *STATES* can be divided up to an arbitrary number of blocks. The pattern of events and missing events in FULL_SCALE is shown in RAM region 1c3 as value NS in addition to the adapted values. The number of stored increments in FULL_SCALE must be equal to 2*(SNU+1-SYN_NS) for SYSF=0 or 2*(SNU+1)-SYN_NS for SYSF=1 . This pattern is written by the CPU beginning from a fixed reference point (maybe beginning of the FULL_SCALE region). The relation to the actual increment is established by setting of the profile RAM pointer APS_1C3 in an appropriate relation to the RAM pointer APS of the actual increment by the CPU. This value can only be written when (RMO=0 and SMC=0) or DEN=0. Set SSL=00 before changing this value and set RMO=1 only after FULL_SCALE with SSL>0. To make shure that this signal is not changed during a mode change SMC=0 means that the status of SMC=0 must be given before and during writing to the register. This register can only be written when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set, the signal cannot be written, the read value is zero.
SYN_NT	21:16	rw	**Synchronization numberof TRIGGER** Summarized number of virtual increments in HALF_SCALE. Sum of all systematic missing *TRIGGER* events in HALF_SCALE; the SYN_NT missing *TRIGGER* can be divided up to an arbitrary number of blocks. The pattern of events and missing events in FULL_SCALE is shown in RAM region 2c as value NT in addition to the adapted values. The number of stored increments in FULL_SCALE must be equal to 2*(TNU-SYN_NT). This pattern is written by the CPU beginning from a fixed reference point (maybe beginning of the FULL_SCALE region). The relation to the actual increment is established by setting of the profile RAM pointer APT_2C in an appropriate relation to the RAM pointer APT of the actual increment by the CPU. This value can only be written when (RMO=1 and SMC=0) or DEN=0. Set TSL=00 before changing this value and set RMO=0 only after FULL_SCALE with TSL>0. To make shure that this signal is not changed during a mode change SMC=0 means that the status of SMC=0 must be given before and during writing to the register.
LCD	22	rw	**Locking condition definition** This bit can only be written when the DPLL is disabled and fixed to zero, when not needed for an implementation. 0_B Locking condition definition is one times missing TRIGGERs, as expected by the profile in HALF_SCALE (one gap) 1_B Locking condition definition is n-1 times missing TRIGGERs, as expected by the profile in HALF_SCALE (one additional tooth)

Field	Bits	Type	Description
SWR	23	rw	**Software reset** Resets all register and internal states of the DPLL. Setting the SWR bit results only in a software reset when the DPLL is not enabled (DEN=0). 0_B No software reset enabled 1_B Software reset enabled
SYSF	24	rw	**SYN_NS for FULL_SCALE** The value SYN_NS does mean the sum of all systematic missing *STATE* events in HALF_SCALE (for SYSF=0) or FULL SCALE (for SYSF=1). This value can only be written when (RMO=0 and SMC=0) or DEN=0. Set SSL=00 before changing this value, and set RMO=1 only after FULL_SCALE with SSL>0. To make sure that this signal is not changed during a mode change, SMC=0 means that the status of SMC=0 must be given before and during writing to the register. 0_B The SYN_NS value is valid for HALF_SCALE 1_B The SYN_NS value is valid for FULL_SCALE
TS0_HRS	25	rw	**Time stamp high resolution STATE** This bit can only be written when the DPLL is disabled. 0_B The resolution of the used DPLL input TBU_TS0 bits is equal to the STATE input time stamp resolution 1_B The STATE input time stamps have an 8 times higher resolution than the TBU_TS0 DPLL input
TS0_HRT	26	rw	**Time stamp high resolution TRIGGER** This bit can only be written when the DPLL is disabled. 0_B The resolution of the used DPLL input TBU_TS0 bits is equal to the TRIGGER input time stamp resolution 1_B The TRIGGER input time stamps have an 8 times higher resolution than the TBU_TS0 input
SMC	27	rw	**Synchronous Motor Control** This bit can only be written when the DPLL is disabled. 0_B TRIGGER and STATE inputs are used for a control different to SMC 1_B The TRIGGER input reflects a combined sensor signal for SMC, and in the case of RMO=1, also STATE reflects a different combined sensor signal

Field	Bits	Type	Description
SSL	29:28	rw	**STATE slope select** Definition of active slope for signal STATE; each active slope is an event defined by SNU. Set by DEN=0 only. If DPLL_STATUS.FSD = '1': Slope sensitive after detection of first STATE input signal. If DPLL_STATUS.FSD = '0': Level sensitive for first STATE input signal edge. *Note: This value can only be written when (RMO=0 and SMC=0) or DEN=0. To make sure that this signal is not changed during a mode change, SMC=0 means that the status of SMC=0 must be given before and during writing to the register.* 00_B FSD=1: No slope of STATE will be used (this value makes only sense in normal mode); FSD=0: No input signal of STATE will be used (this value makes only sense in normal mode). 01_B FSD=1: Low-to-high slope will be used as active slope, only inputs with a signal value of '1' will be considered; FSD=0: "High" input signal level will be used as active slope, only inputs with a signal value of '1' will be considered. 10_B FSD=1: High-to-low slope will be used as active slope, only inputs with a signal value of '0' will be considered; FSD=0: "Low" input signal level will be used as active slope, only inputs with a signal value of '0' will be considered. 11_B FSD=1: Both slopes will be used as active slopes; FSD=0: Both input signal levels will be used as active slopes.

Field	Bits	Type	Description
TSL	31:30	rw	**TRIGGER slope select** Definition of active slope for signal TRIGGER each active slope is an event defined by TNU. Set by DEN=0 only. If DPLL_STATUS.FTD = '1' "slope sensitive after detection of first TRIGGER input signal" If DPLL_STATUS.FTD = '0' "level sensitive for first TRIGGER input signal edge" *Note: This value can only be written when (RMO=1 and SMC=0) or DEN=0. To make sure that this signal is not changed during a mode change, SMC=0 means that the status of SMC=0 must be given before and during writing to the register.* 00_B FTD=1: No slope of TRIGGER will be used; this value makes only sense in emergency mode; FTD=0: No input signal of TRIGGER will be used; this value makes only sense in normal mode 01_B FTD=1: Low high slope will be used as active slope, only inputs with a signal value of "1" will be considered; FTD=0: "high" input signal level will be used as active slope, only inputs with a signal value of "1" will be considered 10_B FTD=1: High low slope will be used as active slope, only inputs with a signal value of "0" will be considered; FTD=0: "low" input signal level will be used as active slope, only inputs with a signal value of "0" will be considered 11_B FTD=1: Both slopes will be used as active slopes; FTD=0: Both input signal levels will be used as active slopes

1) Stored in an independent shadow register for a valid TRIGGER event and for DEN = 1.
2) Stored in an independent shadow register for a valid STATE event and for DEN = 1.
3) Bit is cleared, when transmitted to shadow register.

28.20.12.3Register DPLL_CTRL_2

DPLL Control Register 2

DPLL_CTRL_2
DPLL Control Register 2 **(028008_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								WAD7	WAD6	WAD5	WAD4	WAD3	WAD2	WAD1	WAD0
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEN7	AEN6	AEN5	AEN4	AEN3	AEN2	AEN1	AEN0	0							
rw	rw	rw	rw	rw	rw	rw	rw	r							

Field	Bits	Type	Description
AENx (x=0-7)	x+8	rw	**ACTION_x enable** *Note: This bit can be written only if the correspondent WADx bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1).* *Note: For WADi =1, only the corresponding AENi bits are writable. The AENi bits remain unchanged when the corresponding WADi=0.* 0_B The corresponding action is not enabled 1_B The corresponding action is enabled
WADx (x=0-7)	x+16	rw	**Write control bit of Action_x** For WADx =1, only the corresponding AENx bits are writable. The AENx bits remain unchanged when the corresponding WADx=0. 0_B The corresponding AENx bit is not writeable 1_B The corresponding AENx bit is writeable
0	7:0, 31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.4Register DPLL_CTRL_3

DPLL Control Register 3

Note: For all AENi: This bit can be written only if the correspondent WADi Bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1).

Note: For WADi =1 only the corresponding AENi bits are writable. The AENi bits remain unchanged when the corresponding WADi=0.

DPLL_CTRL_3
DPLL Control Register 3 **($02800C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								WAD15	WAD14	WAD13	WAD12	WAD11	WAD10	WAD9	WAD8
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEN15	AEN14	AEN13	AEN12	AEN11	AEN10	AEN9	AEN8	0							
rw	rw	rw	rw	rw	rw	rw	rw	r							

Field	Bits	Type	Description
AENx (x=8-15)	x	rw	**ACTION_x enable** This bit can be written only if the correspondent WADx bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1). 0_B The corresponding action is not enabled 1_B The corresponding action is enabled
WADx (x=8-15)	x+8	rw	**Write control bit of Action_x** For WADx = 1, only the corresponding AENx bits are writable. The AENx bits remain unchanged when the corresponding WADx = 0. 0_B The corresponding AENx bit is not writeable 1_B The corresponding AENx bit is writeable
0	7:0, 31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.5Register DPLL_CTRL_4

DPLL Control Register 4

Note: For all AENi: This bit can be written only if the correspondent WADi Bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1).

Note: For WADi=1, only the corresponding AENi bits are writable. The AENi bits remain unchanged when the corresponding WADi=0.

DPLL_CTRL_4
DPLL Control Register 4 **(028010_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								WAD23	WAD22	WAD21	WAD20	WAD19	WAD18	WAD17	WAD16
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEN23	AEN22	AEN21	AEN20	AEN19	AEN18	AEN17	AEN16	0							
rw	rw	rw	rw	rw	rw	rw	rw	r							

Field	Bits	Type	Description
AENx (x=16-23)	x-8	rw	**ACTION_x enable** This bit can be written only if the correspondent WADx bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1). 0_B The corresponding action is not enabled 1_B The corresponding action is enabled
WADx (x=16-23)	x	rw	**Write control bit of Action_x** For WADx = 1, only the corresponding AENx bits are writable. The AENx bits remain unchanged when the corresponding WADx = 0. 0_B The corresponding AENx bit is not writeable 1_B The corresponding AENx bit is writeable
0	7:0, 31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.6Register DPLL_CTRL_5

DPLL Control Register 5

Note: For all AENi: This bit can be written only if the correspondent WADi bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1).

Note: For WADi=1, only the corresponding AENi bits are writable. The AENi bits remain unchanged when the corresponding WADi=0.

DPLL_CTRL_5
DPLL Control Register 5 **(028014_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								WAD31	WAD30	WAD29	WAD28	WAD27	WAD26	WAD25	WAD24
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AEN31	AEN30	AEN29	AEN28	AEN27	AEN26	AEN25	AEN24	0							
rw	rw	rw	rw	rw	rw	rw	rw	r							

Field	Bits	Type	Description
AENx (x=24-31)	x-16	rw	**ACTION_x enable** This bit can be written only if the correspondent WADx bit is set in the same access. It can be set for debug purposes by CPU also, when DPLL is disabled. The enable bit becomes active only when the DPLL is in operation (DEN=1). 0_B The corresponding action is not enabled 1_B The corresponding action is enabled
WADx (x=24-31)	x-8	rw	**Write control bit of Action_x** For WADx = 1, only the corresponding AENx bits are writable. The AENx bits remain unchanged when the corresponding WADx = 0. 0_B The corresponding AENx bit is not writeable 1_B The corresponding AENx bit is writeable
0	7:0, 31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.7Register DPLL_ACT_STA

DPLL ACTION Status Register with Connected Shadow Register

DPLL_ACT_STA

DPLL ACTION Status Register with Connected Shadow Register(028018_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACT_N															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACT_N															
rw															

Field	Bits	Type	Description
ACT_N	31:0	rw	**New output data values concerning to action i provided** ACT_N[i] is: • Set (for AENi=1 and a new valid PMTR), that means when new action data are to be calculated for the correspondent action. After each calculation of the new actions values, the ACT_N[i] bit updates the corresponding bit in the connected shadow register. The status of the ACT_N[i] bits in the shadow register is reflected by the corresponding DPLL output signal ACT_V (valid bit). • Reset together with the corresponding shadow register bit for AENi=0. • Reset without the corresponding shadow register bit when the calculated event is in the past (the shadow register bit is set, when it was not set before in that case). • The corresponding shadow register bit is reset, when new PMTR data are written or when the provided action data are read (blocking read). • Writeable for debugging purposes together with the corresponding shadow register when DEN=0. *Note: These bits can only be written for test purposes when the DPLL is disabled.* 00000000_H No new output data available after a recent PMT request or actual event value is in the past or invalid 00000001_H New PMTR data received or calculation is to be precised by taking into account new TRIGGER or STATE values

28.20.12.8 Register DPLL_OSW

DPLL Offset and Switch Old/New Address Register

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set any read/write access to this register will return AEI_STATUS = 0b10.

DPLL_OSW

DPLL Offset and Switch Old/New Address Register($02801C_H$) **Application Reset Value: $0000\ 0200_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						OSS		0						SWON_T	SWON_S
r						rw		r						r	r

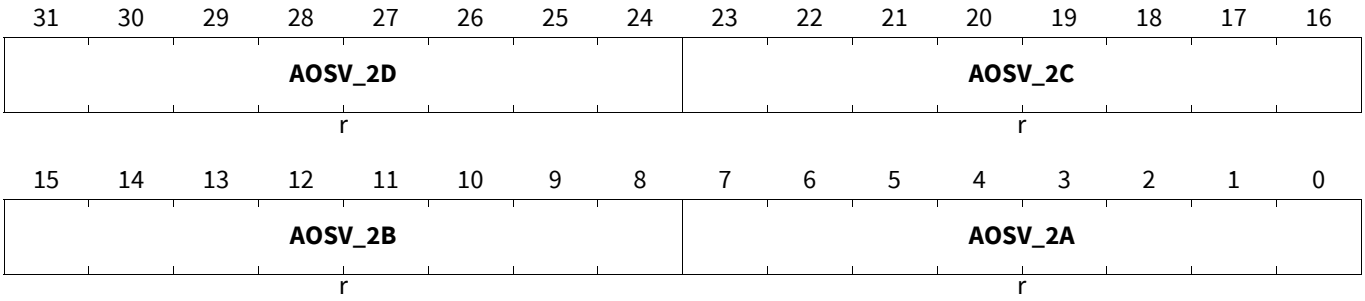
Field	Bits	Type	Description
SWON_S	0	r	**Switch of new STATE** Switch bit for LSB address of STATE. This bit is changed for each write access to TS_S/TS_S_OLD. Using this unchanged address bit SWON_S for any access to TS_S results always in an access to TS_S_OLD. For writing to this address the former old (TS_S_OLD_old) value is overwritten by the new one while the SWON_S bit changes. Thus the former new one is now the old one and the next access is after changing SWON_S directed to this place. Therefore write to TS_S first and after that immediately to FTV_S and PSSM, always before a new TS_S value is to be written. After writing TS_S, FTV_S and PSSM in this order the address pointer AP with LSB(AP)=SWON_S shows for the corresponding address to TS_S_OLD, FTV_S and PSSM while LSB(AP)=/SWON_S results in an access to TS_S, FTV_S_old and PSSM_OLD respectively. The value can be read only. This bit is reset when disabling the DPLL (DEN=0).
SWON_T	1	r	**Switch of new TRIGGER** Switch bit for LSB address of TRIGGER. This bit is changed for each write access to TS_T/TS_T_OLD. Using this unchanged address bit SWON_T for any access to TS_T results always in an access to TS_T_OLD. For writing to this address the former old (TS_T_OLD_old) value is overwritten by the new one while the SWON_T bit changes. Thus the former new one is now the old one and the next access is after changing SWON_T directed to this place. Therefore write to TS_T first and after that immediately to FTV_T and PSTM, always before a new TS_T value is to be written. After writing TS_T, FTV_T and PSTM in this order the address pointer AP with LSB(AP)=SWON_T shows for the corresponding address to TS_T_OLD, FTV_T and PSTM while LSB(AP)=/SWON_T results in an access to TS_T, FTV_T_old and PSTM_OLD respectively. The value can be read only. This bit is reset when disabling the DPLL (DEN=0).
OSS	9:8	rw	**Offset size of RAM region 2** At least 128 and at most 1024 values can be stored in each of the RAM 2 regions a to d accordingly. The value can be set only for DEN=0. The change of the OSS value results in an automatic change of the offset values in the DPLL_AOSV_2 register. This value can only be written when the DPLL is disabled. 00_B Offset size 128 of RAM region 2 01_B Offset size 256 of RAM region 2 10_B Offset size 512 of RAM region 2 11_B Offset size 1024 of RAM region 2
0	7:2, 31:10	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.9Register DPLL_AOSV_2

DPLL Address Offset Register of RAM 2 Regions

DPLL_AOSV_2

DPLL Address Offset Register of RAM 2 Regions(028020_H) **Application Reset Value: $1810\ 0800_H$**



Field	Bits	Type	Description
AOSV_2A	7:0	r	**Address offset value of the RAM 2A region** The value in this field is to be multiplied by 256 (shift left 8 Bits) and added with the start address of the RAM in order to get the start address of RAM region 2a. When the APT value is added to this start address, the current RAM cell RDT_Tx is addressed. Value is set automatically when OSS in the PPLL_OSW register is set: OSS=0x0: AOSV_2A= 0x00 OSS=0x1: AOSV_2A= 0x00 OSS=0x2: AOSV_2A= 0x00 OSS=0x3: AOSV_2A= 0x00
AOSV_2B	15:8	r	**Address offset value of the RAM 2B region** The value in this field is to be multiplied by 256 (shift left 8 Bits) and added with the start address of the RAM in order to get the start address of RAM region 2b. When the APT value is added to this start address, the current RAM cell TSF_Tx is addressed. Value is set automatically when OSS in the PPLL_OSW register is set: OSS=0x0: AOSV_2B= 0x02 OSS=0x1: AOSV_2B= 0x04 OSS=0x2: AOSV_2B= 0x08 OSS=0x3: AOSV_2B= 0x10
AOSV_2C	23:16	r	**Address offset value of the RAM 2C region** The value in this field is to be multiplied by 256 (shift left 8 Bits) and added with the start address of the RAM in order to get the start address of RAM region 2c. When the APT value is added to this start address, the current RAM cell ADT_Tx is addressed. Value is set automatically when OSS in the PPLL_OSW register is set: OSS=0x0: AOSV_2C= 0x04 OSS=0x1: AOSV_2C= 0x08 OSS=0x2: AOSV_2C= 0x10 OSS=0x3: AOSV_2C= 0x20

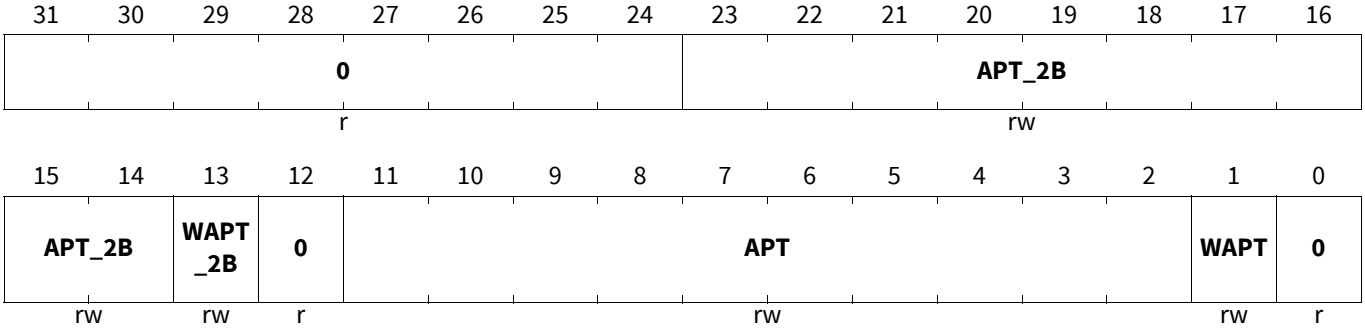
Field	Bits	Type	Description
AOSV_2D	31:24	r	**Address offset value of the RAM 2D region** The value in this field is to be multiplied by 256 (shift left 8 Bits) and added with the start address of the RAM in order to get the start address of RAM region 2d. When the APT value is added to this start address, the current RAM cell DT_Tx is addressed. Value is set automatically when OSS in the PPLL_OSW register is set: OSS=0x0: AOSV_2D= 0x06 OSS=0x1: AOSV_2D= 0x0C OSS=0x2: AOSV_2D= 0x18 OSS=0x3: AOSV_2D= 0x30 The offset values are needed to support a scalable RAM size of region 2 from 1,5 Kbytes to 12 Kbytes. The values above must be in correlation with the offset size defined in the OSW register. All offset values are set automatically in accordance to the OSS value in the DPLL_OSW register. This value can be set only for DEN=0.

28.20.12.10Register DPLL_APT

DPLL Actual RAM Pointer Address for TRIGGER

DPLL_APT

DPLL Actual RAM Pointer Address for TRIGGER (028024_H) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
WAPT	1	rw	**Write bit for address pointer APT** Read as zero. 0_B The APT is not writeable 1_B The APT is writeable

Field	Bits	Type	Description
APT	11:2	rw	**Address pointer TRIGGER** Actual RAM pointer address value offset for DT_T[i] and RDT_T[i] in FULL_SCALE for 2*(TNU+1-SYN_NT) TRIGGER events. This pointer is used for the RAM region 2 subsections 2a and 2d. The pointer APT is incremented for each active *TRIGGER* event (simultaneously with APT_2B, APT_2C) for DIR1=0. For DIR1=1 the APT is decremented. The APT offset value is added in the above shown bit position with the subsection address offset of the corresponding RAM region The APT pointer value is directed to the RAM position, in which the data values are to be written, which corresponds to the last increment. The APT value is not to be changed, when the direction (shown by DIR1) changes, because it points always to a storage place after the considered increment. Changing of DIR1 takes place always after an active *TRIGGER* event and the resulting increment/decrement. This value can only be written when the WAPT bit is set.
WAPT_2B	13	rw	**Write bit for address pointer APT_2B** Read as zero. 0_B The APT_2B is not writeable 1_B The APT_2B is writeable
APT_2B	23:14	rw	**Address pointer TRIGGER for RAM region 2b** Actual RAM pointer address value for TSF_T[i] Actual RAM pointer address of *TRIGGER* events in FULL_SCALE for 2*(TNU+1) *TRIGGER* periods; this pointer is used for the RAM region 2b. The RAM pointer is initially set to zero. For SYT=1: The pointer APT_2B is incremented by SYN_T_OLD for each active *TRIGGER* event (simultaneously with APT and APT_2C) for DIR1=0 when an active *TRIGGER* input appears. For DIR1=1 (backwards) the APT is decremented by SYN_T_OLD. For SYT=0: APT_2B is incremented or decremented by 1. In addition when the APT_2C value is written by the CPU - in order to synchronize the DPLL- with the next active *TRIGGER* event the APT_2B_EXT value is added/subtracted (while APT_2B_STATUS is one; see DPLL_APT_SYNC register at **Section 28.20.12.24**). This value can only be written when the WAPT_2B bit is set.
0	0, 12, 31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.11Register DPLL_APS

DPLL Actual RAM Pointer Address for STATE

DPLL_APS

DPLL Actual RAM Pointer Address for STATE (028028$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												APS_1C2			
r												rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APS_1C2		WAPS_1C2	0					APS						WAPS	0
rw		rw	r					rw						rw	r

Field	Bits	Type	Description
WAPS	1	rw	**Write bit for address pointer APS** Read as zero 0_B The APS is not writeable 1_B The APS is writeable
APS	7:2	rw	**Address pointer STATE** Actual RAM pointer address value for DT_S[i] and RDT_S[i] Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 64 *STATE* events but limited to 2*(SNU+1-SYN_NS) in normal and emergency mode for SYSF=0 or to 2*(SNU+1)-SYN_NS for SYSF=1 respectively; this pointer is used for the RAM region 1c1 and 1c4. APS is incremented (decremented) by one for each active *STATE* event and DIR2=0 DIR2=1). The APS offset value is added in the above shown bit position with the subsection offset of the RAM region. The APS pointer value is directed to the RAM position, in which the data values are to be written, which correspond to the last increment. The APS value is not to be changed, when the direction (shown by DIR2) changes, because it points always to a storage place after the considered increment. Changing of DIR2 takes place always after an active *STATE* event and the resulting increment/decrement. This value can only be written when the WAPS bit is set.
WAPS_1C2	13	rw	**Write bit for address pointer APS_1C2** Read as zero 0_B The APS_1C2 is not writeable 1_B The APS_1C2 is writeable

Field	Bits	Type	Description
APS_1C2	19:14	rw	**Address pointer STATE for RAM region 1c2** Actual RAM pointer address value for TSF_S[i]. Initial value: zero (0x00). Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 64 *STATE* events but limited to 2*(SNU+1) in normal and emergency mode; this pointer is used for the RAM region 1c2. For SYS=1: APS_1C2 is incremented (decremented) by SYN_S_OLD for each active *STATE* event and DIR2=0 (DIR2=1). For SYS=0: APT_1c2 is incremented or decremented by 1 respectively. The APS_1C2 offset value is added in the above shown bit position with the subsection offset of the RAM region. In addition when the APS_1C3 value is written by the CPU - in order to synchronize the DPLL- with the next active *STATE* event the APS_1C2_EXT value is added/subtracted (while APS_1C2_STATUS is one; see DPLL_APT_SYNC register at **Section 28.20.12.25**). This value can only be written when the WAPS_1C2 bit is set
0	0, 12:8, 31:20	r	**Reserved** Read as zero, shall be written as zero.

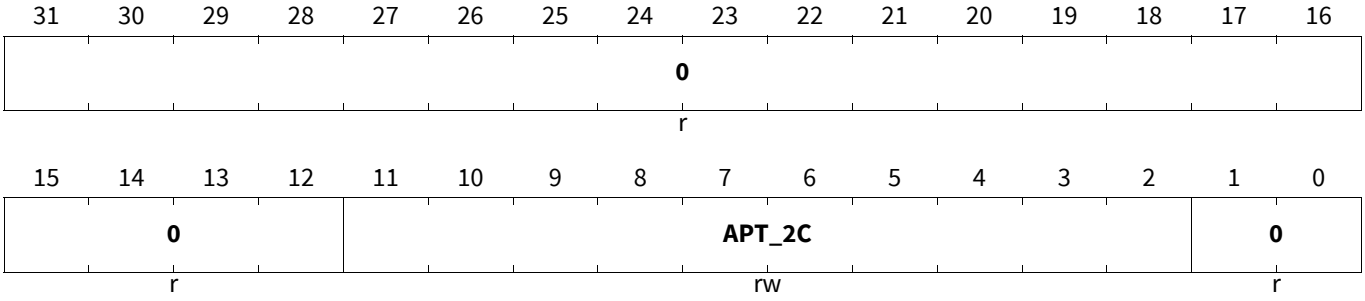
28.20.12.12Register DPLL_APT_2C

DPLL Actual RAM Pointer for Region 2C

Note: The APT_2C pointer values are directed to the RAM position of the profile element in RAM region 2c, which correspond to the current increment. For DIR1=0 (DIR1=1) the pointers APT_2C_x are incremented (decremented) by one simultaneously with APT. For SMC=0 the change of DIR1 takes place always after an active TRIGGER event (by evaluation of the inactive slope) and the resulting increment/decrement. In the case SMC=1 the direction change is known before the input event is processed. The correction of the APT_2C pointer differs: for SMC=0 correct 4 times and for SMC=1 correct only 2 times. The APT_2C_x offset value is added in the above shown bit position with the subsection address offset of the corresponding RAM region.

DPLL_APT_2C

DPLL Actual RAM Pointer for Region 2C (02802C$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
APT_2C	11:2	rw	**Address pointer TRIGGER for RAM region 2c and Actual RAM pointer address value for ADT_T[i]** Actual RAM pointer address value of TRIGGER adapt events in FULL_SCALE for 2*(TNU+1-SYN_NT) TRIGGER periods depending on the size of the used RAM 2; this pointer is used for the RAM region 2 for the subsection 2c only. The RAM pointer is initially set to zero. The APT_2C value is set by the CPU when the synchronization condition was detected. Within the RAM region 2c initially the conditions for synchronization gaps and adapted values are stored by the CPU.
0	1:0, 31:12	r	**Reserved** Read as zero, shall be written as zero.

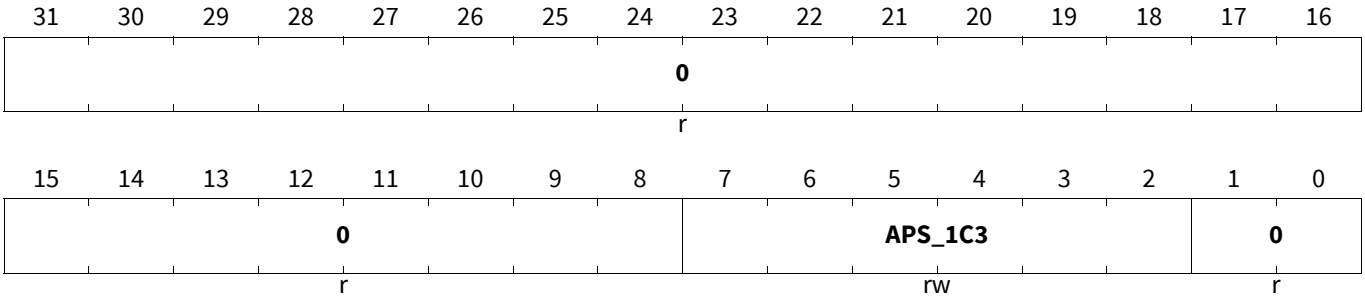
28.20.12.13Register DPLL_APS_1C3

DPLL Actual RAM Pointer for RAM Region 1C3

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set any read/write access to this register will return AEI_STATUS = 0b10.

Note: The APS_1C3 pointer value is directed to the RAM position of the profile element in RAM region 1c2, which corresponds to the current increment. When changing the direction DIR1 or DIR2 respectively, this is always known before an active STATE event is processed. This is because of the pattern recognition in SPE (for PMSM) or because of the direction change recognition by TRIGGER. This direction change results in an automatic increment (forwards) or decrement (backwards) when the input event occurs in addition with a 2 times correction.
The APS_1C3_x offset value is added in the above shown bit position with the subsection address offset of the corresponding RAM region.

DPLL_APS_1C3
DPLL Actual RAM Pointer for RAM Region 1C3 (028030_H) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
APS_1C3	7:2	rw	**Address pointer STATE for RAM region 1c3** Actual RAM pointer address value for ADT_S[i]. Initial value: zero (0x00). Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 64 *STATE* events but limited to 2*(SNU+1-SYN_NS) in normal and emergency mode for SYSF=0 or to 2*(SNU+1)-SYN_NS for SYSF=1 respectively; this pointer is used for the RAM region 1c3. The RAM pointer is set by the CPU accordingly, when the synchronization condition was detected.
0	1:0, 31:8	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.14Register DPLL_NUTC

DPLL Number of Recent TRIGGER Events Used for Calculations

Note: DPLL Number of recent TRIGGER events used for calculations (mod 2(TNU +1-SYN_NT))*.

DPLL_NUTC

DPLL Number of Recent TRIGGER Events Used for Calculations(028034_H) **Application Reset Value: $0001\ 2001_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WVTN	**WSYN**	**WNUT**	**0**				**VTN**						**SYN_T_OLD**		
rw	rw	rw	r				rw						rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYN_T			**0**		**FST**	**NUTE**									
rw			r		rw	rw									

Field	Bits	Type	Description
NUTE	9:0	rw	**Number of recent TRIGGER events used for SUB_INC1 and action calculations modulo 2*(TNUmax+1)** Number of recent TRIGGER events used for SUB_INC1 and action calculations modulo 2*(TNUmax+1). NUTE: number of last nominal increments to be considered for the calculations. No gap is considered in that case for this value, but in the VTN value (see below): This value is set by the CPU, but reset automatically to '1' by a change of direction or loss of LOCK. Each other value can be set by the CPU, maybe Full_SCALE, HALF_SCALE or parts of them. For FULL_SCALE set NUTE= 2*(TNU +1) and for HALF_SCALE NUTE= TNU +1. The relation values QDT_Tx are calculated using NUTE values in the past with its maximum value of 2*(TNU +1). The value zero (in combination with the value FST=1) does mean 2^{11} values in the past. *Note: To prevent that inconsistencies between internal pointer in which NUTE is used and the case decision of different prediction method's for prediction of the next event and PMT(position minus time) occur, the NUTE value is stored internally at that point of time when the internal pointers are calculated for the next event cycle.* *Note: This value can only be written when the WNUT bit is set.* 000_H The NUTE value is less then FULL_SCALE 001_H The NUTE value is equal to FULL_SCALE *This value is set by the CPU, but reset automatically to "0" by a change of direction orloss of LOCK.*
FST	10	rw	**FULL_SCALE of TRIGGER** This value is to be set, when NUTE is set to FULL_SCALE. This value can only be written when the WNUT bit is set.
SYN_T	15:13	rw	**Number of real and virtual events to be considered for the current increment** This value reflects the NT value of the last valid increment, stored in ADT_T[i]; to be updated after all calculations in step 17 of **Table 28.20.8.6.6**. This value can only be written when the WSYN bit in this register is set.
SYN_T_OLD	18:16	rw	**Number of real and virtual events to be considered for the last increment** This value reflects the NT value of the last but one valid increment, stored in ADT_T[i]; is updated automatically when writing SYN_T. This value is updated by the SYN_T value when the WSYN bit in this register is set.

Field	Bits	Type	Description
VTN	24:19	rw	**Virtual TRIGGER number** Number of virtual increments in the current NUTE region This value reflects the number of virtual increments in the current NUTE region; for NUTE=1 this value is zero, when the CPU sets NUTE to a value > 1 , it must also set VTN to the correspondent value; for NUTE is set to FULL_SCALE including NUTE=zero (2^{11} modulo 2^{11}) the VTN is to be set to 2* SYN_NT. The VTN value is subtracted from the NUTE value in order to get the corresponding APT value for the past; the VTN value is not used for the APT_2B pointer. VTN is to be updated by the CPU when a new gap is to be considered for NUTE or a gap is leaving the NUTE region; for this purpose the TINT values in the profile can be used to generate an interrupt for the CPU at the corresponding positions; no further update of VTN is necessary when NUTE is set to FULL_SCALE This value can only be written when the WVTN bit is set.
WNUT	29	rw	**Write control bit for NUTE and FST** Read as zero 0_B The NUTE value is not writeable 1_B The NUTE value is writeable
WSYN	30	rw	**Write control bit for SYN_T and SYN_T_OLD** Read as zero 0_B The SYN_T value is not writeable 1_B The SYN_T value is writeable
WVTN	31	rw	**Write control bit for VTN** Read as zero 0_B The VTN value is not writeable 1_B The VTN value is writeable
0	12:11, 28:25	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.15Register DPLL_NUSC

DPLL Number of Recent STATE Events Used for Calculations

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set any read/write access to this register will return AEI_STATUS = 0b10.

DPLL_NUSC

DPLL Number of Recent STATE Events Used for Calculations(028038_H) Application Reset Value: 0000 2081_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WVSN	WSYN	WNUS	0				VSN						SYN_S_OLD		
rw	rw	rw	r				rw						rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYN_S_OLD			SYN_S						FSS	NUSE					
rw			rw						rw	rw					

Field	Bits	Type	Description
NUSE	5:0	rw	**Number of recent STATE events used for SUB_INCx calculations modulo 2*(SNUmax+1)** No gap is considered in that case for this value, but in the VSN value (see below): This register is set by the CPU but reset automatically to “1” by a change of direction or loss of LOCK. Each other value can be set by the CPU, maybe Full_SCALE, HALF_SCALE or parts of them. The relation values QDT_Sx are calculated using NUSE values in the past with its maximum value of 2*SNU+1. This value can only be written when the WNUS bit is set. *Note: To prevent that inconsistencies between internal pointer in which NUSE is used and the case decision of different prediction method's for prediction of the next event and PMT(position minus time) occur, the NUSE value is stored internally at that point of time when the internal pointers are calculated for the next event cycle.*
FSS	6	rw	**FULL_SCALE of STATE** This value is to be set, when NUSE is set to FULL_SCALE. This value is set by the CPU, but reset automatically to '0' by a change of direction or loss of LOCK. *Note: This value can only be written when the WNUS bit is set.* 0_B The NUSE value is less then FULL_SCALE 1_B The NUSE value is equal to FULL_SCALE
SYN_S	12:7	rw	**Number of real and virtual events to be considered for the current increment** This value reflects the NS value of the last valid increment, stored in ADT_S[i]; to be updated after all calculations in step 37 of **Table 28.20.8.6.6**. This value can only be written when the WSYN bit in this register is set.
SYN_S_OLD	18:13	rw	**Number of real and virtual events to be considered for the last increment** This value reflects the NS value of the last but one valid increment, stored in ADT_S[i]; is updated automatically when writing SYN_S. This value is updated by the SYN_S value when the WSYN bit in this register is set.

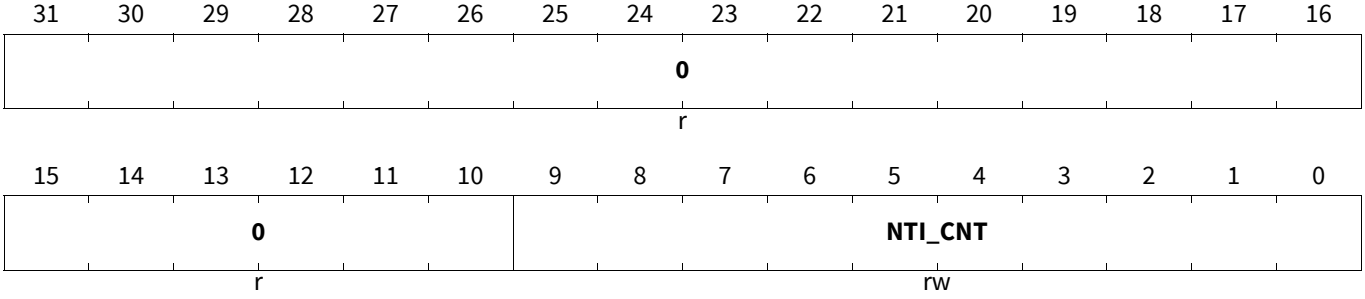
Field	Bits	Type	Description
VSN	24:19	rw	**Virtual STATE number** Number of virtual state increments in the current NUSE region. This value reflects the number of virtual increments in the current NUSE region; for NUSE=1 this value is zero, when the CPU sets NUSE to a value > 1 or zero(2^7 modulo 2^7) , it must also set VSN to the correspondent value; the VSN value is subtracted from the NUSE value in order to get the corresponding APS value for the past; the VSN value is not used for the APS_1C2 pointer. VSN is to be updated by the CPU when a new gap is to be considered for NUSE or a gap is leaving the NUSE region; for this purpose the SASI interrupt can be used; no further update of VSN is necessary when NUSE is set to FULL_SCALE This value can only be written when the WVSN bit is set.
WNUS	29	rw	**Write control bit for NUSE** Read as zero. 0= the NUSE value is not writeable 1= the NUSE value is writeable
WSYN	30	rw	**Write control bit for SYN_S and SYN_S_OLD** Read as zero 0= the SYN_S value is not writeable 1= the SYN_S value is writeable
WVSN	31	rw	**Write control bit for VS** Read as zero 0_B The VSN value is not writeable 1_B The VSN value is writeable
0	28:25	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.16Register DPLL_NTI_CNT

DPLL Number of Active TRIGGER Events to Interrupt

DPLL_NTI_CNT

DPLL Number of Active TRIGGER Events to Interrupt(02803C$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
NTI_CNT	9:0	rw	**Number of TRIGGERs to interrupt** Number of active TRIGGER events to the next DPLL_CDTI interrupt. This value shows the remaining *TRIGGER* events until an active TRIGGER slope results in a DPLL_CDTI interrupt; the value is to be count down for each active *TRIGGER* event.
0	31:10	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.17Register DPLL_IRQ_NOTIFY

DPLL Interrupt Notification Register

Note: All bits in the DPLL_IRQ_NOTIFY register are set permanently until writing a one bit value is performed to the corresponding bit.

DPLL_IRQ_NOTIFY
DPLL Interrupt Notification Register (028040$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				DCGI	SORI	TORI	CDSI	CDTI	TE4I	TE3I	TE2I	TE1I	TE0I	LL2I	GL2I
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EI	LL1I	GL1I	W1I	W2I	PWI	TASI	SASI	MTI	MSI	TISI	SISI	TAXI	TINI	PEI	PDI
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PDI	0	rw	**DPLL disable interrupt; announces the switch off of the DEN bit** This event is combined with the PEI interrupt to the common PDI + PEI interrupt line number 1. 0_B The DPLL disable interrupt is not requested 1_B The DPLL disable interrupt is requested
PEI	1	rw	**DPLL enable interrupt** Announces the switch on of the DEN bit. This event is combined with the PDI interrupt to the common PDI + PEI interrupt line number 1. 0_B The DPLL enable interrupt is not requested 1_B The DPLL enable interrupt is requested
TINI	2	rw	**TRIGGER minimum hold time violation interrupt (dt <= THMI > 0)** 0_B No violation of minimum hold time of TRIGGER is detected 1_B A violation of minimum hold time of TRIGGER is detected
TAXI	3	rw	**TRIGGER maximum hold time violation interrupt (dt > THMA > 0)** 0_B No violation of maximum hold time of TRIGGER is detected 1_B A violation of maximum hold time of TRIGGER is detected

Field	Bits	Type	Description
SISI	4	rw	**STATE inactive slope interrupt** 0_B No inactive slope of STATE is detected 1_B An inactive slope of STATE is detected
TISI	5	rw	**TRIGGER inactive slope interrupt** The TISI bit is only set for an inactive slope when the preceding active slope was accepted. In the case of suppression of the last active slope by the plausibility check the next inactive slope is to be ignored. No set of TISI is performed in this case. 0_B No inactive slope of TRIGGER is detected 1_B An inactive slope of TRIGGER is detected
MSI	6	rw	**Missing STATE interrupt** 0_B The missing STATE interrupt is not requested 1_B The missing STATE interrupt is requested
MTI	7	rw	**Missing TRIGGER interrupt** 0_B The missing TRIGGER interrupt is not requested 1_B The missing TRIGGER interrupt is requested
SASI	8	rw	**STATE active slope interrupt** 0_B No active slope of STATE is detected 1_B An active slope of STATE is detected
TASI	9	rw	**TRIGGER active slope interrupt** 0_B No active slope of TRIGGER is detected 1_B An active slope of TRIGGER is detected
PWI	10	rw	**Plausibility window (PVT) violation interrupt of TRIGGER** 0_B The plausibility window is not violated 1_B The plausibility window is violated
W2I	11	rw	**RAM write access to RAM region 2 interrupt** 0_B The RAM write access interrupt is not requested 1_B The RAM write access interrupt is requested
W1I	12	rw	**Write access to RAM region 1b or 1c interrupt** 0_B The RAM write access interrupt is not requested 1_B The RAM write access interrupt is requested
GL1I	13	rw	**Get of lock interrupt, for SUB_INC1** 0_B The lock getting interrupt is not requested 1_B The lock getting interrupt is requested
LL1I	14	rw	**Loss of lock interrupt for SUB_INC1** 0_B The lock loss interrupt is not requested 1_B The lock loss interrupt is requested

Field	Bits	Type	Description
EI	15	rw	**Error interrupt (see status register bit 31)** 0_B The error interrupt is not requested 1_B The error interrupt is requested
GL2I	16	rw	**Get of lock interrupt, for SUB_INC2** 0_B The lock getting interrupt is not requested 1_B The lock getting interrupt is requested
LL2I	17	rw	**Loss of lock interrupt for SUB_INC2** 0_B The lock loss interrupt is not requested 1_B The lock loss interrupt is requested
TE0I	18	rw	**TRIGGER event interrupt 0** 0_B No interrupt on TRIGGER event 0 requested 1_B Interrupt on TRIGGER event 0 requested
TE1I	19	rw	**TRIGGER event interrupt 1** 0_B No interrupt on TRIGGER event 1 requested 1_B Interrupt on TRIGGER event 1 requested
TE2I	20	rw	**TRIGGER event interrupt 2** 0_B No interrupt on TRIGGER event 2 requested 1_B Interrupt on TRIGGER event 2 requested
TE3I	21	rw	**TRIGGER event interrupt 3** 0_B No interrupt on TRIGGER event 3 requested 1_B Interrupt on TRIGGER event 3 requested
TE4I	22	rw	**TRIGGER event interrupt 4** 0_B No interrupt on TRIGGER event 4 requested 1_B Interrupt on TRIGGER event 4 requested
CDTI	23	rw	**Calculation of TRIGGER duration done, only while NTI_CNT is zero** 0_B No interrupt on calculated TRIGGER duration requested or NTI_CNT is not zero 1_B Interrupt on calculated TRIGGER duration requested while NTI_CNT is zero
CDSI	24	rw	**Calculation of STATE duration done** 0_B No interrupt on calculated STATE duration requested 1_B Interrupt on calculated STATE duration requested

Field	Bits	Type	Description
TORI	25	rw	**TRIGGER out of range interrupt** 0_B TRIGGER is not out of range 1_B TRIGGER is out of range, the TOR bit in the DPLL_STATUS register is set to 1
SORI	26	rw	**STATE out of range** The interrupt occurs at line number 0. 0_B STATE is not out of range 1_B STATE is out of range, the SOR bit in the DPLL_STATUS register is set to 1
DCGI	27	rw	**Direction change interrupt** 0_B No direction change of TRIGGER is detected 1_B Direction change of TRIGGER is detected
0	31:28	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.18Register DPLL_IRQ_EN

DPLL Interrupt Enable Register

DPLL_IRQ_EN
DPLL Interrupt Enable Register **(028044_H)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				DCGI_IRQ_EN	SORI_IRQ_EN	TORI_IRQ_EN	CDSI_IRQ_EN	CDTI_IRQ_EN	TE4I_IRQ_EN	TE3I_IRQ_EN	TE2I_IRQ_EN	TE1I_IRQ_EN	TE0I_IRQ_EN	LL2I_IRQ_EN	GL2I_IRQ_EN
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EI_IRQ_EN	LL1I_IRQ_EN	GL1I_IRQ_EN	W1I_IRQ_EN	W2I_IRQ_EN	PWI_IRQ_EN	TASI_IRQ_EN	SASI_IRQ_EN	MTI_IRQ_EN	MSI_IRQ_EN	TISI_IRQ_EN	SISI_IRQ_EN	TAXI_IRQ_EN	TINI_IRQ_EN	PEI_IRQ_EN	PDI_IRQ_EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PDI_IRQ_EN	0	rw	**DPLL disable interrupt enable, when switch-off of the DEN bit** 0_B The DPLL disable interrupt is not enabled 1_B The DPLL disable interrupt is enabled
PEI_IRQ_EN	1	rw	**DPLL enable interrupt enable, when switch-on of the DEN bit** 0_B The DPLL enable interrupt is not enabled 1_B The DPLL enable interrupt is enabled
TINI_IRQ_EN	2	rw	**TRIGGER minimum hold time violation interrupt enable** 0_B Interrupt on minimum hold time violation of TRIGGER is not enabled 1_B Interrupt on minimum hold time violation of TRIGGER is enabled

Field	Bits	Type	Description
TAXI_IRQ_EN	3	rw	**TRIGGER maximum hold time violation interrupt enable** 0_B Interrupt on maximum hold time violation of TRIGGER is not enabled 1_B Interrupt on maximum hold time violation of TRIGGER is enabled
SISI_IRQ_EN	4	rw	**STATE inactive slope interrupt enable** 0_B Interrupt at the inactive slope of STATE is not enabled 1_B Interrupt at the inactive slope of STATE is enabled
TISI_IRQ_EN	5	rw	**TRIGGER inactive slope interrupt enable** 0_B Interrupt at the inactive slope of TRIGGER is not enabled 1_B Interrupt at the inactive slope of TRIGGER is enabled
MSI_IRQ_EN	6	rw	**Missing STATE interrupt enable** 0_B The missing STATE interrupt is not enabled 1_B The missing STATE interrupt is enabled
MTI_IRQ_EN	7	rw	**Missing TRIGGER interrupt enable** 0_B The missing TRIGGER interrupt is not enabled 1_B The missing TRIGGER interrupt is enabled
SASI_IRQ_EN	8	rw	**STATE active slope interrupt enable** 0_B The active slope STATE interrupt is not enabled 1_B The active slope STATE interrupt is enabled
TASI_IRQ_EN	9	rw	**TRIGGER active slope interrupt enable** 0_B The active slope TRIGGER interrupt is not enabled 1_B The active slope TRIGGER interrupt is enabled
PWI_IRQ_EN	10	rw	**Plausibility window (PVT) violation of TRIGGER interrupt enable** 0_B The plausibility violation interrupt is not enabled 1_B The plausibility violation interrupt is enabled
W2I_IRQ_EN	11	rw	**RAM write access to RAM region 2 interrupt enable** 0_B The RAM write access interrupt is not enabled 1_B The RAM write access interrupt is enabled
W1I_IRQ_EN	12	rw	**Write access to RAM region 1b or 1c interrupt enable** 0_B The RAM write access interrupt is not enabled 1_B The RAM write access interrupt is enabled
GL1I_IRQ_EN	13	rw	**Get of lock interrupt enable, when lock arises** 0_B The get of lock interrupt is not enabled 1_B The get of lock interrupt is enabled
LL1I_IRQ_EN	14	rw	**Loss of lock interrupt enable** 0_B The loss of lock interrupt is not enabled 1_B The loss of lock interrupt is enabled
EI_IRQ_EN	15	rw	**Error interrupt enable (see status register)** 0_B The error interrupt is not enabled 1_B The error interrupt is enabled
GL2I_IRQ_EN	16	rw	**Get of lock interrupt enable for SUB_INC2** 0_B The get of lock interrupt is not enabled 1_B The get of lock interrupt is enabled

Field	Bits	Type	Description
LL2I_IRQ_EN	17	rw	**Loss of lock interrupt enable for SUB_INC2** 0_B The loss of lock interrupt is not enabled 1_B The loss of lock interrupt is enabled
TE0I_IRQ_EN	18	rw	**TRIGGER event interrupt 0 enable** 0_B Interrupt on TRIGGER event 0 is not enabled 1_B Interrupt on TRIGGER event 0 is enabled
TE1I_IRQ_EN	19	rw	**TRIGGER event interrupt 1 enable** 0_B Interrupt on TRIGGER event 1 is not enabled 1_B Interrupt on TRIGGER event 1 is enabled
TE2I_IRQ_EN	20	rw	**TRIGGER event interrupt 2 enable** 0_B Interrupt on TRIGGER event 2 is not enabled 1_B Interrupt on TRIGGER event 2 is enabled
TE3I_IRQ_EN	21	rw	**TRIGGER event interrupt 3 enable** 0_B Interrupt on TRIGGER event 3 is not enabled 1_B Interrupt on TRIGGER event 3 is enabled
TE4I_IRQ_EN	22	rw	**TRIGGER event interrupt 4 enable** 0_B Interrupt on TRIGGER event 4 is not enabled 1_B Interrupt on TRIGGER event 4 is enabled
CDTI_IRQ_EN	23	rw	**Interrupt enable for calculation of TRIGGER duration done** 0_B Interrupt on calculated TRIGGER duration is not enabled 1_B Interrupt on calculated TRIGGER duration is enabled
CDSI_IRQ_EN	24	rw	**Interrupt enable for calculation of STATE duration done** 0_B Interrupt on calculated STATE duration is not enabled 1_B Interrupt on calculated STATE duration is enabled
TORI_IRQ_EN	25	rw	**TRIGGER out of range interrupt enable** 0_B Interrupt on TRIGGER out of range is not enabled 1_B Interrupt on TRIGGER out of range is enabled
SORI_IRQ_EN	26	rw	**STATE out of range interrupt enable** 0_B Interrupt on STATE out of range is not enabled 1_B Interrupt on STATE out of range is enabled
DCGI_IRQ_EN	27	rw	**Direction change interrupt enable** 0_B Interrupt on direction change of TRIGGER is not enabled 1_B Interrupt on direction change of TRIGGER is enabled
0	31:28	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.19Register DPLL_IRQ_FORCINT

DPLL Interrupt Force Register

DPLL_IRQ_FORCINT

DPLL Interrupt Force Register **(028048$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				TRG_DCGI	TRG_SORI	TRG_TORI	TRG_CDSI	TRG_CDTI	TRG_TE4I	TRG_TE3I	TRG_TE2I	TRG_TE1I	TRG_TE0I	TRG_LL2I	TRG_GL2I
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRG_EI	TRG_LL1I	TRG_GL1I	TRG_W1I	TRG_W2I	TRG_PWI	TRG_TASI	TRG_SASI	TRG_MTI	TRG_MSI	TRG_TISI	TRG_SISI	TRG_TAXI	TRG_TINI	TRG_PEI	TRG_PDI
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TRG_PDI	0	rw	**Force Interrupt PDI** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B The corresponding interrupt is not forced 1_B The corresponding interrupt is forced for one clock
TRG_PEI	1	rw	**Force Interrupt PEI** Coding see bit 0.
TRG_TINI	2	rw	**Force Interrupt TINI** Coding see bit 0.
TRG_TAXI	3	rw	**Force Interrupt TAXI** Coding see bit 0.
TRG_SISI	4	rw	**Force Interrupt SISI** Coding see bit 0.
TRG_TISI	5	rw	**Force Interrupt TISI** Coding see bit 0.
TRG_MSI	6	rw	**Force Interrupt MSI** Coding see bit 0.
TRG_MTI	7	rw	**Force Interrupt MTI** Coding see bit 0.
TRG_SASI	8	rw	**Force Interrupt SASI** Coding see bit 0.
TRG_TASI	9	rw	**Force Interrupt TASI** Coding see bit 0.
TRG_PWI	10	rw	**Force Interrupt PWI** Coding see bit 0.
TRG_W2I	11	rw	**Force Interrupt W2I** Coding see bit 0.

Field	Bits	Type	Description
TRG_W1I	12	rw	**Force Interrupt W1I** Coding see bit 0.
TRG_GL1I	13	rw	**Force Interrupt GL1I** Coding see bit 0.
TRG_LL1I	14	rw	**Force Interrupt LL1I** Coding see bit 0.
TRG_EI	15	rw	**Force Interrupt EI** Coding see bit 0.
TRG_GL2I	16	rw	**Force Interrupt GL2I** Coding see bit 0.
TRG_LL2I	17	rw	**Force Interrupt LL2I** Coding see bit 0.
TRG_TE0I	18	rw	**Force Interrupt TE0I** Coding see bit 0.
TRG_TE1I	19	rw	**Force Interrupt TE1I** Coding see bit 0.
TRG_TE2I	20	rw	**Force Interrupt TE2I** Coding see bit 0.
TRG_TE3I	21	rw	**Force Interrupt TE3I** Coding see bit 0.
TRG_TE4I	22	rw	**Force Interrupt TE4I** Coding see bit 0.
TRG_CDTI	23	rw	**Force Interrupt CDTI** Coding see bit 0.
TRG_CDSI	24	rw	**Force Interrupt CDSI** Coding see bit 0.
TRG_TORI	25	rw	**Force Interrupt TORI** Coding see bit 0.
TRG_SORI	26	rw	**Force Interrupt SORI** Coding see bit 0.
TRG_DCGI	27	rw	**Force interrupt DCGI** Coding see bit 0.
0	31:28	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.20Register DPLL_IRQ_MODE

DPLL Interrupt Mode Register

DPLL_IRQ_MODE
DPLL Interrupt Mode Register **(02804C$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														IRQ_MODE	
r														rw	

Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** The interrupt modes are described in **Section 28.4.5**. 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero

28.20.12.21Register DPLL_EIRQ_EN

DPLL Error Interrupt Enable Register

DPLL_EIRQ_EN
DPLL Error Interrupt Enable Register **(028050$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				DCGI_EIRQ_EN	SORI_EIRQ_EN	TORI_EIRQ_EN	CDSI_EIRQ_EN	CDTI_EIRQ_EN	TE4I_EIRQ_EN	TE3I_EIRQ_EN	TE2I_EIRQ_EN	TE1I_EIRQ_EN	TE0I_EIRQ_EN	LL2I_EIRQ_EN	GL2I_EIRQ_EN
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EI_EIRQ_EN	LL1I_EIRQ_EN	GL1I_EIRQ_EN	W1I_EIRQ_EN	W2I_EIRQ_EN	PWI_EIRQ_EN	TASI_EIRQ_EN	SASI_EIRQ_EN	MTI_EIRQ_EN	MSI_EIRQ_EN	TISI_EIRQ_EN	SISI_EIRQ_EN	TAXI_EIRQ_EN	TINI_EIRQ_EN	PEI_EIRQ_EN	PDI_EIRQ_EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PDI_EIRQ_EN	0	rw	**DPLL disable interrupt enable, when switch off of the DEN bit** 0_B The DPLL disable interrupt is not enabled 1_B The DPLL disable interrupt is enabled
PEI_EIRQ_EN	1	rw	**DPLL enable interrupt enable, when switch on of the DEN bit** 0_B The DPLL enable interrupt is not enabled 1_B The DPLL enable interrupt is enabled
TINI_EIRQ_EN	2	rw	**TRIGGER minimum hold time violation interrupt enable bit** 0_B Minimum hold time violation of TRIGGER interrupt is not enabled 1_B The minimum hold time violation of TRIGGER interrupt is enabled
TAXI_EIRQ_EN	3	rw	**TRIGGER maximum hold time violation interrupt enable bit** 0_B Maximum hold time violation of TRIGGER interrupt is not enabled 1_B The maximum hold time violation of TRIGGER interrupt is enabled
SISI_EIRQ_EN	4	rw	**STATE inactive slope interrupt enable bit** 0_B The interrupt at the inactive slope of STATE is not enabled 1_B The interrupt at the inactive slope of STATE is enabled
TISI_EIRQ_EN	5	rw	**TRIGGER inactive slope interrupt enable bit** 0_B The interrupt at the inactive slope of TRIGGER is not enabled 1_B The interrupt at the inactive slope of TRIGGER is enabled
MSI_EIRQ_EN	6	rw	**Missing STATE interrupt enable** 0_B The missing STATE interrupt is not enabled 1_B The missing STATE interrupt is enabled
MTI_EIRQ_EN	7	rw	**Missing TRIGGER interrupt enable** 0_B The missing TRIGGER interrupt is not enabled 1_B The missing TRIGGER interrupt is enabled
SASI_EIRQ_EN	8	rw	**STATE active slope interrupt enable** 0_B The active slope STATE interrupt is not enabled. 1_B The active slope STATE interrupt is enabled
TASI_EIRQ_EN	9	rw	**TRIGGER active slope interrupt enable** 0_B The active slope TRIGGER interrupt is not enabled 1_B The active slope TRIGGER interrupt is enabled
PWI_EIRQ_EN	10	rw	**Plausibility window (PVT) violation interrupt of TRIGGER enable** 0_B The plausibility violation interrupt is not enabled 1_B The plausibility violation interrupt is enabled
W2I_EIRQ_EN	11	rw	**RAM write access to RAM region 2 interrupt enable** 0_B The RAM write access interrupt is not enabled 1_B The RAM write access interrupt is enabled
W1I_EIRQ_EN	12	rw	**Write access to RAM region 1b or 1c interrupt** 0_B The RAM write access interrupt is not enabled 1_B The RAM write access interrupt is enabled.
GL1I_EIRQ_EN	13	rw	**Get of lock interrupt enable, when lock arises** 0_B The lock getting interrupt is not enabled 1_B The lock getting interrupt is enabled

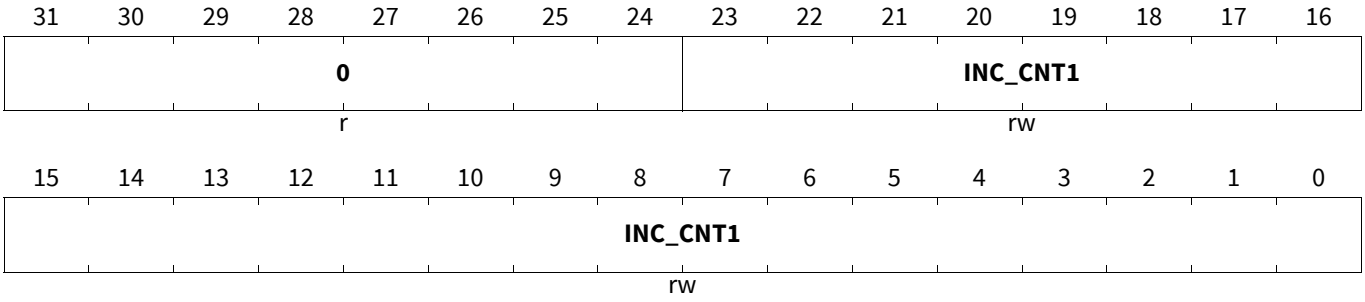
Field	Bits	Type	Description
LL1I_EIRQ_EN	14	rw	**Loss of lock interrupt enable** 0_B The lock loss interrupt is not enabled 1_B The lock loss interrupt is enabled
EI_EIRQ_EN	15	rw	**Error interrupt enable (see status register)** 0_B The error interrupt is not enabled 1_B The error interrupt is enabled
GL2I_EIRQ_EN	16	rw	**Get of lock interrupt enable for SUB_INC2** 0_B The lock getting interrupt is not requested 1_B The lock getting interrupt is requested
LL2I_EIRQ_EN	17	rw	**Loss of lock interrupt enable for SUB_INC2** 0_B The lock loss interrupt is not requested 1_B The lock loss interrupt is requested
TE0I_EIRQ_EN	18	rw	**TRIGGER event interrupt 0 enable** 0_B No Interrupt on TRIGGER event 0 enabled 1_B Interrupt on TRIGGER event 0 enabled
TE1I_EIRQ_EN	19	rw	**TRIGGER event interrupt 1 enable** 0_B No Interrupt on TRIGGER event 1 enabled 1_B Interrupt on TRIGGER event 1 enabled
TE2I_EIRQ_EN	20	rw	**TRIGGER event interrupt 2 enable** 0_B No Interrupt on TRIGGER event 2 enabled 1_B Interrupt on TRIGGER event 2 enabled
TE3I_EIRQ_EN	21	rw	**TRIGGER event interrupt 3 enable** 0_B No Interrupt on TRIGGER event 3 enabled 1_B Interrupt on TRIGGER event 3 enabled
TE4I_EIRQ_EN	22	rw	**TRIGGER event interrupt 4 enable** 0_B No Interrupt on TRIGGER event 4 enabled 1_B Interrupt on TRIGGER event 4 enabled
CDTI_EIRQ_EN	23	rw	**Enable interrupt when calculation of TRIGGER duration done** 0_B No Interrupt on calculated TRIGGER duration enabled 1_B Interrupt on calculated TRIGGER duration enabled
CDSI_EIRQ_EN	24	rw	**Enable interrupt when calculation of TRIGGER duration done** 0_B No Interrupt on calculated STATE duration enabled 1_B Interrupt on calculated STATE duration enabled
TORI_EIRQ_EN	25	rw	**TRIGGER out of range interrupt** 0_B No Interrupt when TRIGGER is out of range enabled 1_B Interrupt when TRIGGER is out of range enabled
SORI_EIRQ_EN	26	rw	**STATE out of range** 0_B No Interrupt when STATE is out of range enabled 1_B Interrupt when STATE is out of range enabled
DCGI_EIRQ_EN	27	rw	**Direction change interrupt** 0_B No Interrupt when a direction change of TRIGGER is detected 1_B Interrupt when a direction change of TRIGGER is detected
0	31:28	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.22Register DPLL_INC_CNT1

DPLL Counter for Pulses for TBU_CH1_BASE to be Sent in Automatic End Mode

Counter Value of Sent SUB_INC1 Pulses

DPLL_INC_CNT1
DPLL Counter for Pulses for TBU_CH1_BASE to be Sent in Automatic End Mode(0280B0$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
INC_CNT1	23:0	rw	**Actual number of pulses to be still sent out at the current increment until the next active input signal in automatic end mode** Automatic addition of the number of demanded pulses MLT/MLS1 when getting an active *TRIGGER*/*STATE* input in normal or emergency mode respectively **when SGE1=1**;writeable only for test purposes when DEN=0. In the case of a change of the direction the wrong number of pulses are corrected twice: Add the difference between NMB_T and INC_CNT1 twice to INC_CNT1 before sending out the correction pulses. This value can only be written when the DPLL is disabled.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

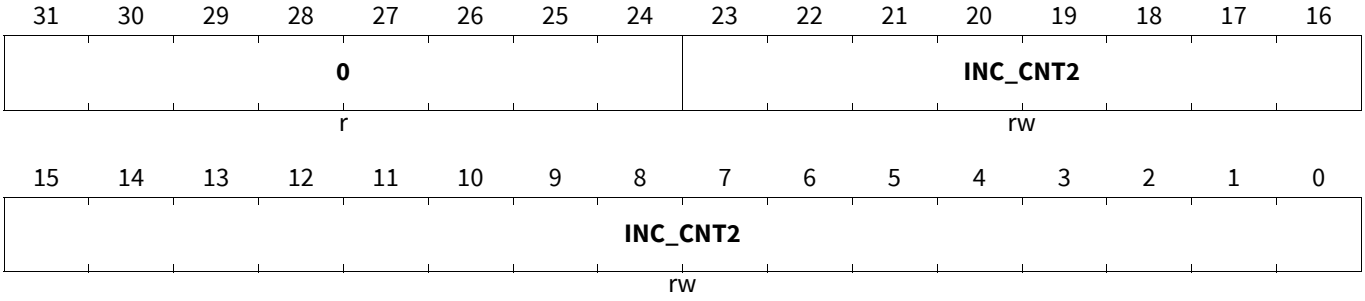
28.20.12.23Register DPLL_INC_CNT2

DPLL Counter for Pulses for TBU_TS2 to be Sent in Automatic End Mode

Counter Value of sent SUB_INC2 values (for SMC=1 and RMO=1)

DPLL_INC_CNT2

DPLL Counter for Pulses for TBU_TS2 to be Sent in Automatic End Mode(0280B4$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
INC_CNT2	23:0	rw	**Actual number of pulses to be still sent out at the current increment until the next active input signal in automatic end mode** Automatic addition of the number of demanded pulses MLS2 when getting an active *TRIGGER*/*STATE* input in normal or emergency mode respectively **when SGE2=1**; writeable only for test purposes when DEN=0; In the case of a change of the direction the wrong number of pulses are corrected twice: Add the difference between NMB_S and INC_CNT2 twice to INC_CNT2 before sending out the correction pulses. This value can only be written when the DPLL is disabled.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

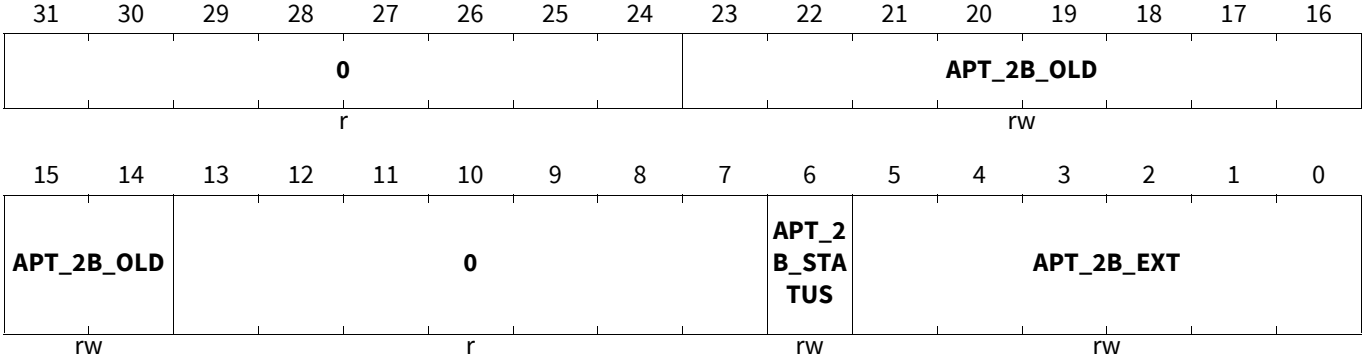
28.20.12.24Register DPLL_APT_SYNC

DPLL Old RAM Pointer and Offset Value for TRIGGER

TRIGGER Time Stamp Field Offset at Synchronization Time

DPLL_APT_SYNC

DPLL Old RAM Pointer and Offset Value for TRIGGER(0280B8$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
APT_2B_EXT	5:0	rw	**Address pointer 2b extension** This offset value determines, by which value the APT_2B is changed at the synchronization time; set by CPU before the synchronization is performed. This offset value is the number of virtual increments to be inserted in the TSF for an imminent intended synchronization; the CPU sets its value dependent on the gaps until the synchronization time taking into account the considered NUTE value to be set and including the next future increment (when SYN_T_OLD is still 1). When the synchronization takes place, this value is to be added to the APT_2B address pointer (for forward direction, DIR1=0) and the APT_2B_STATUS bit is cleared after it. For backward direction subtract APT_2B_EXT accordingly. This correction is done after updating the RAM TSF with the last TS_T value. When the synchronization is intended and the NUTE value is to be set to FULL_SCALE after it, the APT_2B_EXT value must be set to 2*SYN_NT in order to be able to fill all gaps in the extended TSF_T with the corresponding values by the CPU. When still not all values for FULL_SCALE are available, the APT_2B_EXT value considers only a share according to the corresponding NUTE value to be set after the synchronization.
APT_2B_STATUS	6	rw	**Address pointer 2b status** Set by CPU before the synchronization is performed. The value is cleared when the APT_2B_OLD value is written. 0_B APT_2B_EXT is not to be considered 1_B APT_2B_EXT has to be considered for time stamp field extension
APT_2B_OLD	23:14	rw	**Address pointer TRIGGER for RAM region 2b at synchronization time** This value is set by the current APT_2B value when the synchronization takes place for the first active TRIGGER event after writing APT_2C but before adding the offset value APT_2B_EXT (that means: when APT_2B_STATUS=1). Address pointer APT_2B value at the moment of synchronization, before the offset value is added, that means the pointer with this value points to the last value before the additional inserted gap.
0	13:7, 31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.25Register DPLL_APS_SYNC

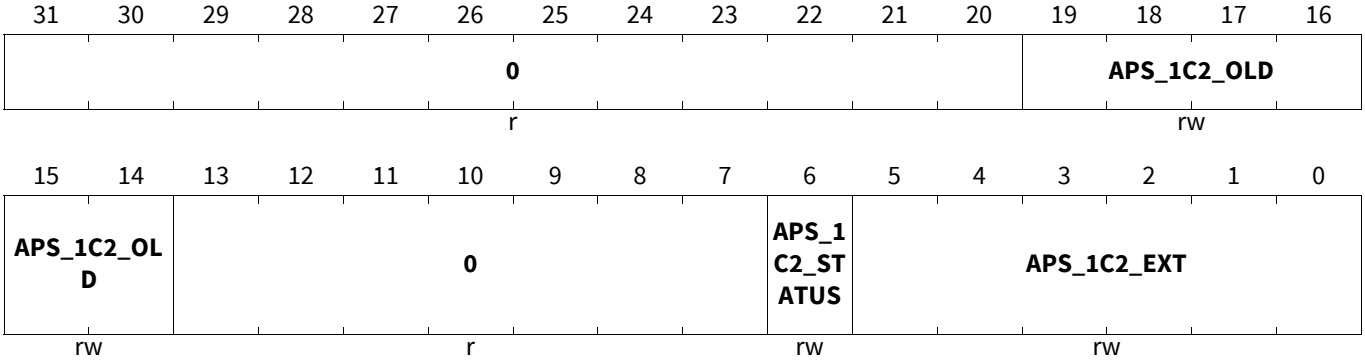
DPLL Old RAM Pointer and Offset Value for STATE

Note: STATE Time Stamp Field Offset at Synchronization Time

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is not set. If DPLL_CTRL_11.STATE_EXT is set any read/write access to this register will return AEI_STATUS = 0b10.

DPLL_APS_SYNC

DPLL Old RAM Pointer and Offset Value for STATE(0280BC$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
APS_1C2_EXT	5:0	rw	**Address pointer 1c2 extension** This offset value determines, by which value the APS_1C2 is changed at the synchronization time; set by CPU before the synchronization is performed. This offset value is the number of virtual increments to be inserted in the TSF for an imminent intended synchronization; the CPU sets its value dependent on the gaps until the synchronization time taking into account the considered NUSE value to be set and including the next future increment (when SYN_S_OLD is still 1). When the synchronization takes place, this value is to be added to the APS_1C2 address pointer (for forward direction, DIR2=0) and the APT_1c2_status bit is cleared after it. For backward direction subtract APS_1C2_EXT accordingly. When the synchronization is intended and the NUSE value is to be set to FULL_SCALE after it, the APS_1C2_EXT value must be set to SYN_NS (for SYSF=1) or 2*SYN_NS (for SYSF=0) in order to be able to fill all gaps in the extended TSF_S with the corresponding values by the CPU. When still not all values for FULL_SCALE are available, the APS_1C2_EXT value considers only a share according to the NUSE value to be set after the synchronization.
APS_1C2_STATUS	6	rw	**Address pointer 1c2 status** Set by CPU before the synchronization is performed. The value is cleared automatically when the APS_1C2_OLD value is written. 0_B APS_1C2_EXT is not to be considered 1_B APS_1C2_EXT has to be considered for time stamp field extension
APS_1C2_OLD	19:14	rw	**Address pointer STATE for RAM region 1c2 at synchronization time** This value is set by the current APS_1C2 value when the synchronization takes place for the first active STATE event after writing APS_1C3 but before adding the offset value APS_1C2_EXT (that means: when APS_1C2_STATUS=1). Address pointer APS_1C2 value at the moment of synchronization, before the offset value is added, that means the pointer with this value points to the last value before the additional inserted gap.

Field	Bits	Type	Description
0	13:7, 31:20	r	**Reserved** Read as zero, shall be written as zero.

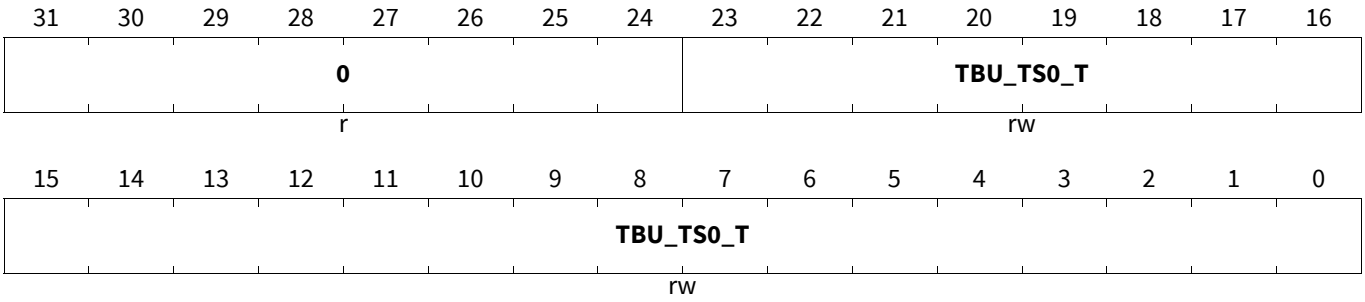
28.20.12.26Register DPLL_TBU_TS0_T

DPLL TBU_TS0 Value at Last TRIGGER Event

Time Stamp Value for the last active TRIGGER

DPLL_TBU_TS0_T

DPLL TBU_TS0 Value at Last TRIGGER Event (0280C0$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TBU_TS0_T	23:0	rw	**Value of TBU_TS0 at the last TRIGGER event** For each T_valid the value of TBU_TS0 is stored in this register; the register is writeable only for test purposes when DEN=0. This value can only be written when the DPLL is disabled.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

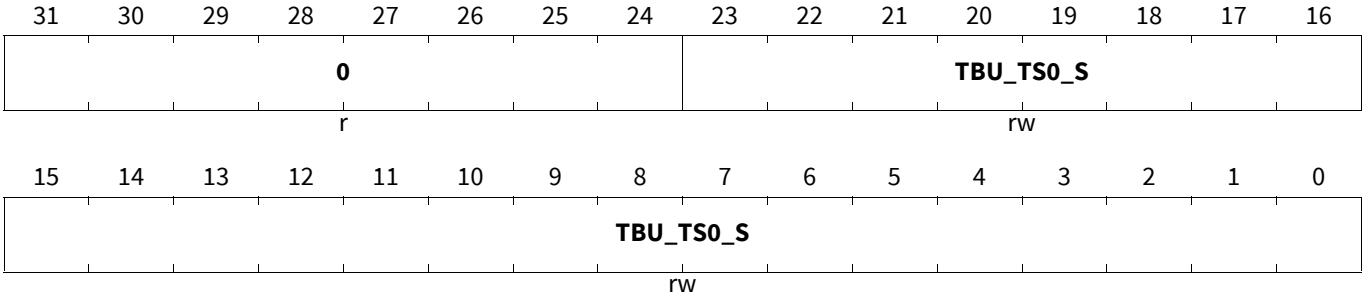
28.20.12.27Register DPLL_TBU_TS0_S

DPLL TBU_TS0 Value at Last STATE Event

Time Stamp Value for the last active STATE

DPLL_TBU_TS0_S

DPLL TBU_TS0 Value at Last STATE Event (0280C4$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TBU_TS0_S	23:0	rw	**Value of TBU_TS0 at the last STATE event** For each S_VALID the value of TBU_TS0 is stored in this register;the register is writeable only for test purposes when DEN=0. This value can only be written when the DPLL is disabled.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.28Register DPLL_ADD_IN_LD1

DPLL Direct Load Input Value for SUB_INC1

ADD_IN Value in Direct Load Mode for TRIGGER

DPLL_ADD_IN_LD1

DPLL Direct Load Input Value for SUB_INC1 ($0280C8_H$) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								**ADD_IN_LD1**							
r								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD_IN_LD1															
rw															

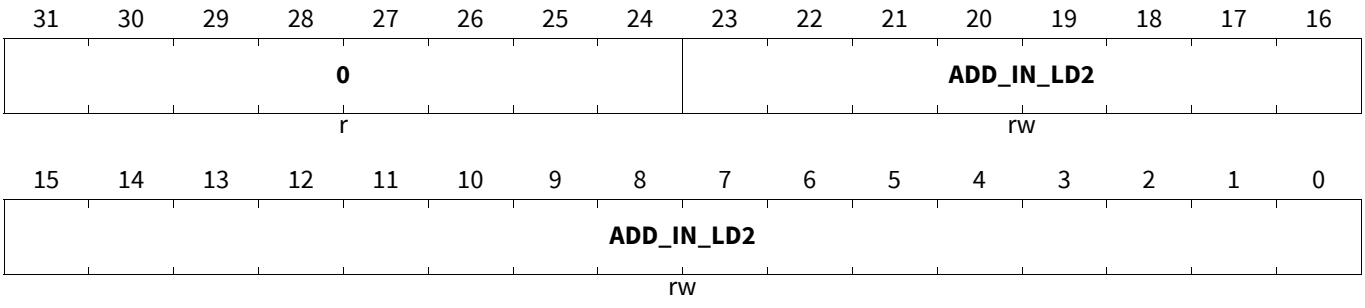
Field	Bits	Type	Description
ADD_IN_LD1	23:0	rw	**Input value for SUB_INC1 generation** Given by CPU. This value can be used in normal und emergency mode (SMC=0) as well as for SMC=1. **For DLM1 = 1:** The value is loaded by the CPU but used by the DPLL only for DLM1=1 (see DPLL_CTRL_1 register). When switching DLM1 to 1, the value in the register is used for the SUB_INC1 generation beginning from the next active *TRIGGER* or *STATE* event respectively independently if new values are written by the CPU or not. When a new value is written the output frequency changes according to the given value beginning immediately from the moment of writing. Do not wait for performing step 10 in the state machine for ADD_IN calculations. If the ADD_IN_LD1 value is zero all pulses are sent with the highest possible frequency. **For DLM1 = 0:** The value loaded by the CPU is stored directly in the internal add_in register which is used to control the sub increment pulse generator directly (see DPLL_CTRL_1 register, DLM1 = 0). When a new ADD_IN_LD1 value is written the output frequency is immediately changed from the moment of writing. The ADD_IN values calculated internally of the DPLL are written to the internal ADD_IN register as well. In the moment when the internal calculation of the ADD_IN values is writing the results into the internal ADD_IN register of the pulse generator the internally calculated ADD_IN values does always have higher priority compared to the values written via the ADD_IN_LD1 register. If the ADD_IN_LD1 value is zero all pulses are sent with the highest possible frequency.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.29Register DPLL_ADD_IN_LD2

DPLL Direct Load Input Value for SUB_INC2

DPLL_ADD_IN_LD2

DPLL Direct Load Input Value for SUB_INC2 (0280CC$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
ADD_IN_LD2	23:0	rw	**Input value for SUB_INC2 generation** Given by CPU. This value can be used for SMC=1 while RMO=1. **For DLM2 = 1:** The value is loaded by the CPU but used by the DPLL only for DLM2=1 (see DPLL_CTRL_1 register). When switching DLM2 to 1, the value in the register is used for the SUB_INC2 generation beginning from the next *STATE* event respectively independently if new values are written by the CPU or not. When a new value is written the output frequency changes according to the given value beginning immediately from the moment of writing. Do not wait for performing step 30 in the state machine for ADD_IN calculations. If the ADD_IN_LD2 value is zero all pulses are sent with the highest possible frequency. **For DLM2 = 0:** The value loaded by the CPU is stored directly in the internal add_in register which is used to control the sub increment pulse generator directly (see DPLL_CTRL_1 register, DLM2 = 0). When a new ADD_IN_LD2 value is written the output frequency is immediately changed from the moment of writing. The ADD_IN values calculated internally of the DPLL are written to the internal ADD_IN register as well. In the moment when the internal calculation of the ADD_IN values is writing the results into the internal ADD_IN register of the pulse generator the internally calculated ADD_IN values does always have higher priority compared to the values written via the ADD_IN_LD2 register. If the ADD_IN_LD2 value is zero all pulses are sent with the highest possible frequency.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.30Register DPLL_STATUS

DPLL Status Register

Note: The DPLL_STATUS register is reset, when the DPLL is disabled (switching DEN from 1 to 0).

DPLL_STATUS
DPLL Status Register **(0280FC$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ERR	**LOCK1**	**FTD**	**FSD**	**SYT**	**SYS**	**LOCK2**	**0**	**BWD1**	**BWD2**	**ITN**	**ISN**	**CAIP1**	**CAIP2**	**CSVT**	**CSVS**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOW_RES	**0**		**RAM2_ERR**	**MT**	**TOR**	**MS**	**SOR**	**PSE**	**RCT**	**RCS**	**CRO**	**CTO**	**0**	**CSO**	**FPCE**
r	r		rw	rw	rw	rw	rw	r	r	r	rw	rw	r	rw	rw

Field	Bits	Type	Description
FPCE	0	rw	**Fast pulse correction error** 0_B No error at fast pulse correction detected 1_B Negative value of MPVAL1/2 used for fast pulse correction mode
CSO	1	rw	**Calculated STATE duration overflow** Bit is set when equations DPLL-10a or DPLL-10b lead to an overflow. 0_B No overflow at equation DPLL-10a or b 1_B Overflow at equation DPLL-10a or b
CTO	3	rw	**Calculated TRIGGER duration overflow** Bit is set when equations DPLL-5a or DPLL-5b lead to an overflow. When one of the above bits is set the corresponding register contains the maximum value 0xFFFFFF. 0_B No overflow at equation DPLL-5a or b 1_B Overflow at equation DPLL-5a or b
CRO	4	rw	**Calculated Reciprocal value overflow** Bit is set when the calculation of RDT_T_ACT or RDT_S_ACT leads to an overflow. An overflow in calculation of reciprocal values can occur, when the condition of Note [3)] to the DPLL_CTRL_0 register is violated (see **Section 28.20.12.1**). Such an overflow can occur according to the calculations in equations (DPLL-1c) or (DPLL-6c). The overflow is detected when after the calculation and shifting left 32 bits at least one of the bits 31 to 24 is not zero. In that case the corresponding register is set to 0xFFFFFF. 0_B No overflow at any reciprocal calculation 1_B Overflow for at least one reciprocal calculation
RCS	5	r	**Resolution conflict STATE** 0_B No resolution conflict detected 1_B The TS0_HRS value is set to 1 while LOW_RES=0
RCT	6	r	**Resolution conflict TRIGGER** 0_B No resolution conflict detected 1_B The TS0_HRT value is set to 1 while LOW_RES=0
PSE	7	r	**Prediction space configuration error** 0_B No prediction space error detected 1_B Configured offset value of RAM2 is too small in order to store all TNU+1 values twice in FULL_SCALE

Field	Bits	Type	Description
SOR	8	rw	**STATE out of range** The SOR bit is set, when the time to the next active *STATE* slope exceeds the value of the last nominal *STATE* duration multiplied with the value of the SLR register (see **Section 28.20.12.73**) and is reset, when at the current or last active input event a direction change was detected. The SYS bit is not influenced by setting the SOR bit. 0_B All STATE signal events appear within SLR interval or a direction change was detected 1_B At least one STATE signal event is out of SLR; address pointers APS, APS_1C2 and APS_1C3 are frozen, and the generation of pulses SUB_INC1, 2, respectively, is stopped
MS	9	rw	**Missing STATE detected according to SOV** 0_B No missing STATE detected or a new valid STATE slope occurred 1_B At least one missing STATE detected after the last valid slope
TOR	10	rw	**TRIGGER out of range** The TOR bit is set, when the time to the next active *TRIGGER* slope exceeds the value of the last nominal *TRIGGER* duration multiplied with the value of the TLR register (see **Section 28.20.12.72**) and is reset, when at the current or last active input event a direction change was detected. The SYT bit is not influenced by setting the TOR bit. 0_B All TRIGGER signal events appear within TLR interval or a direction change was detected 1_B At least one TRIGGER signal event is out of TLR; address pointers APT, APT_2B and APT_2C are frozen, and the generation of pulses SUB_INC1 is stopped
MT	11	rw	**Missing TRIGGER detected according to TOV** 0_B No missing TRIGGER detected or a new valid TRIGGER slope occurred 1_B At least one missing TRIGGER detected after the last valid slope
RAM2_ERR	12	rw	**DPLL internal access to not configured RAM2 memory space** 0_B No access to not configured RAM2 memory space 1_B Access to not configured RAM2 memory space
LOW_RES	15	r	**Low resolution of TBU_TS0 is used for DPLL input** This value reflects the input signal LOW_RES. 0_B The lower 24 bits of TBU_TS0 are used as input for the DPLL 1_B The higher 24 bits of TBU_TS0 are used as input for the DPLL
CSVS	16	r	**Current signal value STATE** 0_B The last STATE_S value was 0 1_B The last STATE_S value was 1
CSVT	17	r	**Current signal value TRIGGER** 0_B The last TRIGGER_S value was 0 1_B The last TRIGGER_S value was 1
CAIP2	18	r	**Calculation of upper half actions in progress** 0_B Currently no action calculation, new data requests possible 1_B Action calculation in progress, no new data requests possible

Field	Bits	Type	Description
CAIP1	19	r	**Calculation of lower half actionsin progress** 0_B Currently no action calculation, new data requests possible 1_B Action calculation in progress, no new data requests possible
ISN	20	r	**Increment number of STATE is not plausible** 0_B The number of STATE events between synchronization gaps is plausible, a direction change is detected, or the APS_1C3 pointer is written 1_B After setting LOCK1 in emergency mode (SMC=0 and RMO=1) or LOCK2 for SMC=RMO=1, missing or additional STATE signals detected; bit is cleared when a direction change is detected or the APS_1C3 is written
ITN	21	r	**Increment number of TRIGGER is not plausible** Bit is set when the number of TRIGGERS is different to profile. 0_B The number of TRIGGER events between synchronization gaps is plausible, a direction change is detected, or the address pointer APT_2C is written 1_B After setting LOCK1 in normal mode (for SMC=0 or SMC=1) or in emergency mode (only for SMC=0), missing or additional TRIGGER signals detected; bit is cleared when a direction change is detected or the APT_2C is written
BWD2	22	r	**Backwards drive of SUB_INC2** 0_B Forward direction 1_B Backward direction
BWD1	23	r	**Backwards drive of SUB_INC1** 0_B Forward direction 1_B Backward direction
LOCK2	25	r	**DPLL Lock status concerning SUB_INC2** Locking of SUB_INC2 appears **for RMO=SMC=1:** Bit is set, when SYS is set and the number of events between two missing *STATE*s is as expected by the SYN_S values. LOCK2 is set for SMC=RMO=1: for an active STATE event when SYS is set and SYN_NS=0or when SYS is set and the profile stored in the ADT_Si field matches once between two gaps. LOCK2 is reset: for SMC=RMO=1 when a missing STATE event occurs while SYN_S=1. This does mean an unexpected missing STATE. when the corresponding input signal STATE is out of locking range SLR 0_B The DPLL is not locked concerning STATE for SMC=1 1_B The DPLL is locked concerning STATE for SMC=1
SYS	26	r	**Synchronization condition of STATE fixed** This bit is set when the CPU writes to the APS_1C3 address pointer.
SYT	27	r	**Synchronization condition of TRIGGER fixed** This bit is set when the CPU writes to the APT_2C address pointer.

Field	Bits	Type	Description
FSD	28	r	**First STATE detected** No change of FSD for switching from normal to emergency mode or vice versa. 0_B Still no active STATE event was detected after enabling DPLL 1_B At least one active STATE event was detected after enabling DPLL
FTD	29	r	**First TRIGGER detected** No change of FTD for switching from normal to emergency mode or vice versa. 0_B No active TRIGGER event was detected after enabling DPLL 1_B At least one active TRIGGER event was detected after enabling DPLL

Field	Bits	Type	Description
LOCK1	30	r	**DPLL Lock status concerning SUB_INC1** LOCK1 is set: • in normal mode (for RMO=SMC=0, LCD=0): Bit is set for an active TRIGGER event when SYT is set and the number of events between two gaps is as expected by the profile (NT values in the ADT_T[i] field) or when SYN_NT=0 and SYT=1. • in normal mode (for RMO=SMC=0, LCD=1): Bit is set for an active TRIGGER event when SYT is set and the number of events between two increments without missing TRIGGER (no gap) is as expected by the profile (NT values in the ADT_T[i] field) • in emergency mode (for RMO=1 and SMC=0): Bit is set for an active STATE event, when SYS is set and the received events are in correspondence to the profile (NS values in the ADT_S[i] field) for at least two (four in case of direction change) expected missing STATE events or when SYN_NS=0. • for SMC=1: Bit is set for an active TRIGGER even when SYT is set and SYN_NT=0 or when SYT is set and the profile stored in the ADT_T[i] field matches once between two gaps. LOCK1 is reset for RMO=SMC=0: • when a corresponding missing TRIGGER event occurs while SYN_T=1. This does mean an unexpected missing TRIGGER. • when the corresponding input signal TRIGGER is out of locking range TLR, • when a corresponding direction change is detected: for RMO=1 and SMC=0: • when a corresponding missing STATE event occurs while SYN_S=1. This does mean an unexpected missing STATE. • when the corresponding input signal STATE is out of locking range TLR for SMC=1: • when a corresponding missing TRIGGER event occurs while SYN_T=1. This does mean an unexpected missing TRIGGER. • when the corresponding input signal TRIGGER is out of locking range TLR, • when a corresponding direction change is detected 0_B The DPLL is not locked for TRIGGER (while SMC=RMO=0 or SMC=1) or for STATE (while SMC=0 and RMO=1) 1_B The DPLL is locked for TRIGGER (while SMC=RMO=0 or SMC=1) or for STATE (while SMC=0 and RMO=1)
ERR	31	r	**Error during configuration or operation resulting in unexpected values** 0_B When all bits in position 8 to 0 and 10 and 12 are zero 1_B When at least one bit in position 8 to 0 or 10 or 12 is one

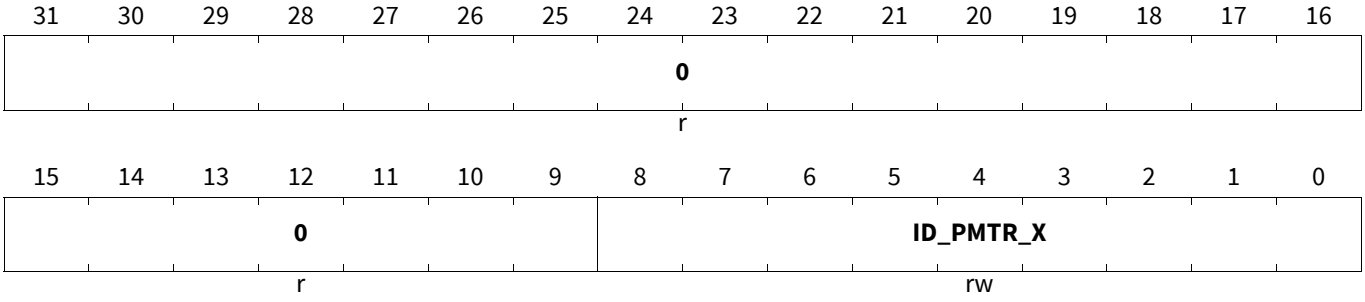
Field	Bits	Type	Description
0	2, 14:13, 24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.31Register DPLL_ID_PMTR_[z]

DPLL ID Information for Input Signal PMT z Register

DPLL_ID_PMTR_z (z=0-31)

DPLL ID Information for Input Signal PMT z Register(028100_H+z*4) **Application Reset Value: 0000 $01FE_H$**



Field	Bits	Type	Description
ID_PMTR_X	8:0	rw	**ID information to the input signal PMTR[z] from the ARU** This value can only be written when the action [z] is disabled by the correspondent bit AENz=0 of the registers DPLL_CTRL_2, ...5, respectively, or when the DPLL is disabled (DEN=0).
0	31:9	r	**Reserved** Read as zero, shall be written as zero.

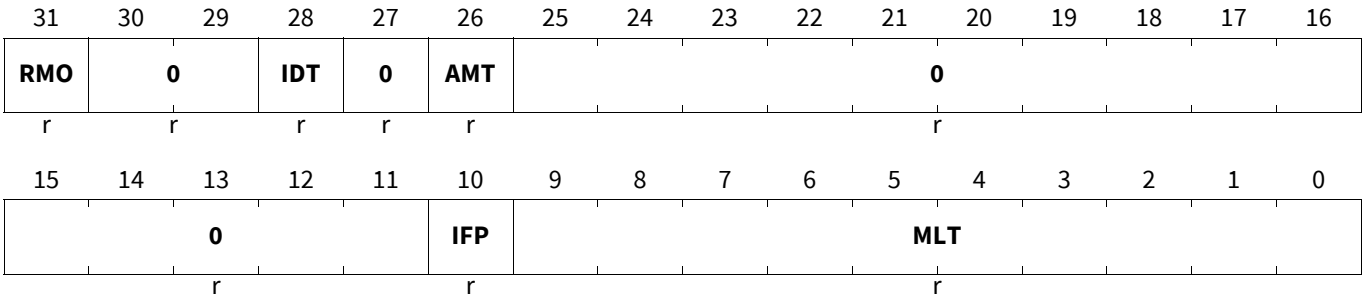
28.20.12.32Register DPLL_CTRL_0_SHADOW_TRIGGER

DPLL Control 0 Shadow Trigger Register

Shadow Register of DPLL_CTRL_0 controlled by an active TRIGGER Slope

DPLL_CTRL_0_SHADOW_TRIGGER

DPLL Control 0 Shadow Trigger Register **($0281E0_H$)** **Application Reset Value: 0000 0257_H**



Field	Bits	Type	Description
MLT	9:0	r	**Multiplier for TRIGGER** 1) MLT+1 is number of SUB_INC1 pulses between two TRIGGER events in normal mode (1…1024).
IFP	10	r	**Input filter position** 1) Value contains position or time related information.
AMT	26	r	**Adapt mode TRIGGER** 1) Use of adaptation information of TRIGGER.
IDT	28	r	**Input delay TRIGGER** 1) Use of input delay information transmitted in FT part of the TRIGGER signal.
RMO	31	r	**Reference mode** 1) Selection of the relevant the input signal for generation of SUB_INC1.
0	25:11, 27, 30:29	r	**Reserved** Read as zero, shall be written as zero.

1) Only the values characterized by 1) are stored for an active TRIGGER slope. All other values remain 0. When DEN=0 the relevant bit values of the original register DPLL_CTRL_0 are transferred without any input event at the next system clock. This results in the above reset value.

28.20.12.33Register DPLL_CTRL_0_SHADOW_STATE

DPLL Control 0 Shadow STATE Register

Shadow Register of DPLL_CTRL_0 controlled by an active STATE Slope

DPLL_CTRL_0_SHADOW_STATE
DPLL Control 0 Shadow STATE Register (0281E4$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMO		**0**		**IDS**	**0**	**AMS**					**0**				
r		r		r	r	r					r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		0			**IFP**	**0**									
		r			r	r									

Field	Bits	Type	Description
IFP	10	r	**Input filter position** 1) **Value contains position or time related information.**

Field	Bits	Type	Description
AMS	25	r	**Adapt mode STATE** 1) Use of adaptation information of STATE.
IDS	27	r	**Input delay STATE** 1) Use of input delay information transmitted in FT part of the STATE signal.
RMO	31	r	**Reference mode** 1) Selection of the relevant the input signal for generation of SUB_INC1.
0	9:0, 24:11, 26, 30:28	r	**Reserved** Read as zero, shall be written as zero.

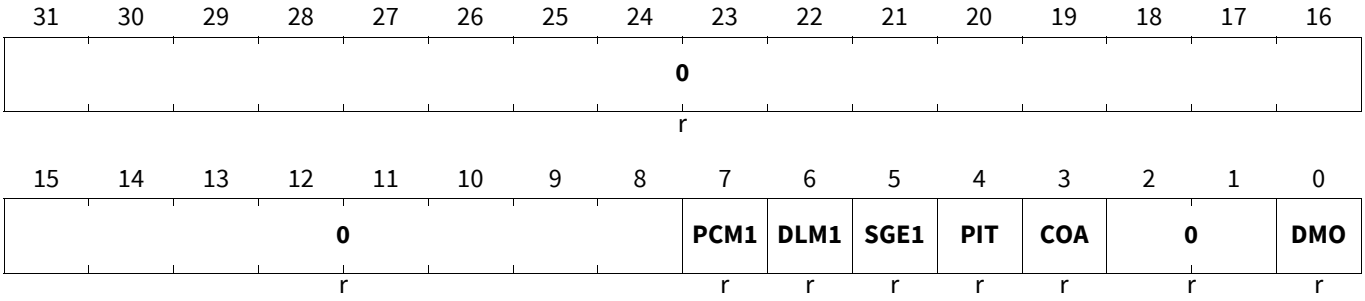
1) Only the values characterized by 1) are stored for an active STATE slope. All other values remain 0. When DEN=0 the relevant bit values of the original register DPLL_CTRL_0 are transferred without any input event at the next system clock.

28.20.12.34Register DPLL_CTRL_1_SHADOW_TRIGGER

DPLL Control 1 Shadow TRIGGER Register

Shadow Register of DPLL_CTRL_1 controlled by an active TRIGGER Slope

DPLL_CTRL_1_SHADOW_TRIGGER
DPLL Control 1 Shadow TRIGGER Register (0281E8$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
DMO	0	r	**DPLL mode select** 1)
COA	3	r	**Correction strategy in automatic end mode (DMO=0)** 1)
PIT	4	r	**Plausibility value PVT to next valid TRIGGER is time related** 1)
SGE1	5	r	**SUB_INC1 generator enable** 1)
DLM1	6	r	**Direct Load Mode for SUB_INC1 generation** 1)

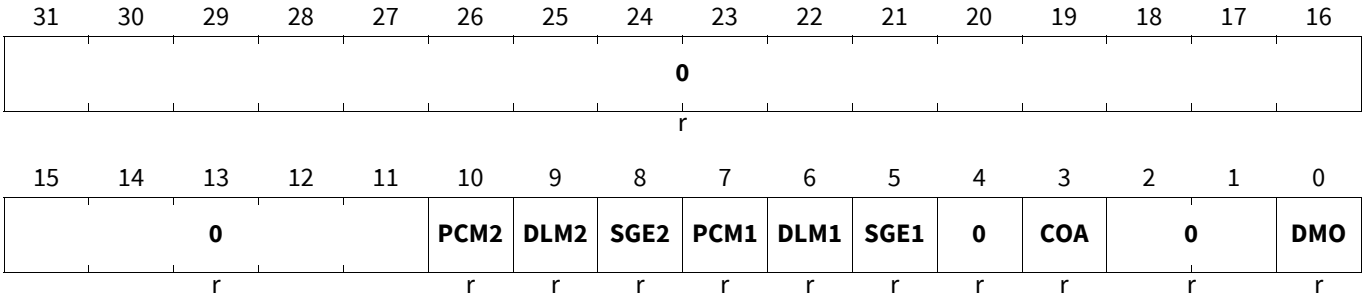
Field	Bits	Type	Description
PCM1	7	r	**Pulse Correction Mode for SUB_INC1 generation** 1)
0	2:1, 31:8	r	**Reserved** Read as zero, shall be written as zero.

1) Only the values characterized by 1) are stored for an active TRIGGER slope. All other values remain 0. When DEN=0 the relevant bit values of the original register DPLL_CTRL_1 are transferred without any input event at the next system clock.

28.20.12.35Register DPLL_CTRL_1_SHADOW_STATE

DPLL Control 1 Shadow STATE Register

DPLL_CTRL_1_SHADOW_STATE
DPLL Control 1 Shadow STATE Register (0281EC$_H$) Application Reset Value: 0000 0000$_H$



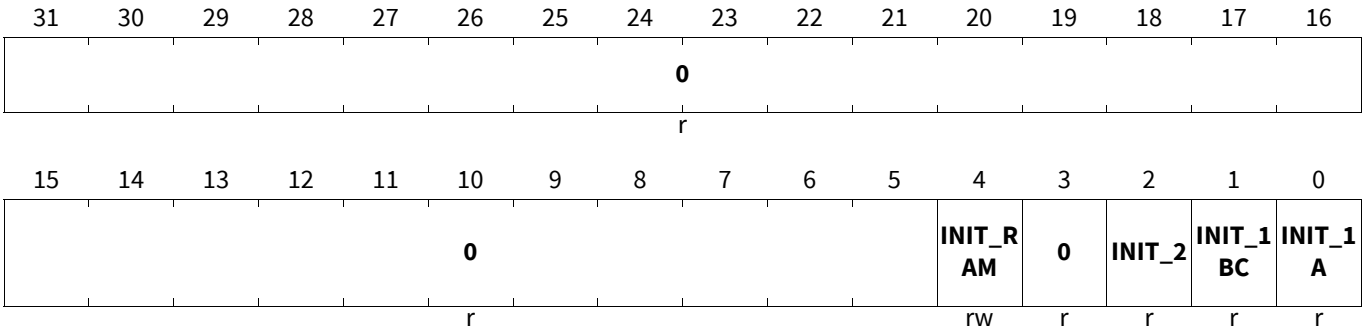
Field	Bits	Type	Description
DMO	0	r	**DPLL mode select** 1)
COA	3	r	**Correction strategy in automatic end mode (DMO=0)** 1)
SGE1	5	r	**SUB_INC1 generator enable** 1)
DLM1	6	r	**Direct Load Mode for SUB_INC1 generation** 1)
PCM1	7	r	**Pulse Correction Mode for SUB_INC1 generation** 1)
SGE2	8	r	**SUB_INC2 generator enable** 1)
DLM2	9	r	**Direct Load Mode for SUB_INC2 generation** 1)
PCM2	10	r	**Pulse Correction Mode for SUB_INC2 generation** 1)
0	2:1, 4, 31:11	r	**Reserved** Read as zero, shall be written as zero.

1) Only the values characterized by 1) are stored for an active STATE slope. All other values remain 0. When DEN=0 the relevant bit values of the original register DPLL_CTRL_1 are transferred without any input event at the next system clock.

28.20.12.36Register DPLL_RAM_INI

DPLL RAM Initialization Register

DPLL_RAM_INI
DPLL RAM Initialization Register **(0281FC$_H$)** **Application Reset Value: 0000 0000$_H$**

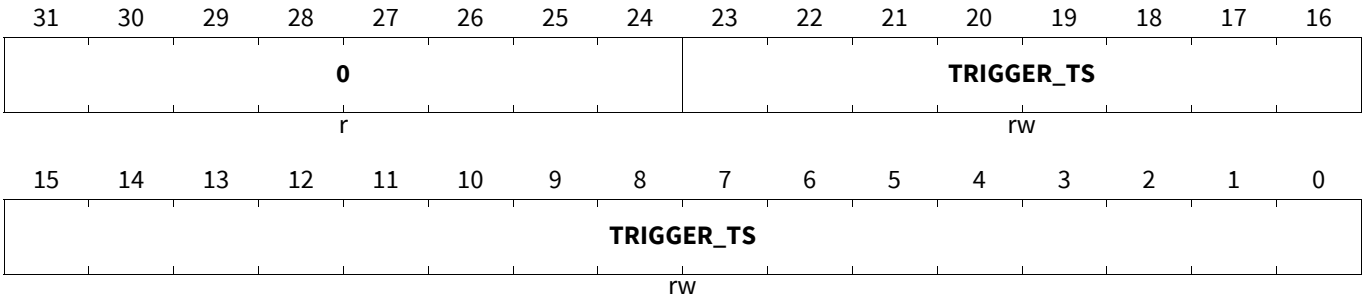


Field	Bits	Type	Description
INIT_1A	0	r	**RAM region 1a initialization in progress** 0_B No initialization of considered RAM region in progress 1_B Initialization of considered RAM region in progress
INIT_1BC	1	r	**RAM region 1b and 1c initialization in progress** Coding see bit 0.
INIT_2	2	r	**RAM region 2 initialization in progress** Coding see bit 0.
INIT_RAM	4	rw	**RAM regions 1a, 1b and 2 are to be initialized** Setting the INIT_RAM bit results only in a RAM reset when the DPLL is not enabled (DEN=0). Depending on the vendor configuration, the connected RAM regions are initialized to zero in the case of a module HW reset or for setting the RST bit in the GTM_RST register. In the case of no RAM initialization, it must be ensured that all relevant parameters are configured correctly. Otherwise, there is no guarantee to get a predictable behavior. 0_B Do not start initialization of all RAM regions 1_B Start initialization of all RAM regions
0	3, 31:5	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.37Memory DPLL_TS_T

DPLL Actual TRIGGER Time Stamp Value

DPLL_TS_T
DPLL Actual TRIGGER Time Stamp Value **(028400_H)** **Reset Value: Table 82**



Field	Bits	Type	Description
TRIGGER_TS	23:0	rw	**Time stamp value of the last active TRIGGER input** Measured TRIGGER time stamp. The LSB address is determined using the SWON_T value in the OSW register (see **Section 28.20.12.8**).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

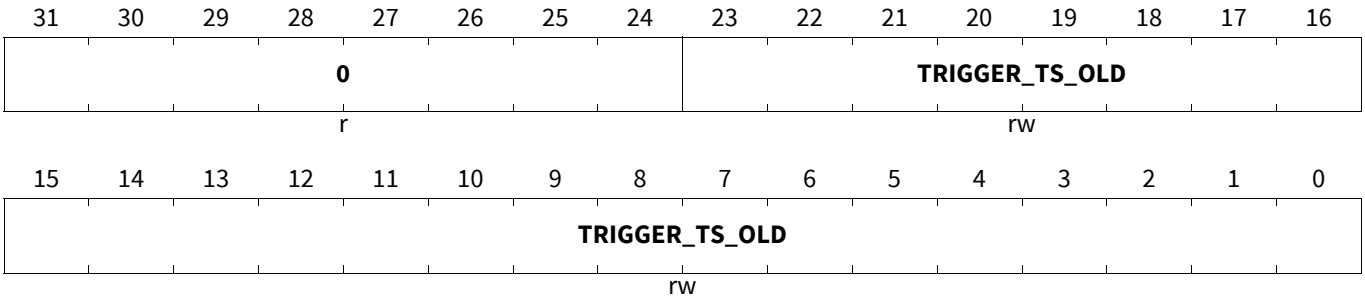
Table 82 Reset Values of DPLL_TS_T

Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000_H	Cleared by state machine

28.20.12.38Memory DPLL_TS_T_OLD

DPLL Previous TRIGGER Time Stamp Value

DPLL_TS_T_OLD
DPLL Previous TRIGGER Time Stamp Value **(028404_H)** **Reset Value: Table 83**



Field	Bits	Type	Description
TRIGGER_TS_OLD	23:0	rw	**Time stamp value of the last but one active TRIGGER input** Previous measured TRIGGER time stamp. The LSB address is determined using the SWON_T value in the OSW register (see **Section 28.20.12.8**).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

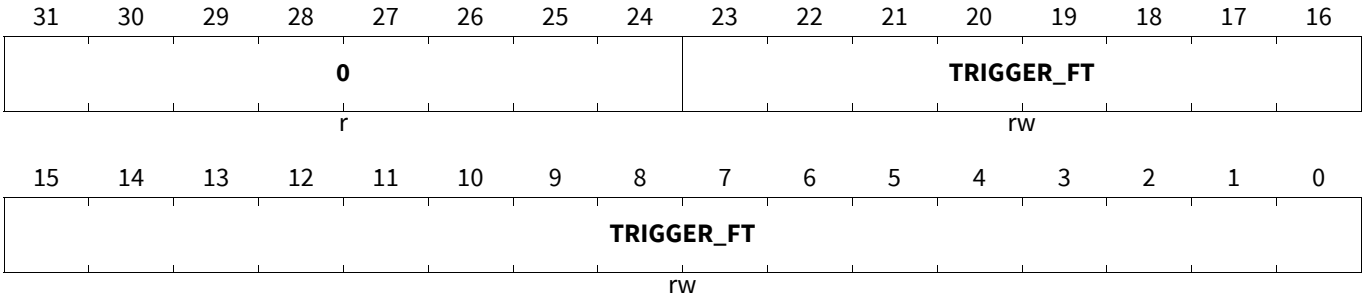
Table 83 Reset Values of DPLL_TS_T_OLD

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.39Memory DPLL_FTV_T

DPLL Actual TRIGGER Filter Value

DPLL_FTV_T
DPLL Actual TRIGGER Filter Value (028408$_H$) Reset Value: Table 84



Field	Bits	Type	Description
TRIGGER_FT	23:0	rw	**Filter value of the last active TRIGGER input** Transmitted filter value. The LSB address is determined using the SWON_T value in the OSW register (see **Section 28.20.12.8**).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

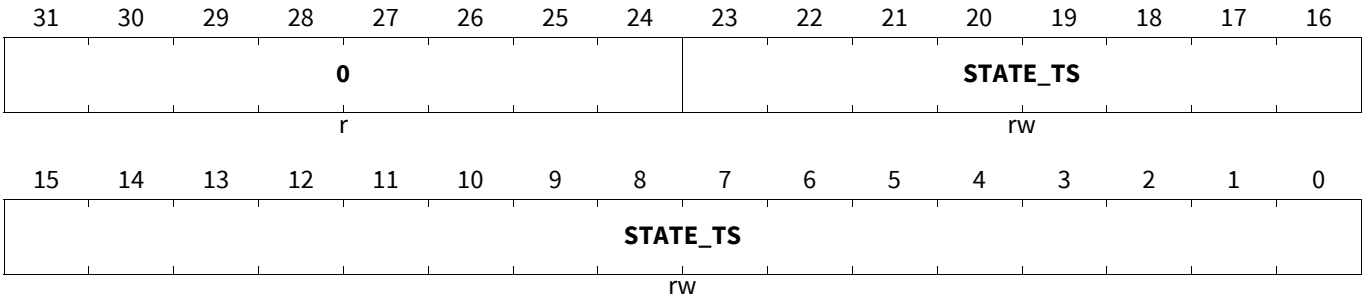
Table 84 Reset Values of DPLL_FTV_T

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.40Memory DPLL_TS_S

DPLL Actual STATE Time Stamp

DPLL_TS_S
DPLL Actual STATE Time Stamp **(028410_H)** **Reset Value: Table 85**



Field	Bits	Type	Description
STATE_TS	23:0	rw	**Time stamp value of the last active STATE input** The LSB address is determined using the SWON_S value in the OSW register (see **Section 28.20.12.8**).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

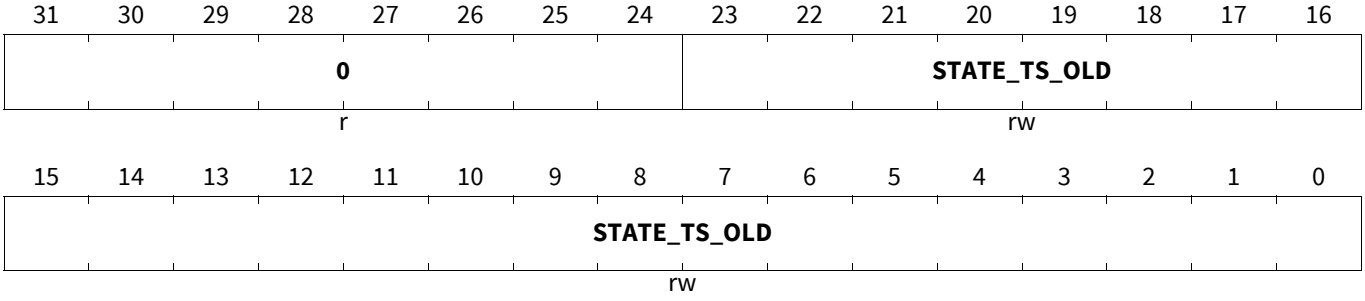
Table 85 Reset Values of DPLL_TS_S

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000$_H$	Cleared by state machine

28.20.12.41Memory DPLL_TS_S_OLD

DPLL Previous STATE Time Stamp

DPLL_TS_S_OLD
DPLL Previous STATE Time Stamp **(028414_H)** **Reset Value: Table 86**



Field	Bits	Type	Description
STATE_TS_OLD	23:0	rw	**Time stamp value of the last active STATE input** The LSB address is determined using the SWON_S value in the OSW register (see **Section 28.20.12.8**).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

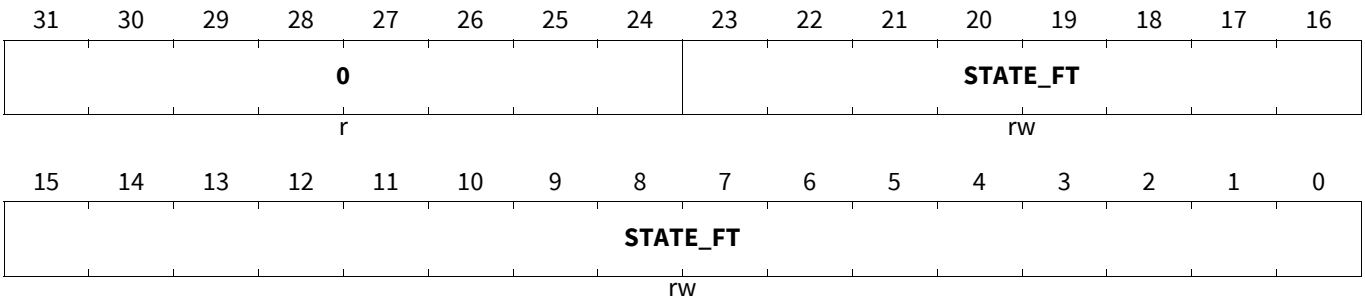
Table 86 Reset Values of DPLL_TS_S_OLD

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.42Memory DPLL_FTV_S

DPLL Actual STATE Filter Value

DPLL_FTV_S
DPLL Actual STATE Filter Value (028418$_H$) Reset Value: Table 87



Field	Bits	Type	Description
STATE_FT	23:0	rw	**Filter value of the last active STATE input** transmitted filter value The LSB address is determined using the SWON_S value in the OSW register (see **Section 28.20.12.8**).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 87 Reset Values of DPLL_FTV_S

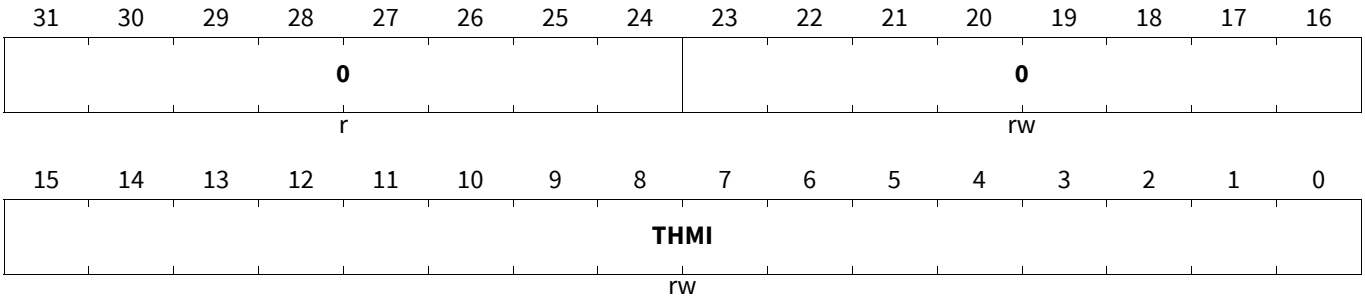
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.43Memory DPLL_THMI

DPLL TRIGGER Hold Time Minimum Value

DPLL_THMI

DPLL TRIGGER Hold Time Minimum Value **(028420$_H$)** **Reset Value: Table 88**



Field	Bits	Type	Description
THMI	15:0	rw	**Minimal time between active and inactive TRIGGER slope (uint16)** The time value corresponds to the time stamp clock counts: this does mean the clock selected for the TBU_CH0_BASE (see TBU_CH0_CTRL register) set min. value; generate the TINI interrupt in the case of a violation for THMI>0. Typical retention time values after an active slope can be e.g. between 45 µs (forwards) and 90 µs (backwards). When THMI is zero, consider always a THMI violation (forwards).
0	23:16	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 88 Reset Values of DPLL_THMI

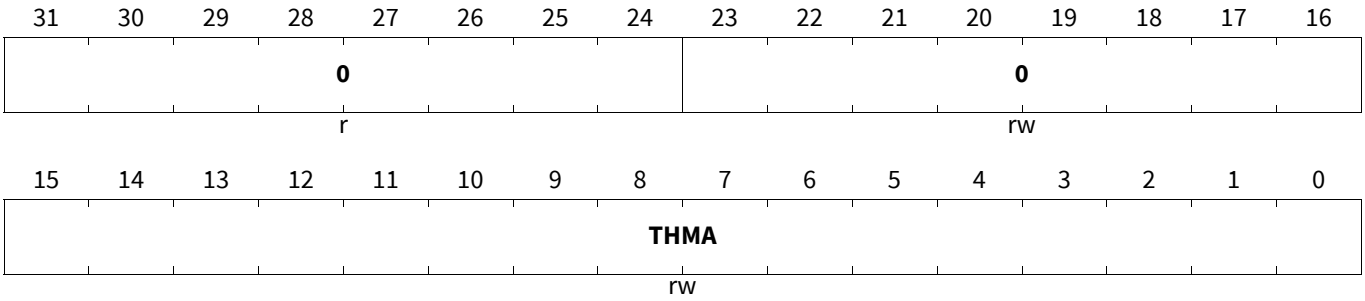
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.44Memory DPLL_THMA

DPLL TRIGGER Hold Time Maximum Value

DPLL_THMA

DPLL TRIGGER Hold Time Maximum Value (028424$_H$) **Reset Value: Table 89**



Field	Bits	Type	Description
THMA	15:0	rw	**Maximal time between active and inactive TRIGGER slope (uint16)** The time value corresponds to the time stamp clock counts: This does mean the clock selected for the TBU_CH0_BASE (see TBU_CH0_CTRL register) Max. value to be set; generate the TAX interrupt in the case of a violation for THMA>0.
0	23:16	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 89 Reset Values of DPLL_THMA

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000$_H$	Cleared by state machine

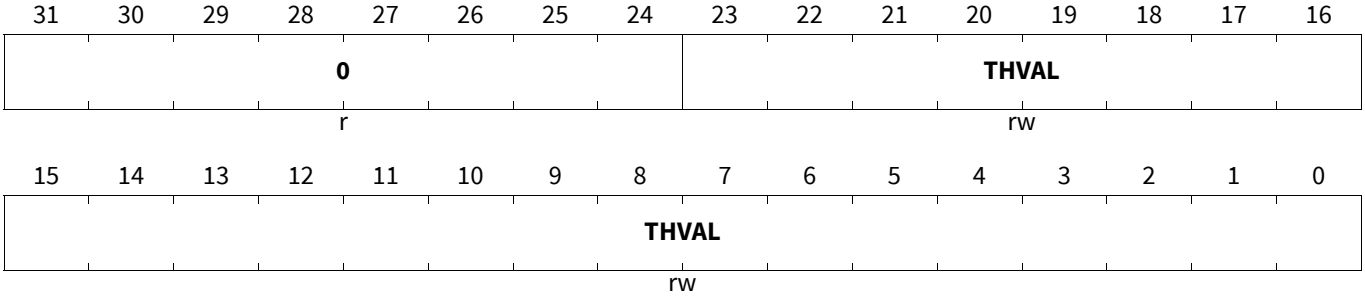
28.20.12.45Memory DPLL_THVAL

DPLL Measured TRIGGER Hold Time Value

Note: In the case of LOW_RES=1 and TBU_HRT=0 the difference between the time stamps of active and inactive slope is multiplied by 8. The register contains this value.

DPLL_THVAL

DPLL Measured TRIGGER Hold Time Value (028428$_H$) **Reset Value: Table 90**



Field	Bits	Type	Description
THVAL	23:0	rw	**Measured time from the last active slope to the next inactive TRIGGER slope in time stamp clock counts: this does mean the clock selected for the TBU_CH0_BASE (uint16)** The measured value considers all input slope filter delays. From the received input the corresponding filter delays are subtracted before the time stamp difference of active and inactive slope is calculated.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 90 Reset Values of DPLL_THVAL

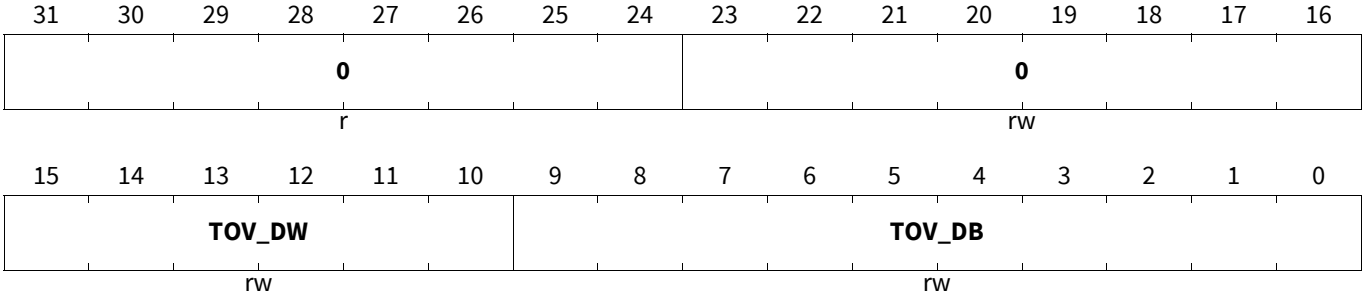
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.46Memory DPLL_TOV

DPLL Time Out Value of Active TRIGGER Slope

DPLL_TOV

DPLL Time Out Value of Active TRIGGER Slope (028430$_H$) **Reset Value: Table 91**



Field	Bits	Type	Description
TOV_DB	9:0	rw	**Decision value (fractional part) for missing TRIGGER interrupt**

Field	Bits	Type	Description
TOV_DW	15:10	rw	**Decision value (integer part) for missing TRIGGER interrupt** TOV(15:0) is to be multiplied with the duration of the last increment and divided by 1024 in order to get the time-out time value for a missing TRIGGER event. For the case of LOW_RES=1 (see DPLL_STATUS register) consider for the calculation of the time out value the following cases: LOW_RES=1 and DPLL_CTRL_1/TS0_HRT=1: multiply the TBU_TS0 value by 8 LOW_RES=1 and DPLL_CTRL_1/TS0_HRT=0: multiply the TBU_TS0 value by 8 multiply the estimated time point value (using TS_T, dt_t_ACT and TOV) by 8 LOW_RES=0 and DPLL_CTRL_1/TS0_HRT=0: use TBU_TS0 and the estimated time point value unchanged.
0	23:16	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 91 Reset Values of DPLL_TOV

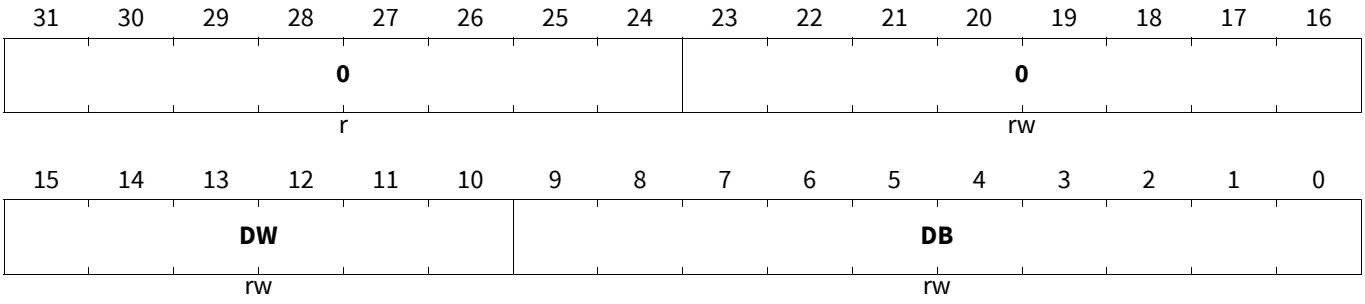
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000$_H$	Cleared by state machine

28.20.12.47Memory DPLL_TOV_S

DPLL Time Out Value of Active STATE Slope

DPLL_TOV_S

DPLL Time Out Value of Active STATE Slope (028434$_H$) **Reset Value: Table 92**



Field	Bits	Type	Description
DB	9:0	rw	**Decision value (fractional part) for missing STATE interrupt**

Field	Bits	Type	Description
DW	15:10	rw	**Decision value (integer part) for missing STATE interrupt** TOV_S (15:0) is to be multiplied with the duration of the last increment and divided by 1024 in order to get the time-out time value for a missing STATE event. For the case of LOW_RES=1 (see DPLL_STATUS register) consider for the calculation of the time out value the following cases: LOW_RES=1 and DPLL_CTRL_1/TS0_HRS=1: multiply the TBU_TS0 value by 8 LOW_RES=1 and DPLL_CTRL_1/TS0_HRS=0: multiply the TBU_TS0 value by 8 multiply the estimated time point value (using TS_T, dt_s_ACT and SOV) by 8 LOW_RES=0 and DPLL_CTRL_1/TS0_HRS=0: use TBU_TS0 and the estimated time point value unchanged.
0	23:16	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 92 Reset Values of DPLL_TOV_S

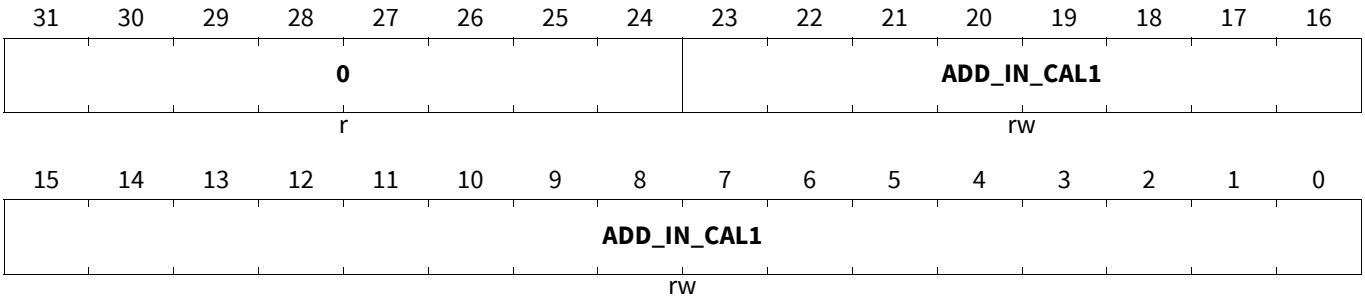
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.48Memory DPLL_ADD_IN_CAL1

DPLL Calculated ADD_IN Value for SUB_INC1 Generation

DPLL_ADD_IN_CAL1

DPLL Calculated ADD_IN Value for SUB_INC1 Generation(028438$_H$) **Reset Value: Table 93**



Field	Bits	Type	Description
ADD_IN_CAL1	23:0	rw	**Calculated input value for SUB_INC1 generation, calculated by the DPLL** Calculated value. The update of the ADD_IN value by the new calculated value ADD_IN_CAL1 is suppressed for one increment when an unexpected missing TRIGGER (SMC=1 or RMO=0) or an unexpected STATE (RMO=1 and SMC=0) is detected.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

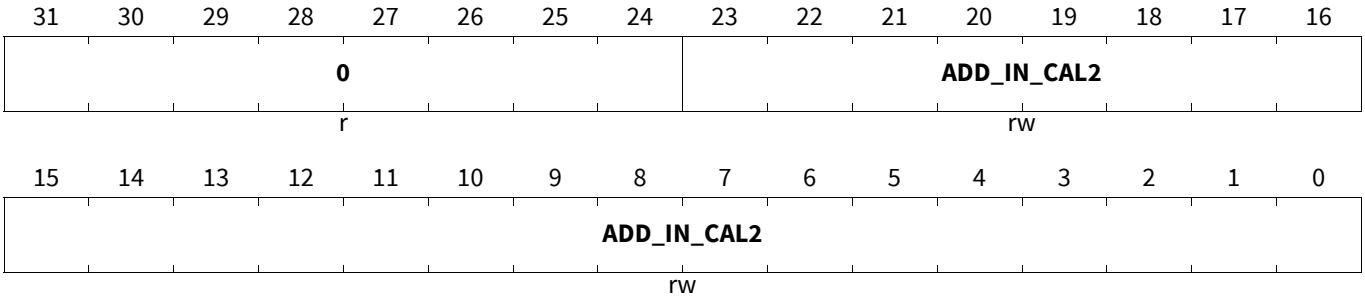
Table 93 Reset Values of DPLL_ADD_IN_CAL1

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.49Memory DPLL_ADD_IN_CAL2

DPLL Calculated ADD_IN Value for SUB_INC2 Generation

DPLL_ADD_IN_CAL2
DPLL Calculated ADD_IN Value for SUB_INC2 Generation(02843C$_H$) **Reset Value: Table 94**



Field	Bits	Type	Description
ADD_IN_CAL2	23:0	rw	**Input value for SUB_INC2 generation, calculated by the DPLL for SMC=RMO=1** Calculated value. The update of the ADD_IN value by the calculated value ADD_IN_CAL2 is suppressed for one increment when an unexpected missing STATE (RMO=SMC=1) is detected.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 94 Reset Values of DPLL_ADD_IN_CAL2

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

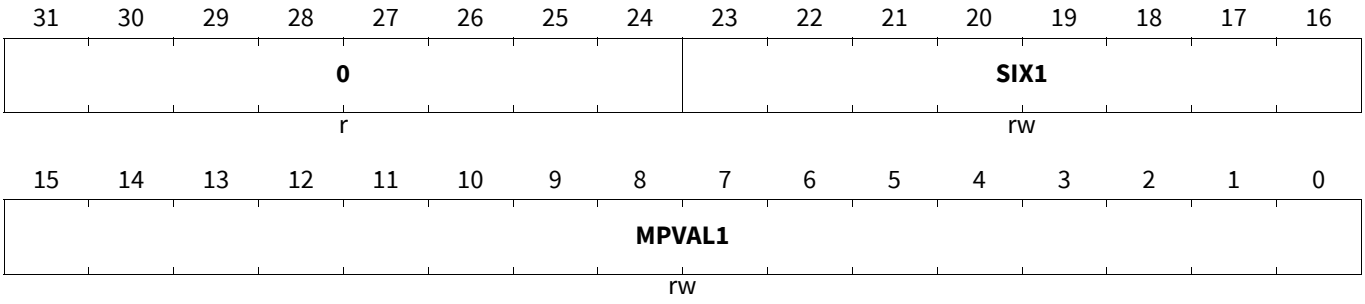
28.20.12.50Memory DPLL_MPVAL1

DPLL Missing Pulses to be Added or Subtracted Directly 1

Note: Do not provide negative values which exceed the amount of NT*(MLT+1) or MLS1 respectively; when considered negative PD values the sum of both (MPVAL1 + NT*PD) should not exceed the amount of NT*(MLT+1) or MLS1 respectively.

DPLL_MPVAL1

DPLL Missing Pulses to be Added or Subtracted Directly 1(028440$_H$) **Reset Value: Table 95**



Field	Bits	Type	Description
MPVAL1	15:0	rw	**Missing pulses for direct correction of SUB_INC1 pulses by the CPU (sint16)** Used only for RMO=0 or SMC=1 for the case PCM1=1. Add MPVAL1 once to INC_CNT1 and reset PCM1 after applying once.
SIX1	23:16	rw	**Sign extension for MPVAL1** All bits must be written to either all zeros or all ones. 00_H MPVAL is a positive number 00_H MPVAL1 is a negative number
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 95 Reset Values of DPLL_MPVAL1

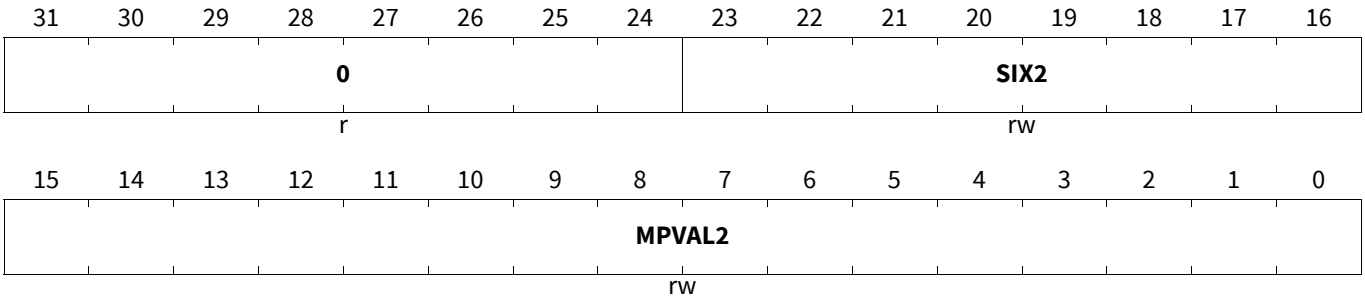
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.51Memory DPLL_MPVAL2

DPLL Missing Pulses to be Added or Subtracted Directly 2

DPLL_MPVAL2

DPLL Missing Pulses to be Added or Subtracted Directly 2(028444_H) **Reset Value: Table 96**



Field	Bits	Type	Description
MPVAL2	15:0	rw	**Missing pulses for direct correction of SUB_INC2 pulses by the CPU (sint16)** Used only for SMC=RMO=1 for the case PCM2=1. Add MPVAL2 once to INC_CNT2, and reset PCM2 after applying once. Do not provide negative values which exceed the amount of MLS2; when considered negative PD_S values, the sum of both should not exceed the amount of MLS2.
SIX2	23:16	rw	**Sign extension for MPVAL2** All bits must be written to either all zeros or all ones. 00_H MPVAL2 is a positive number FF_H MPVAL2 is a negative number
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 96 Reset Values of DPLL_MPVAL2

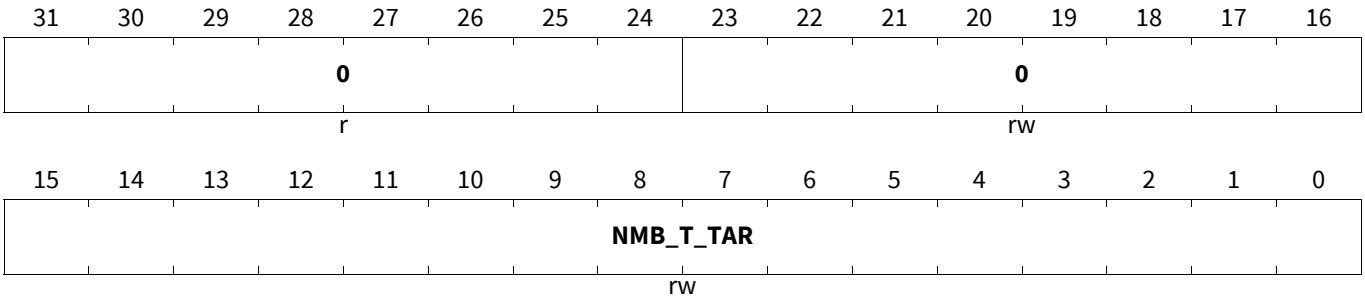
Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000_H	Cleared by state machine

28.20.12.52Memory DPLL_NMB_T_TAR

DPLL Target Number of Pulses to be Sent in Normal Mode

DPLL_NMB_T_TAR

DPLL Target Number of Pulses to be Sent in Normal Mode(028448$_H$) **Reset Value: Table 97**



Field	Bits	Type	Description
NMB_T_TAR	15:0	rw	**Target Number of pulses for TRIGGER** Calculated number of pulses in normal mode for the current *TRIGGER* increment without missing pulses. Calculated target pulse number. The LSB address is determined using the SWON_T value in the OSW register (see **Section 28.20.12.8**).
0	23:16	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 97 Reset Values of DPLL_NMB_T_TAR

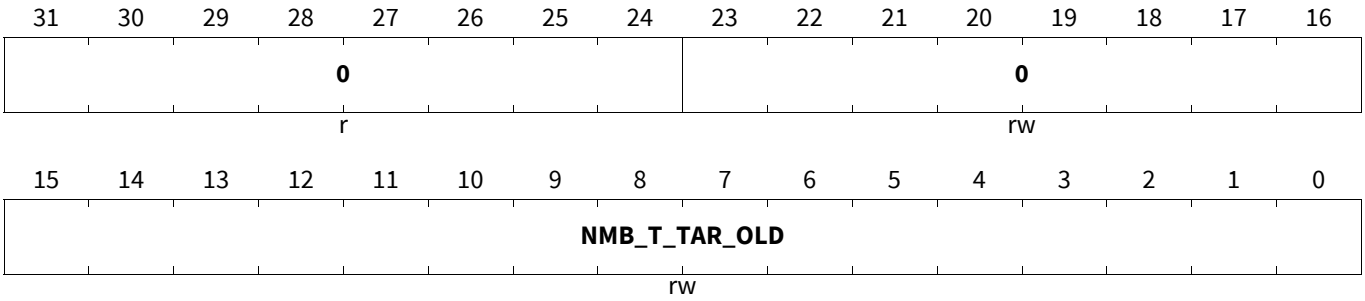
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000$_H$	Cleared by state machine

28.20.12.53Memory DPLL_NMB_T_TAR_OLD

DPLL Last but One Target Number of Pulses to be Sent in Normal Mode

DPLL_NMB_T_TAR_OLD

DPLL Last but One Target Number of Pulses to be Sent in Normal Mode(02844C$_H$) **Reset Value: Table 98**



Field	Bits	Type	Description
NMB_T_TAR_OLD	15:0	rw	**Target Number of pulses for TRIGGER** Calculated number of pulses in normal mode for the current *TRIGGER* increment without missing pulses. Calculated target pulse number. The LSB address is determined using the SWON_T value in the OSW register (see **Section 28.20.12.8**).
0	23:16	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 98 Reset Values of DPLL_NMB_T_TAR_OLD

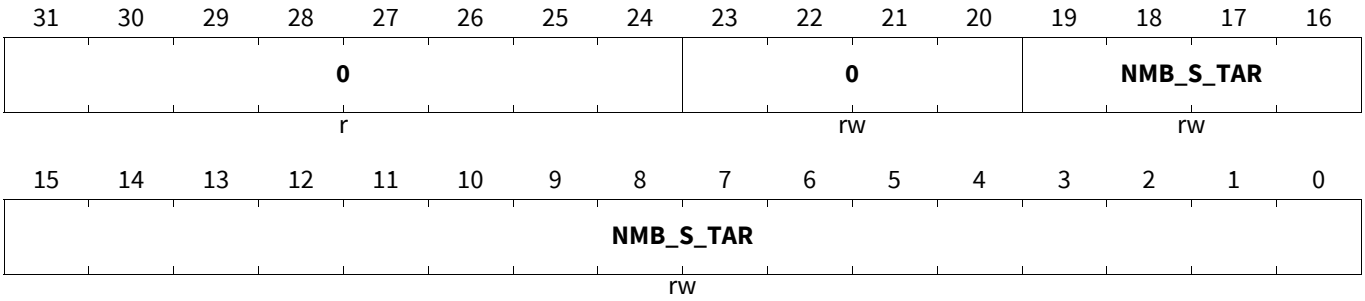
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000$_H$	Cleared by state machine

28.20.12.54Memory DPLL_NMB_S_TAR

DPLL Target Number of Pulses to be Sent in Emergency Mode

DPLL_NMB_S_TAR

DPLL Target Number of Pulses to be Sent in Emergency Mode(028450_H) **Reset Value: Table 99**



Field	Bits	Type	Description
NMB_S_TAR	19:0	rw	**Target Number of pulses for STATE** Calculated number of pulses in emergency mode for the current *STATE* increment without missing pulses. Calculated target pulse number. The LSB address is determined using the SWON_S value in the OSW register (see **Section 28.20.12.8**).
0	23:20	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

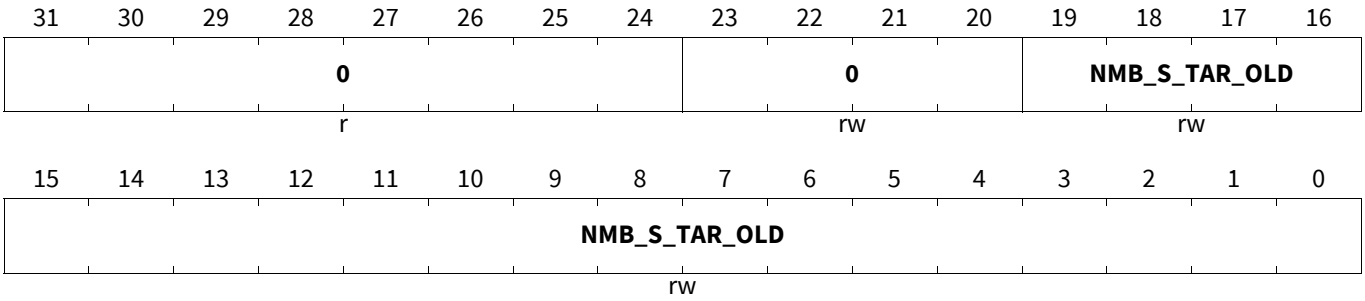
Table 99 Reset Values of DPLL_NMB_S_TAR

Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	––00 0000_H	Cleared by state machine

28.20.12.55Memory DPLL_NMB_S_TAR_OLD

DPLL Last but One Target Number of Pulses to be Sent in Emergency Mode

DPLL_NMB_S_TAR_OLD
DPLL Last but One Target Number of Pulses to be Sent in Emergency Mode(028454_H) **Reset Value:**
Table 100



Field	Bits	Type	Description
NMB_S_TAR_OLD	19:0	rw	**Target Number of pulses for STATE** Calculated number of pulses in emergency mode for the current *STATE* increment without missing pulses. Calculated target pulse number. The LSB address is determined using the SWON_S value in the OSW register (see **Section 28.20.12.8**).
0	23:20	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 100 Reset Values of DPLL_NMB_S_TAR_OLD

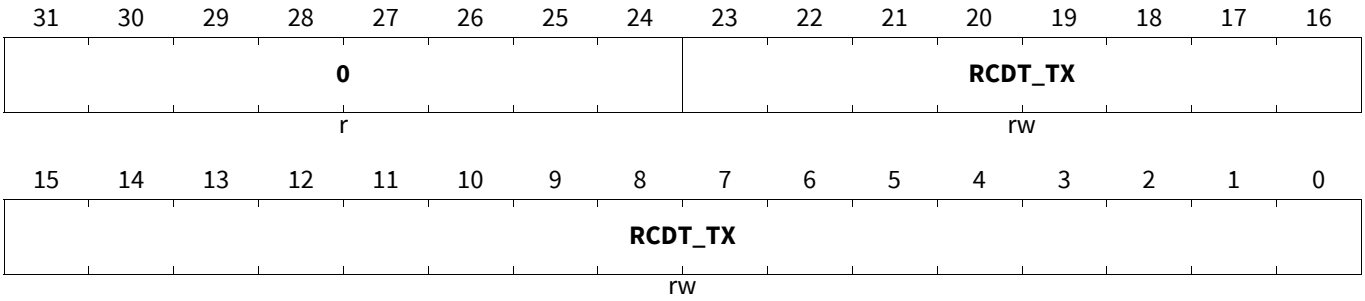
Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000_H	Cleared by state machine

28.20.12.56Memory DPLL_RCDT_TX

DPLL Reciprocal Value of the Expected Increment Duration of TRIGGER

DPLL_RCDT_TX

DPLL Reciprocal Value of the Expected Increment Duration of TRIGGER(028460_H) Reset Value: Table 101



Field	Bits	Type	Description
RCDT_TX	23:0	rw	**Reciprocal value of expected increment duration *2^32 while only the lower 24 bits are used** Calculated value; when an overflow occurs in the calculation, the value is set to 0xFFFFFF.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 101 Reset Values of DPLL_RCDT_TX

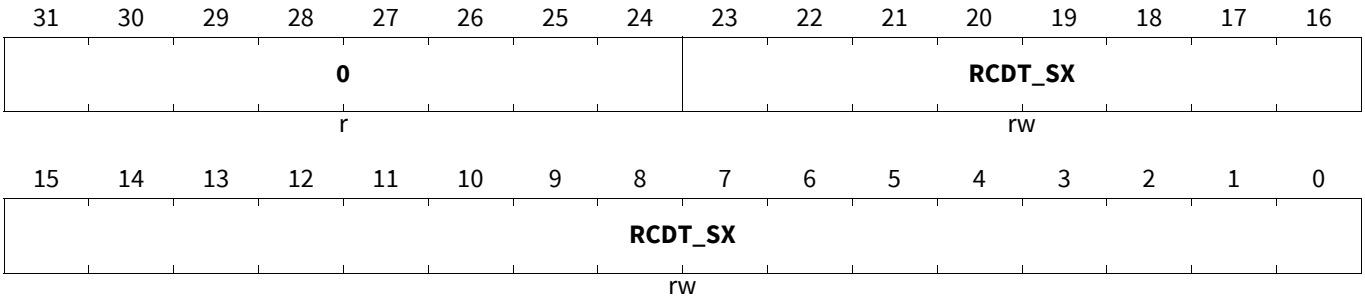
Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000_H	Cleared by state machine

28.20.12.57Memory DPLL_RCDT_SX

DPLL Reciprocal Value of the Expected Increment Duration of STATE

DPLL_RCDT_SX

DPLL Reciprocal Value of the Expected Increment Duration of STATE(028464_H) Reset Value: Table 102



Field	Bits	Type	Description
RCDT_SX	23:0	rw	**Reciprocal value of expected increment duration *2^32 while only the lower 24 bits are used** Calculated value; when an overflow occurs in the calculation, the value is set to 0xFFFFFF.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

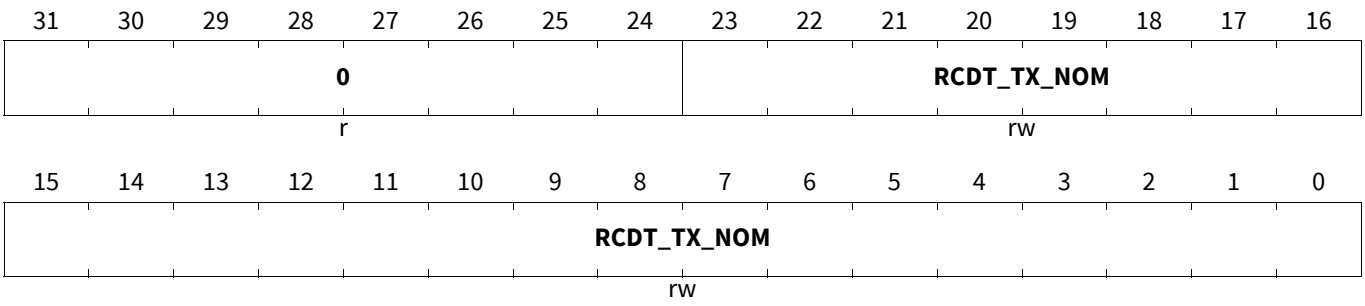
Table 102 Reset Values of DPLL_RCDT_SX

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.58Memory DPLL_RCDT_TX_NOM

DPLL Reciprocal Value of the Expected Nominal Increment Duration of TRIGGER

DPLL_RCDT_TX_NOM
DPLL Reciprocal Value of the Expected Nominal Increment Duration of TRIGGER(028468$_H$) Reset Value: Table 103



Field	Bits	Type	Description
RCDT_TX_NOM	23:0	rw	**Reciprocal value of nominal increment duration *2^32 while only the lower 24 bits are used** Calculated value; when an overflow occurs in the calculation, the value is set to 0xFFFFFF.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 103 Reset Values of DPLL_RCDT_TX_NOM

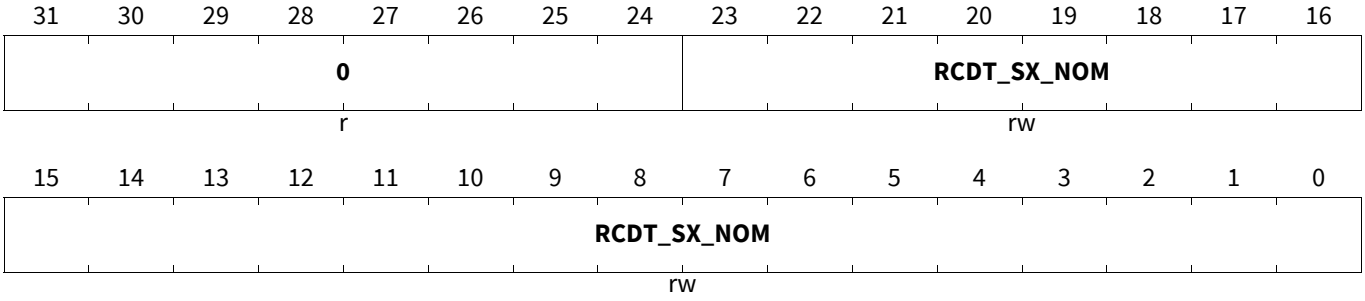
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.59Memory DPLL_RCDT_SX_NOM

DPLL Reciprocal Value of the Expected Nominal Increment Duration of STATE

Note: RCDT_TX_NOM and RCDT_SX_NOM are calculated by the values RCDT_TX and RCDT_SX to be multiplied with SYN_T or SYN_S respectively.

DPLL_RCDT_SX_NOM
DPLL Reciprocal Value of the Expected Nominal Increment Duration of STATE(02846C$_H$) **Reset Value: Table 104**



Field	Bits	Type	Description
RCDT_SX_NOM	23:0	rw	**Reciprocal value of nominal increment duration *2^32 while only the lower 24 bits are used** Calculated value; when an overflow occurs in the calculation, the value is set to 0xFFFFFF.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 104 Reset Values of DPLL_RCDT_SX_NOM

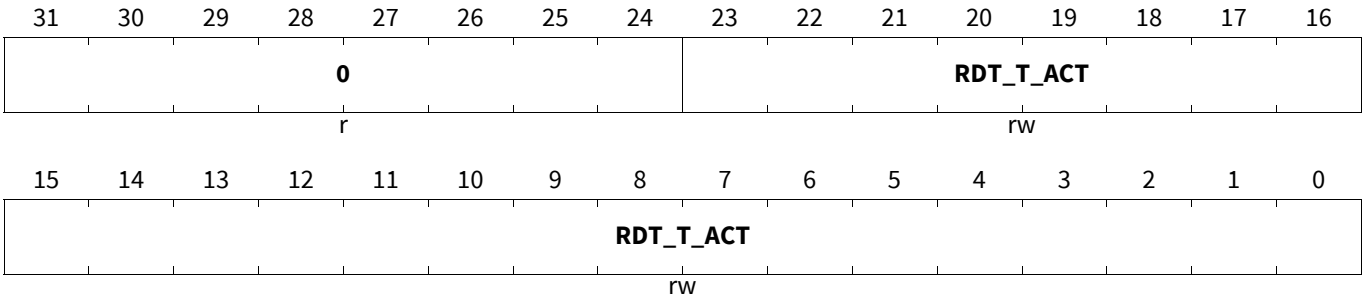
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.60Memory DPLL_RDT_T_ACT

DPLL Reciprocal Value of the Last Increment of TRIGGER

DPLL_RDT_T_ACT

DPLL Reciprocal Value of the Last Increment of TRIGGER(028470_H) **Reset Value: Table 105**



Field	Bits	Type	Description
RDT_T_ACT	23:0	rw	**Reciprocal value of last TRIGGER increment *2^32, only the lower 24 bits are used** The LSB is rounded up when the next truncated bit is 1. Calculated value; when an overflow occurs in the calculation, the value is set to 0xFFFFFF and the CRO bit in the DPLL_STATUS register is set (see **Section 28.20.12.30**).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 105 Reset Values of DPLL_RDT_T_ACT

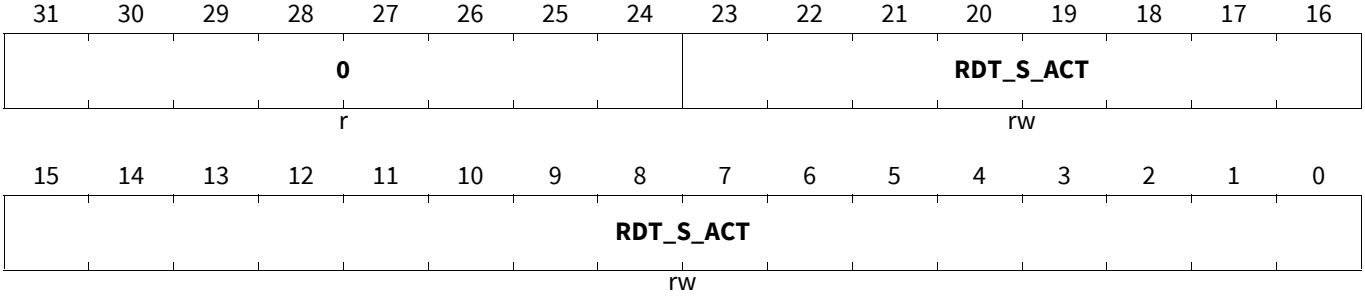
Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000_H	Cleared by state machine

28.20.12.61Memory DPLL_RDT_S_ACT

DPLL Reciprocal Value of the Last Increment of STATE

DPLL_RDT_S_ACT

DPLL Reciprocal Value of the Last Increment of STATE(028474_H) **Reset Value: Table 106**



Field	Bits	Type	Description
RDT_S_ACT	23:0	rw	**Reciprocal value of last STATE increment *2^32, only the lower 24 bits are used** The LSB is rounded up when the next truncated bit is 1. Calculated value; when an overflow occurs in the calculation, the value is set to 0xFFFFFF and the CRO bit in the DPLL_STATUS register is set (see **Section 28.20.12.30**).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 106 Reset Values of DPLL_RDT_S_ACT

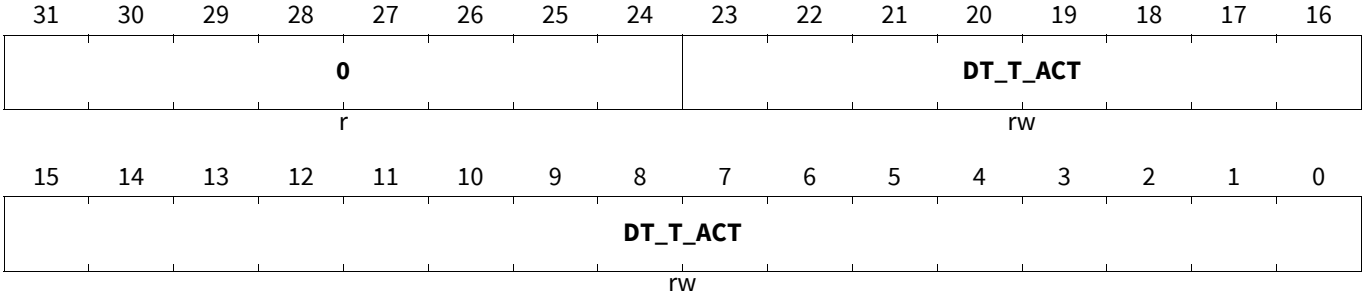
Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000_H	Cleared by state machine

28.20.12.62Memory DPLL_DT_T_ACT

DPLL Duration of the Last TRIGGER Increment

DPLL_DT_T_ACT

DPLL Duration of the Last TRIGGER Increment (028478_H) **Reset Value: Table 107**



Field	Bits	Type	Description
DT_T_ACT	23:0	rw	**Calculated duration of the last TRIGGER increment** Calculated duration of the last increment; Value will be written into the corresponding RAM field, when all calculations for the considered increment are done and APT is valid.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

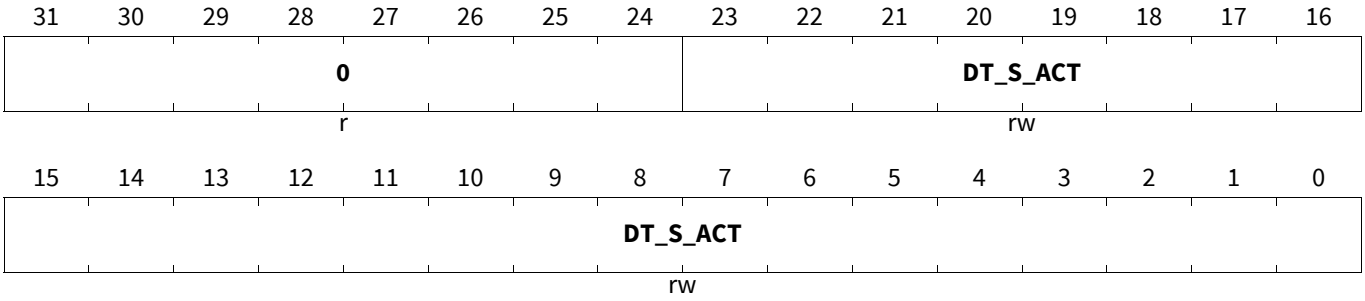
Table 107 Reset Values of DPLL_DT_T_ACT

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.63Memory DPLL_DT_S_ACT

DPLL Duration of the Last STATE Increment

DPLL_DT_S_ACT
DPLL Duration of the Last STATE Increment (02847C$_H$) **Reset Value: Table 108**



Field	Bits	Type	Description
DT_S_ACT	23:0	rw	**Calculated duration of the last STATE increment** Calculated increment duration. Value will be written into the corresponding RAM field, when all calculations for the considered increment are done and APS is valid.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 108 Reset Values of DPLL_DT_S_ACT

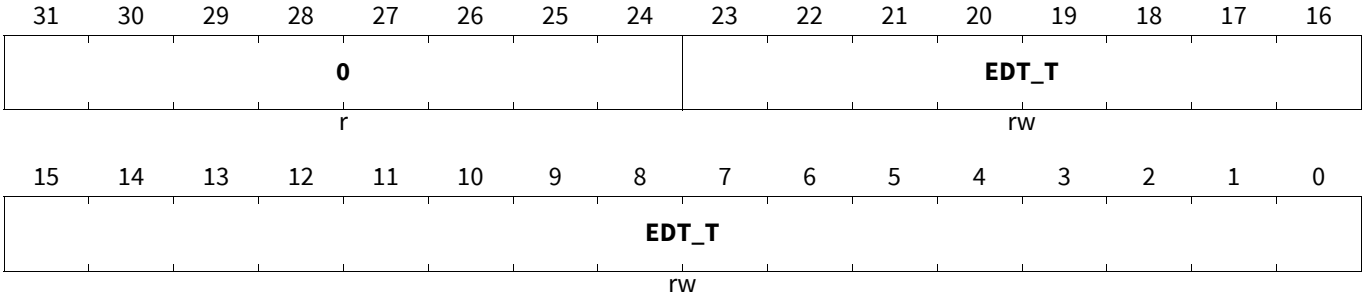
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.64Memory DPLL_EDT_T

DPLL Difference of Prediction to Actual Value of the Last TRIGGER Increment

DPLL_EDT_T

DPLL Difference of Prediction to Actual Value of the Last TRIGGER Increment(028480_H) **Reset Value: Table 109**



Field	Bits	Type	Description
EDT_T	23:0	rw	**Signed difference between actual value and a simple prediction of the last TRIGGER increment: sint24** Calculated error value.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 109 Reset Values of DPLL_EDT_T

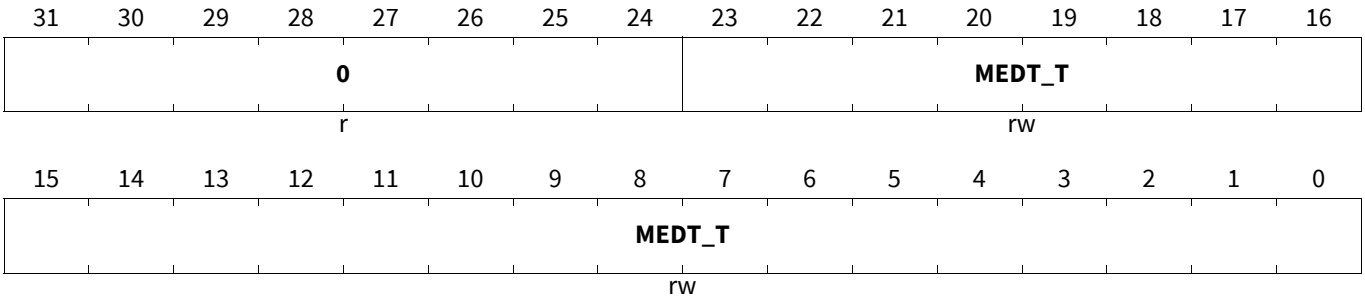
Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000_H	Cleared by state machine

28.20.12.65Memory DPLL_MEDT_T

DPLL Weighted Difference of Prediction Errors of TRIGGER

DPLL_MEDT_T

DPLL Weighted Difference of Prediction Errors of TRIGGER(028484_H) **Reset Value: Table 110**



Field	Bits	Type	Description
MEDT_T	23:0	rw	**Signed middle weighted difference between actual value and prediction of the last TRIGGER increments: sint24** Only calculated for SYT=1. Calculated medium error value, see **Section 28.20.6.2.6**. The value is calculated only after synchronization (SYT=1), and the update is suppressed for one increment when an unexpected missing TRIGGER is detected.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

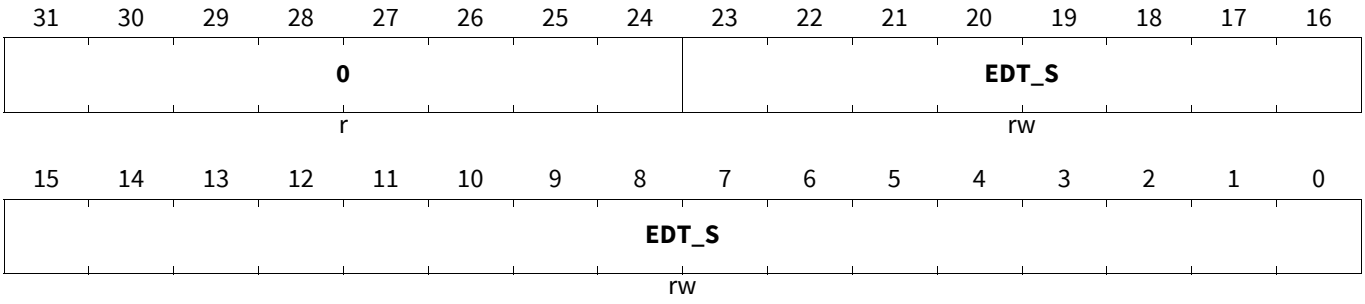
Table 110 Reset Values of DPLL_MEDT_T

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000$_H$	Cleared by state machine

28.20.12.66Memory DPLL_EDT_S

DPLL Difference of Prediction to Actual Value of the Last STATE Increment

DPLL_EDT_S
DPLL Difference of Prediction to Actual Value of the Last STATE Increment(028488$_H$) Reset Value: Table 111



Field	Bits	Type	Description
EDT_S	23:0	rw	**Signed difference between actual value and prediction of the last STATE increment: sint24** Calculated error value, see **Section 28.20.6.3.5**.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 111 Reset Values of DPLL_EDT_S

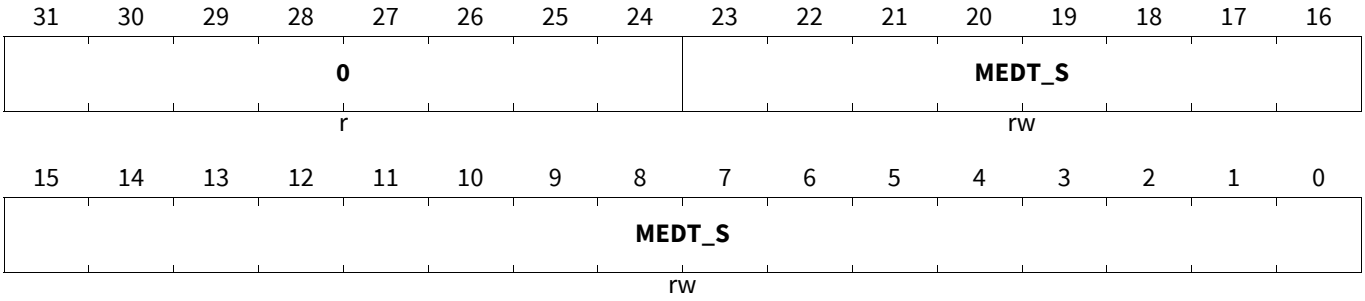
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.67Memory DPLL_MEDT_S

DPLL Weighted Difference of Prediction Errors of STATE

DPLL_MEDT_S

DPLL Weighted Difference of Prediction Errors of STATE(02848C$_H$) **Reset Value: Table 112**



Field	Bits	Type	Description
MEDT_S	23:0	rw	**Signed middle weighted difference between actual value and prediction of the last STATE increments: sint24; only calculated for SYS=1** Calculated medium error value, see **Section 28.20.6.3.6**. The value is calculated only after synchronization (SYS=1), and the update is suppressed for one increment when an unexpected missing STATE is detected.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 112 Reset Values of DPLL_MEDT_S

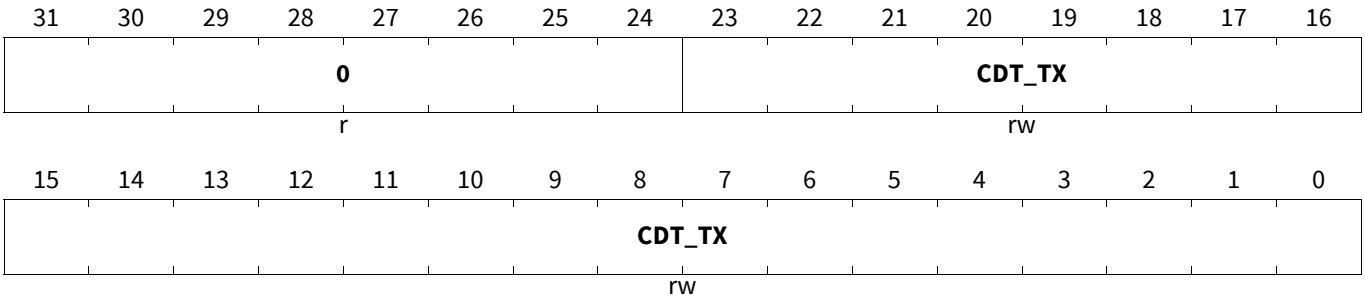
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.68Memory DPLL_CDT_TX

DPLL Prediction of the Actual TRIGGER Increment Duration

DPLL_CDT_TX

DPLL Prediction of the Actual TRIGGER Increment Duration(028490_H) **Reset Value: Table 113**



Field	Bits	Type	Description
CDT_TX	23:0	rw	**Calculated duration of the current TRIGGER increment** Calculated value.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 113 Reset Values of DPLL_CDT_TX

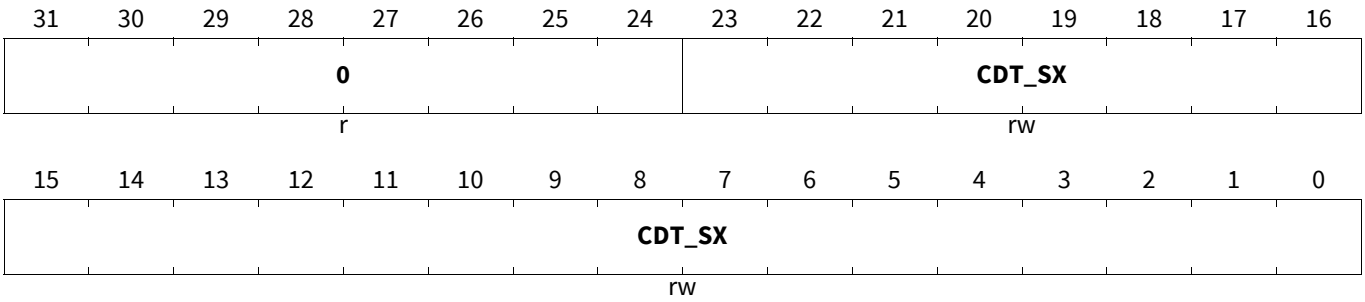
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000$_H$	Cleared by state machine

28.20.12.69Memory DPLL_CDT_SX

DPLL Prediction of the Actual STATE Increment Duration

DPLL_CDT_SX

DPLL Prediction of the Actual STATE Increment Duration(028494_H) **Reset Value: Table 114**



Field	Bits	Type	Description
CDT_SX	23:0	rw	**Calculated duration of the current STATE increment** Calculated value.

Field	Bits	Type	Description
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 114 Reset Values of DPLL_CDT_SX

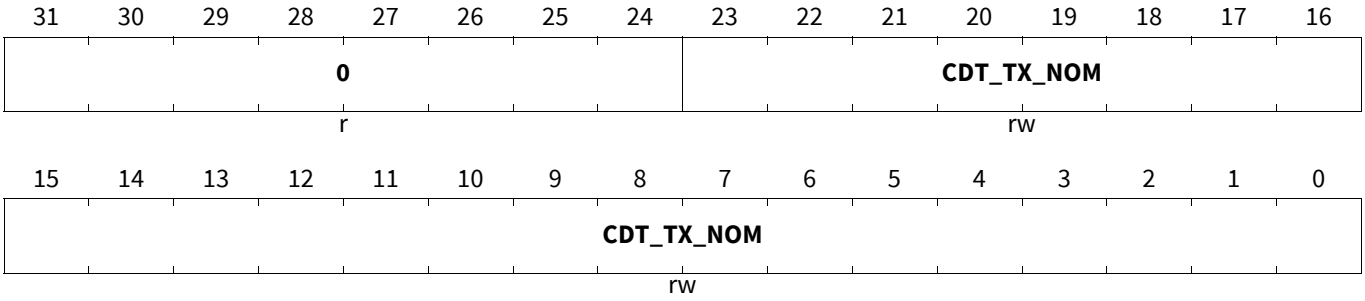
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.70Memory DPLL_CDT_TX_NOM

DPLL Prediction of the Nominal TRIGGER Increment Duration

DPLL_CDT_TX_NOM

DPLL Prediction of the Nominal TRIGGER Increment Duration(028498$_H$) **Reset Value: Table 115**



Field	Bits	Type	Description
CDT_TX_NOM	23:0	rw	**Calculated duration of the current nominal TRIGGER event** Calculated value.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 115 Reset Values of DPLL_CDT_TX_NOM

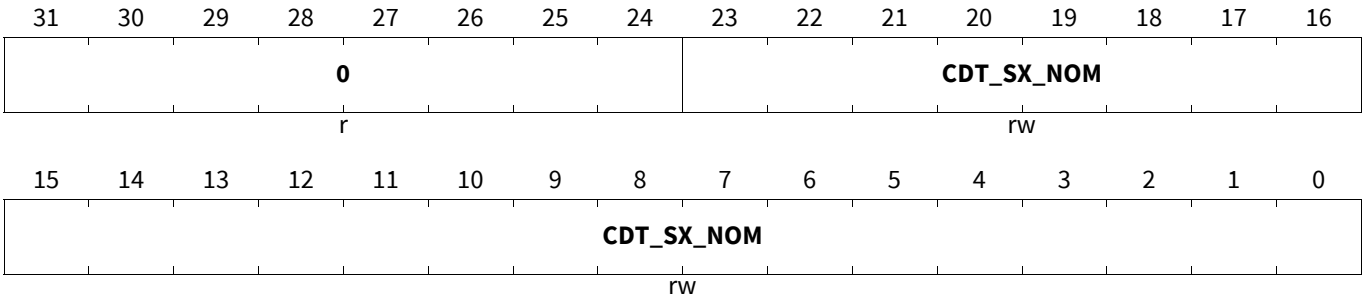
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.71Memory DPLL_CDT_SX_NOM

DPLL Prediction of the Nominal STATE Increment Duration

DPLL_CDT_SX_NOM

DPLL Prediction of the Nominal STATE Increment Duration(02849C$_H$) **Reset Value: Table 116**



Field	Bits	Type	Description
CDT_SX_NOM	23:0	rw	**Calculated duration of the current nominal STATE event** Calculated value.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 116 Reset Values of DPLL_CDT_SX_NOM

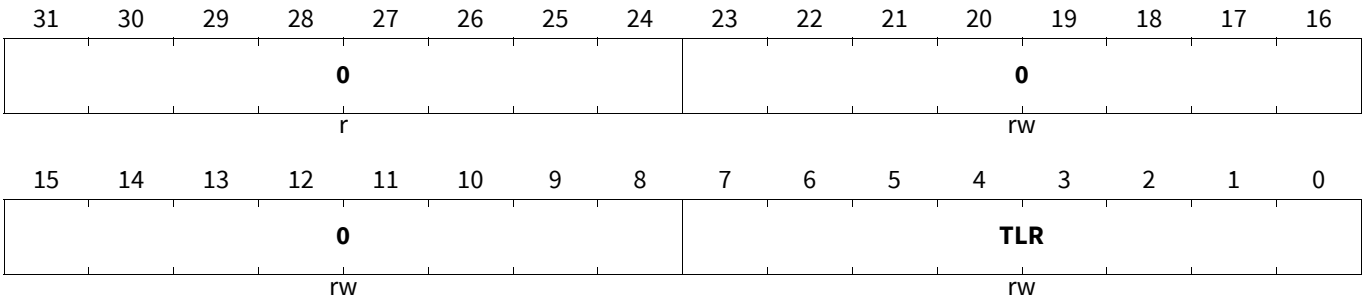
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.72Memory DPLL_TLR

DPLL TRIGGER Locking Range

DPLL_TLR

DPLL TRIGGER Locking Range (0284A0$_H$) **Reset Value: Table 117**



Field	Bits	Type	Description
TLR	7:0	rw	**Value is to be multiplied with the last nominal TRIGGER duration in order to get the range for the next TRIGGER event without setting TOR in the DPLL_STATUS register** Multiply value with the last nominal increment duration and check violation; when TLR=0 don't perform the check.
0	23:8	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

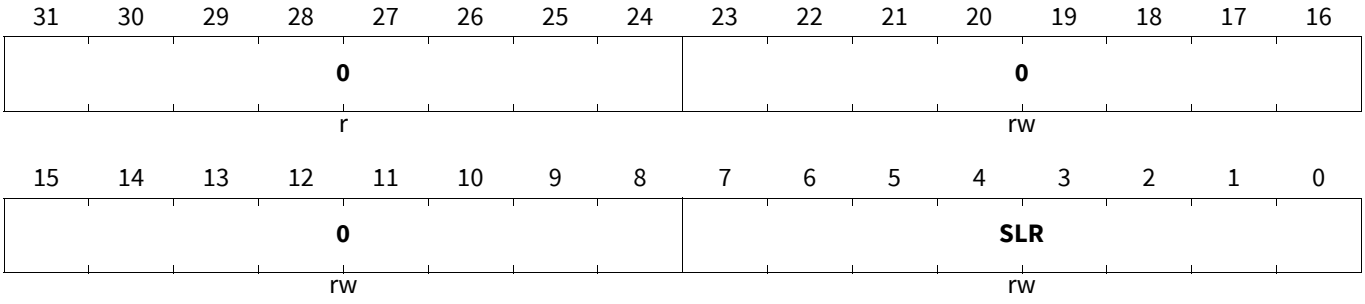
Table 117 Reset Values of DPLL_TLR

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000$_H$	Cleared by state machine

28.20.12.73Memory DPLL_SLR

DPLL STATE Locking Range

DPLL_SLR
DPLL STATE Locking Range **(0284A4$_H$)** **Reset Value: Table 118**



Field	Bits	Type	Description
SLR	7:0	rw	**Value is to be multiplied with the last nominal STATE duration in order to get the range for the next STATE event without setting SOR in the DPLL_STATUS register** Multiply value with the last nominal increment duration and check violation; when SLR=0 don't perform the check.
0	23:8	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

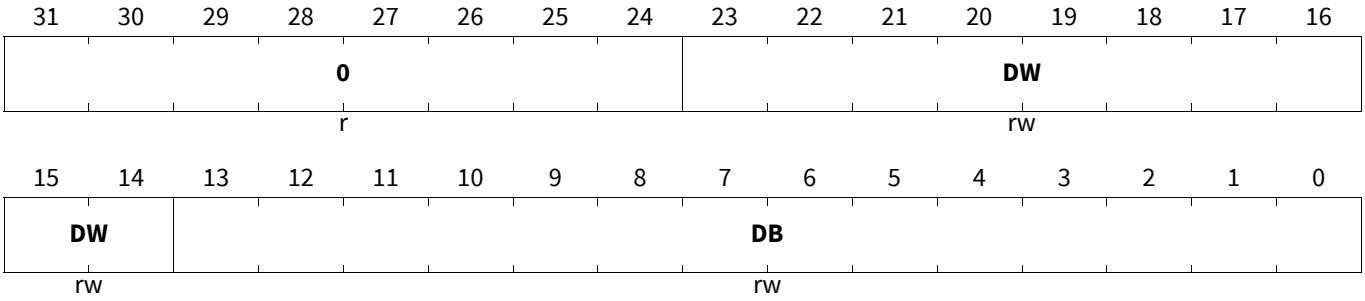
Table 118 Reset Values of DPLL_SLR

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.74Memory DPLL_PDT_[z]

DPLL Projected Increment Sum Relations for Action z

DPLL_PDT_z (z=0-31)
DPLL Projected Increment Sum Relations for Action z(028500$_H$+z*4) **Reset Value: Table 119**



Field	Bits	Type	Description
DB	13:0	rw	**Fractional part of relation between TRIGGER or STATE increments**
DW	23:14	rw	**Integer part of relation between TRIGGER or STATE increments** Definition of relation values between *TRIGGER* or *STATE* increments PDT[i] according to Equations DPLL-11 or DPLL-13 (i = 0-31).[1)] The PDT[i] values for actions i=24...31 are only available for device 4 or 5.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

1) The PDT[z] values for actions i=24...31 are not available for all devices.

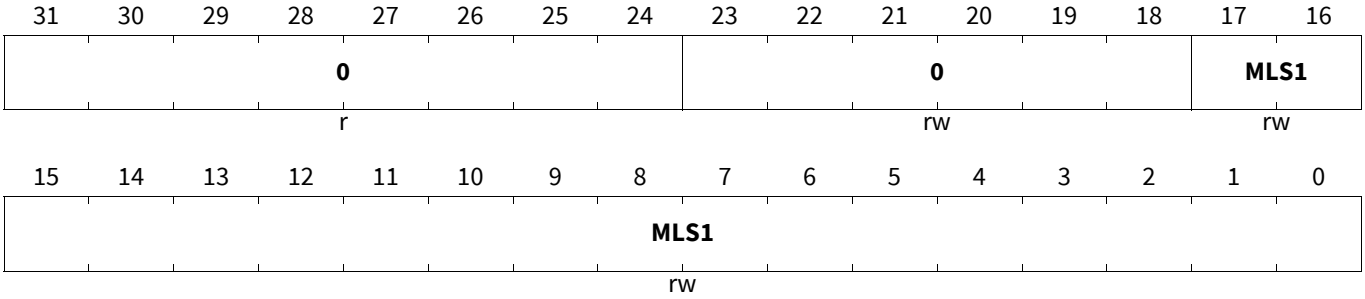
Table 119 Reset Values of DPLL_PDT_z (z=0-31)

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.75Memory DPLL_MLS1

DPLL Calculated Number of Sub-Pulses between two Nominal STATE Events for SMC = 0

DPLL_MLS1
DPLL Calculated Number of Sub-Pulses between two Nominal STATE Events for SMC = 0(0285C0$_H$) Reset Value: Table 120



Field	Bits	Type	Description
MLS1	17:0	rw	**Number of pulses between two STATE events** For SMC=0 the value of MLS1 is calculated once by the CPU for fixed values in the DPLL_CTRL_0 register by the formula MLS1 = ((MLT+1)*(TNU+1)/(SNU+1)) and set accordingly. For SMC=1 the value of MLS1 represents the number of pulses between two nominal *TRIGGER* events (to be set and updated by the CPU).
0	23:18	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

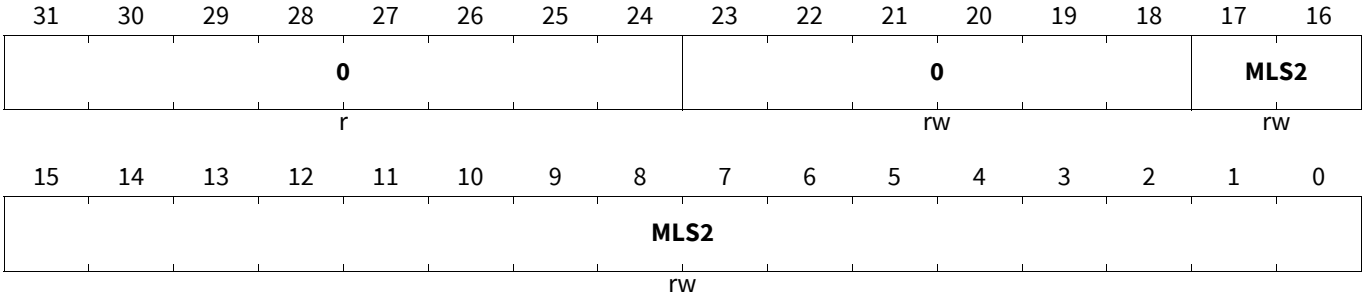
Table 120 Reset Values of DPLL_MLS1

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.76Memory DPLL_MLS2

DPLL Calculated Number of Sub-Pulses between two Nominal STATE Events for SMC = 1 and RMO = 1

DPLL_MLS2
DPLL Calculated Number of Sub-Pulses between two Nominal STATE Events for SMC = 1 and RMO = 1 ($0285C4_H$) **Reset Value: Table 121**



Field	Bits	Type	Description
MLS2	17:0	rw	**Number of pulses between two STATE events (to be set and updated by the CPU)** Using adapt information and the missing *STATE* event information SYN_S, this value can be corrected for each increment automatically.
0	23:18	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

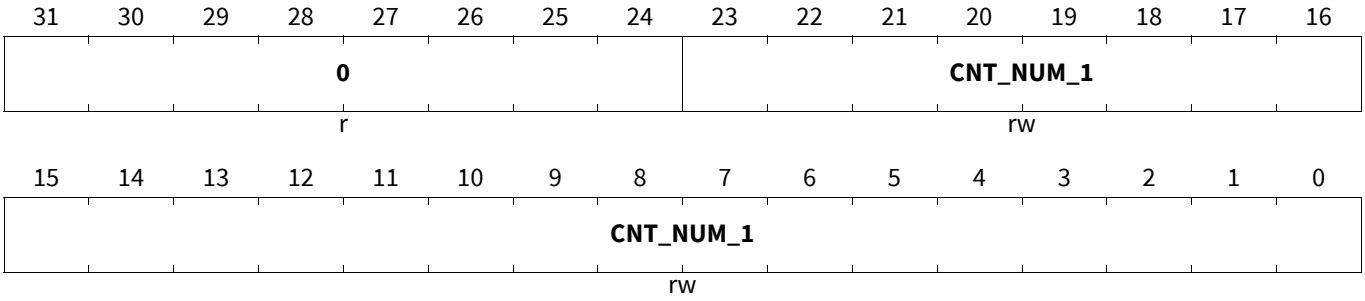
Table 121 Reset Values of DPLL_MLS2

Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	–-00 0000_H	Cleared by state machine

28.20.12.77Memory DPLL_CNT_NUM_1

DPLL Number of Sub-Pulses of SUB_INC1 in Continuous Mode

DPLL_CNT_NUM_1
DPLL Number of Sub-Pulses of SUB_INC1 in Continuous Mode($0285C8_H$) **Reset Value: Table 122**



Field	Bits	Type	Description
CNT_NUM_1	23:0	rw	**Counter for number of SUB_INC1 pulses** Number of pulses in continuous mode for a nominal increment in normal and emergency mode for SUB_INC1, given and updated by CPU only. Count value for continuous mode.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

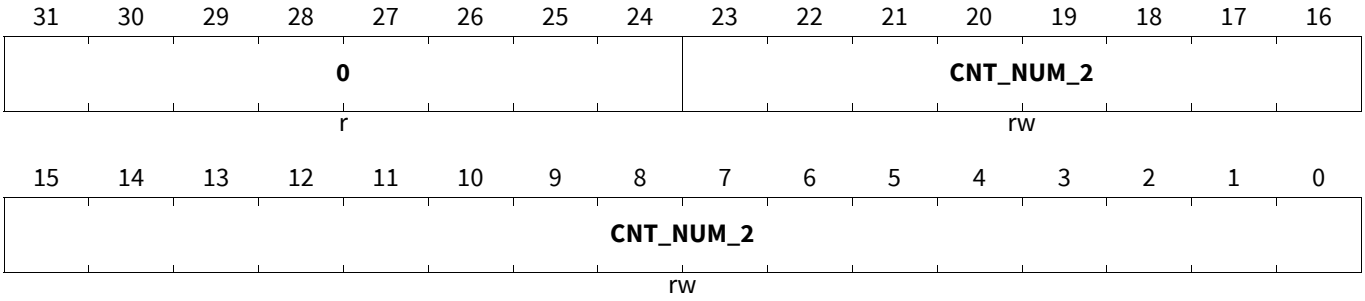
Table 122 Reset Values of DPLL_CNT_NUM_1

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.78Memory DPLL_CNT_NUM_2

DPLL Number of Sub-Pulses of SUB_INC2 in Continuous Mode

DPLL_CNT_NUM_2
DPLL Number of Sub-Pulses of SUB_INC2 in Continuous Mode(0285CC$_H$) **Reset Value: Table 123**



Field	Bits	Type	Description
CNT_NUM_2	23:0	rw	**Counter for number of SUB_INC2 pulses** Number of pulses in continuous mode for a nominal increment in normal and emergency mode for SUB_INC2, given and updated by CPU only. Count value for continuous mode.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 123 Reset Values of DPLL_CNT_NUM_2

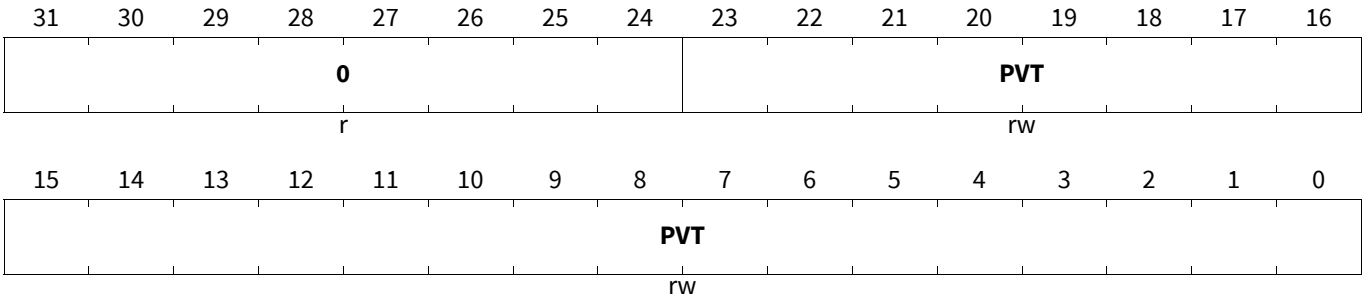
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.79Memory DPLL_PVT

DPLL Plausibility Value of Next TRIGGER Slope

DPLL_PVT

DPLL Plausibility Value of Next TRIGGER Slope (0285D0$_H$) **Reset Value: Table 124**



Field	Bits	Type	Description
PVT	23:0	rw	**Plausibility value of next active TRIGGER slope** The meaning of the value depends on the value of the PIT value in the DPLL_CTRL_1 register. For PIT=0: the number of SUB_INC1 pulses to be waited for until a next active *TRIGGER* event is accepted. For PIT=1: PVT is to be multiplied with the **last nominal increment time DT_T_ACT** and divided by 1024 and reduced to a 24 bit value in order to get the time to be waited for until the next active *TRIGGER* event is accepted. The wait time must be exceeded for an active slope. *Note: When an active TRIGGER slope is detected while the wait condition is not fulfilled the interrupt PWI is generated. Please note, that the SGE1 must be set, when PIT=0 in order to provide the necessary SUB_INC1 pulses for checking. After an unexpected missing TRIGGER the plausibility check is suppressed for the following increment. In case of direction change the PVT value is automatically set to zero in order to deactivate the check.*
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 124 Reset Values of DPLL_PVT

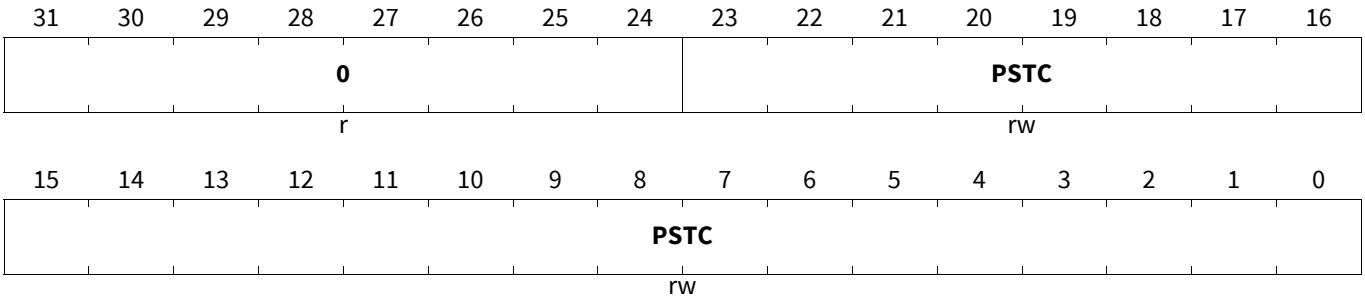
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.80Memory DPLL_PSTC

DPLL Actual Calculated Position Stamp of TRIGGER

DPLL_PSTC

DPLL Actual Calculated Position Stamp of TRIGGER(0285E0$_H$) **Reset Value: Table 125**



Field	Bits	Type	Description
PSTC	23:0	rw	**Calculated position stamp of last TRIGGER input** Value is set by the DPLL and can be updated by the CPU when filter values are to be considered for the exact position (see **DPLL_STATUS** and **DPLL_CTRL** registers for explanation of the status and control bits used). For each active slope of *TRIGGER* in normal mode. When FTD=0: PSTC is set from actual position value, for the first active *TRIGGER* event (no filter delay considered) the CPU must update the value once, taking into account the filter value. When FTD=1: PSTC is incremented at each *TRIGGER* event by SMC=0: (MLT+1)*(SYN_T) +PD; while PD=0 for AMT=0 SMC=1: (MLS1)*(SYN_T) +PD; while PD=0 for AMT=0
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

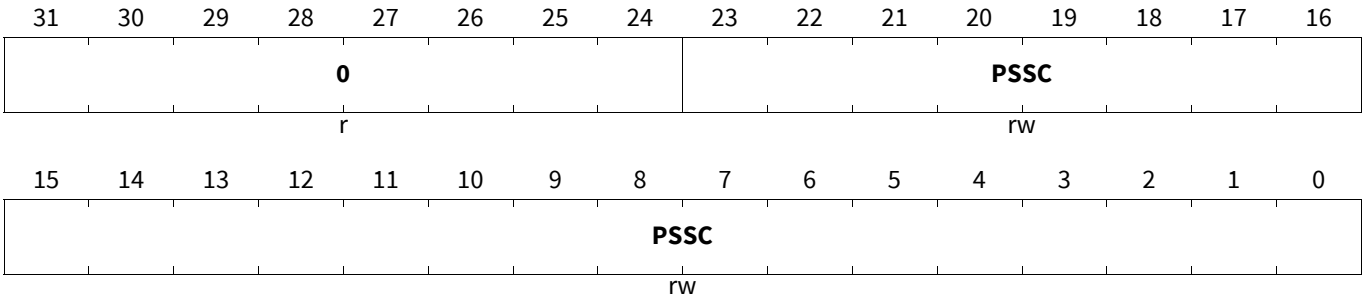
Table 125 Reset Values of DPLL_PSTC

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.81Memory DPLL_PSSC

DPLL Actual Calculated Position Stamp of STATE

DPLL_PSSC
DPLL Actual Calculated Position Stamp of STATE(0285E4$_H$) **Reset Value: Table 126**



Field	Bits	Type	Description
PSSC	23:0	rw	**Calculated position stamp for the last STATE input** First value is set by the DPLL and can be updated by the CPU when the filter delay is to be considered. For each active slope of *STATE* in emergency mode. When FSD=0: PSSC is set from actual position value(no filter delay considered), the CPU must update the value once, taking into account the filter value When FSD=1: at each active slope of *STATE* (PD_S_store=0 for AMS=0): SMC=0: add MLS1*(SYN_S) + PD_S_store; SMC=1: add MLS2*(SYN_S) + PD_S_store;
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 126 Reset Values of DPLL_PSSC

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

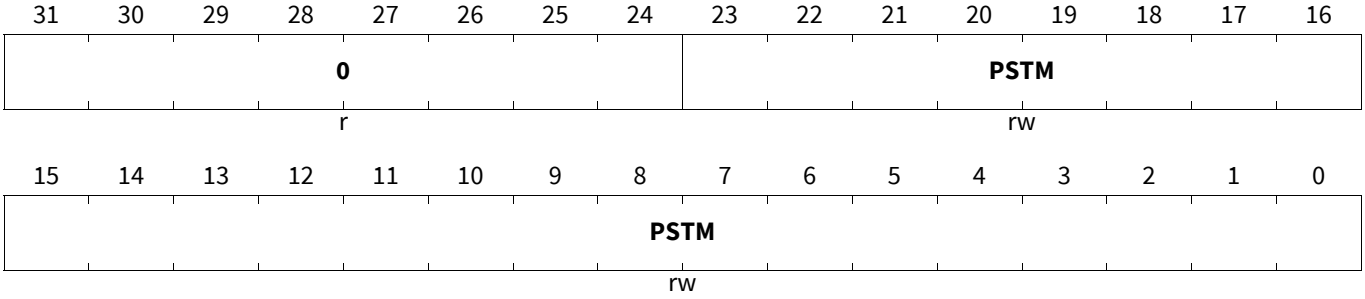
28.20.12.82Memory DPLL_PSTM

DPLL Measured Position Stamp at Last TRIGGER Input

*Note: The LSB address is determined using the SWON_T value in the OSW register (see **Section 28.20.12.8**).*

DPLL_PSTM

DPLL Measured Position Stamp at Last TRIGGER Input(0285E8$_H$) **Reset Value: Table 127**



Field	Bits	Type	Description
PSTM	23:0	rw	**Position stamp of TRIGGER, measured** Measured position stamp of last active *TRIGGER* input. Store the value TBU_TS1 when an active TRIGGER event occurs. The value of PSTM is invalid for (RMO=1 and SMC=0).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 127 Reset Values of DPLL_PSTM

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

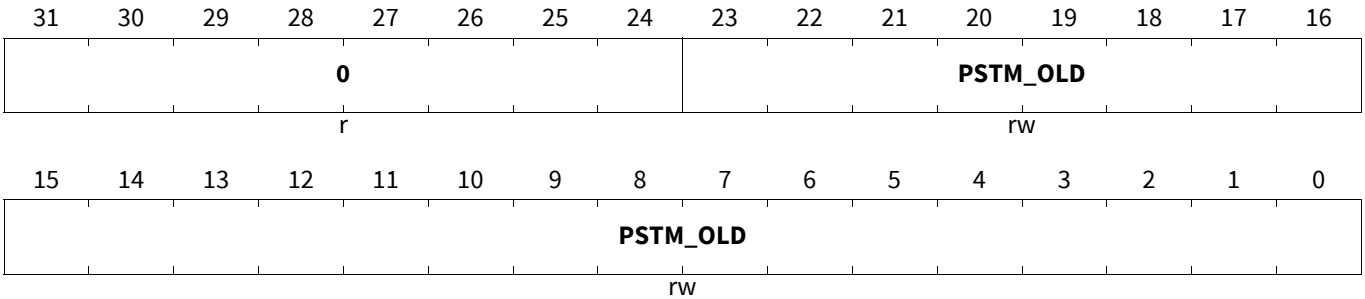
28.20.12.83Memory DPLL_PSTM_OLD

DPLL Measured Position Stamp at Last but One TRIGGER Input

*Note: The LSB address is determined using the SWON_T value in the OSW register (see **Section 28.20.12.8**).*

DPLL_PSTM_OLD

DPLL Measured Position Stamp at Last but One TRIGGER Input(0285EC$_H$) **Reset Value: Table 128**



Field	Bits	Type	Description
PSTM_OLD	23:0	rw	**Last but one position stamp of TRIGGER, measured** Measured position stamp of last but one active *TRIGGER* input. Last PSTM value: see explanation of PSTM.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 128 Reset Values of DPLL_PSTM_OLD

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

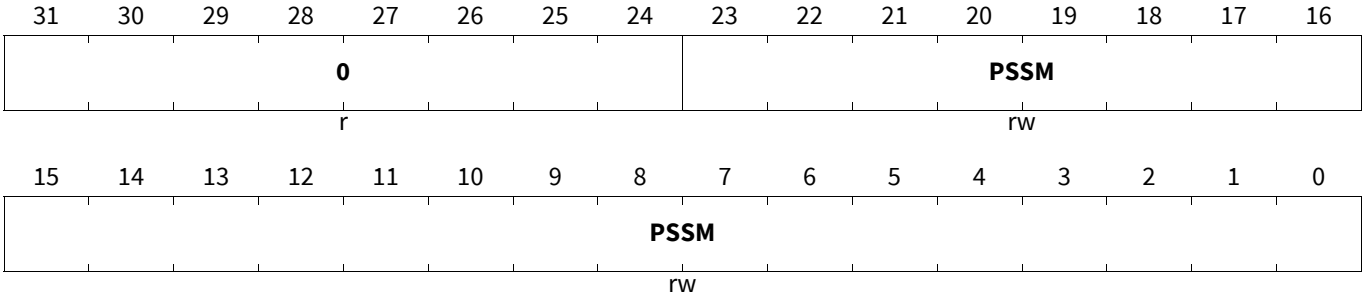
28.20.12.84Memory DPLL_PSSM

DPLL Measured Position Stamp at Last STATE Input

*Note: The LSB address is determined using the SWON_S value in the OSW register (see **Section 28.20.12.8**).*

DPLL_PSSM

DPLL Measured Position Stamp at Last STATE Input(0285F0$_H$) **Reset Value: Table 129**



Field	Bits	Type	Description
PSSM	23:0	rw	**Position stamp of STATE, measured** Measured position stamp of last active STATE input. Store the value TBU_TS1 or TBU_TS2, respectively, at the moment when an active STATE event occurs. The value of PSSM is invalid for (RMO=0 and SMC=0).
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 129 Reset Values of DPLL_PSSM

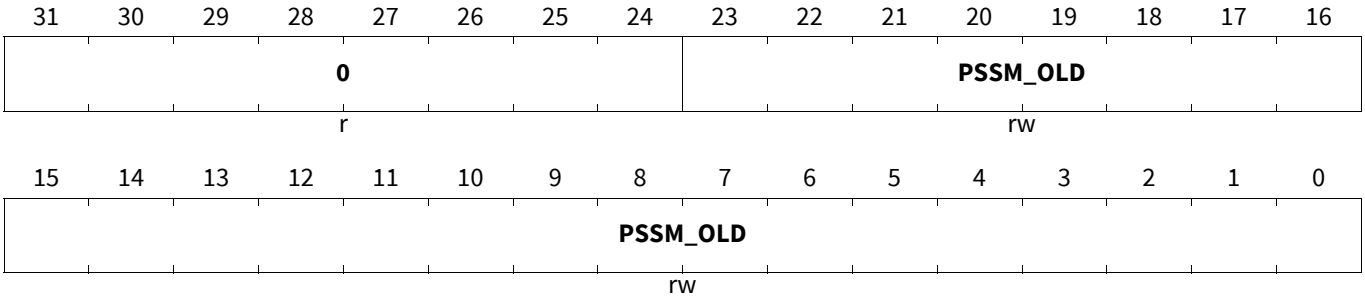
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.85Memory DPLL_PSSM_OLD

DPLL Measured Position Stamp at Last but One STATE Input

*Note: The LSB address is determined using the SWON_S value in the OSW register (see **Section 28.20.12.8**).*

DPLL_PSSM_OLD
DPLL Measured Position Stamp at Last but One STATE Input(0285F4$_H$) **Reset Value: Table 130**



Field	Bits	Type	Description
PSSM_OLD	23:0	rw	**Last but one position stamp of STATE, measured** Measured position stamp of last but one active STATE input. Last PSSM value: see explanation of PSSM.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 130 Reset Values of DPLL_PSSM_OLD

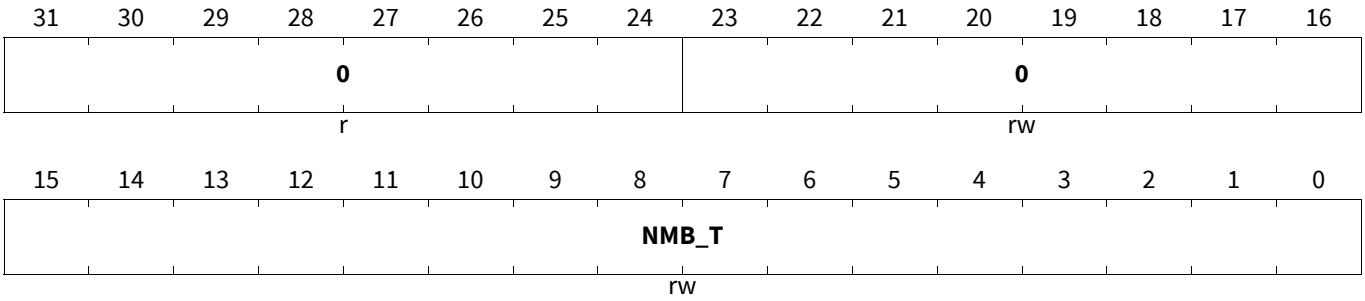
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.86Memory DPLL_NMB_T

DPLL Number of Pulses to be Sent in Normal Mode

DPLL_NMB_T

DPLL Number of Pulses to be Sent in Normal Mode(0285F8$_H$) **Reset Value: Table 131**



Field	Bits	Type	Description
NMB_T	15:0	rw	**Number of pulses for TRIGGER** Calculated number of pulses in normal mode for the current TRIGGER increment. Calculated pulse number.
0	23:16	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 131 Reset Values of DPLL_NMB_T

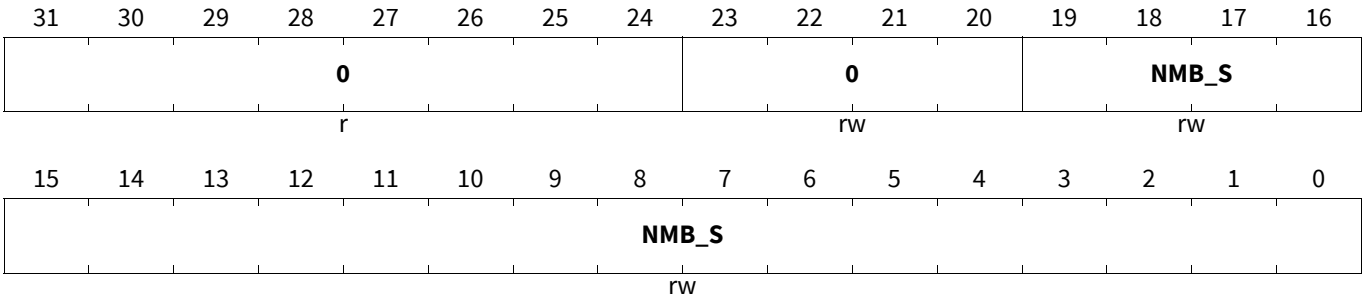
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.87Memory DPLL_NMB_S

DPLL Number of Pulses to be Sent in Emergency Mode

DPLL_NMB_S

DPLL Number of Pulses to be Sent in Emergency Mode(0285FC$_H$) **Reset Value: Table 132**



Field	Bits	Type	Description
NMB_S	19:0	rw	**Number of pulses for STATE** Calculated number of pulses in emergency mode for the current STATE increment. Calculated pulse number.
0	23:20	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 132 Reset Values of DPLL_NMB_S

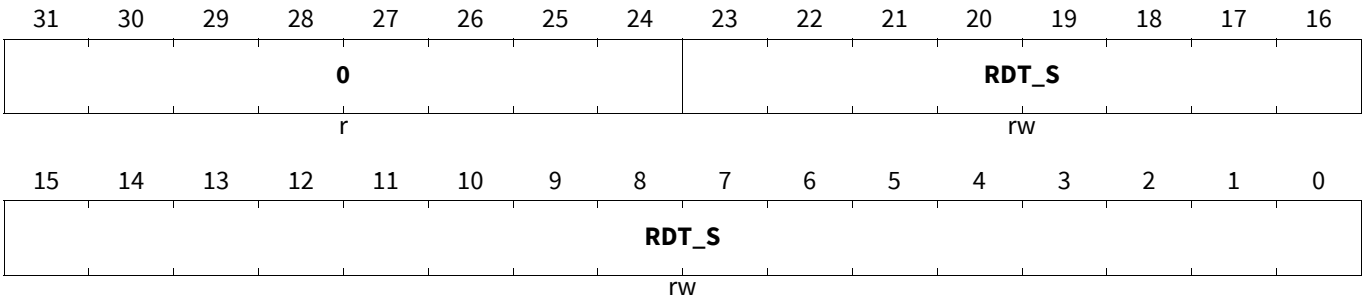
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.88Memory DPLL_RDT_S[i]

DPLL Reciprocal Values of the Nominal STATE i Increment Duration in FULL_SCALE

Note: If DPLL_CTRL_11.STATE_EXT is set, this memory range is not used by the DPLL, but emulated outside the DPLL. The DPLL will access the MCS to DPLL interface 18.15 and will expect data to be correctly stored there. This means in fact, that the handling of the RDT_S values has to be done outside, in the MCS integrated in the same cluster.

DPLL_RDT_Si (i=0-63)
DPLL Reciprocal Values of the Nominal STATE i Increment Duration in FULL_SCALE(028600$_H$+i*4) Reset Value: Table 133



Field	Bits	Type	Description
RDT_S	23:0	rw	**Reciprocal difference time of STATE** Nominal reciprocal value of the number of time stamp clocks measured in the corresponding increment *2^{32} while only the lower 24 bits are used; no gap considered. The LSB is rounded up when the next truncated bit is 1. There are 2*(SNU+1-SYN_NS) entries for SYSF=0 or 2*(SNU+1)-SYN_NS entries for SYSF=1 respectively.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 133 Reset Values of DPLL_RDT_Si (i=0-63)

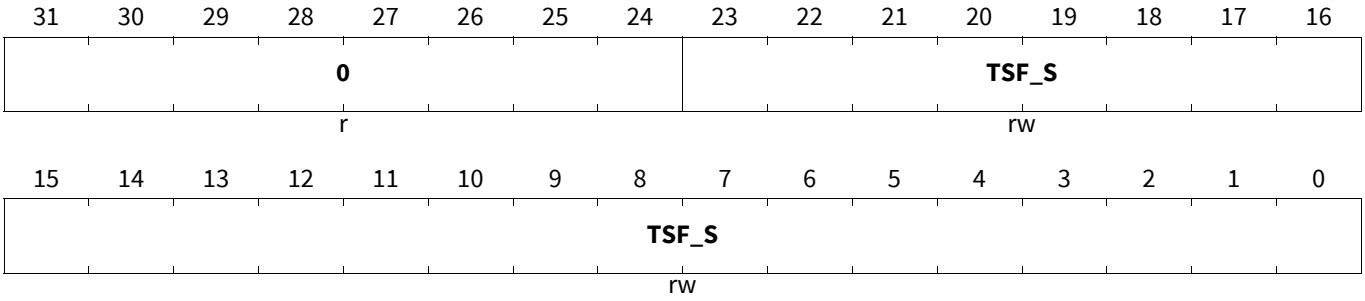
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.89Memory DPLL_TSF_S[i]

DPLL Time Stamp Values of the Nominal STATE i Events in FULL_SCALE

Note: If DPLL_CTRL_11.STATE_EXT is set, this memory range is not used by the DPLL, but emulated outside the DPLL. The DPLL will access the MCS to DPLL interface 18.15 and will expect data to be correctly stored there. This means in fact, that the handling of the TSF_S values has to be done outside, in the MCS integrated in the same cluster.

DPLL_TSF_Si (i=0-63)
DPLL Time Stamp Values of the Nominal STATE i Events in FULL_SCALE(028700$_H$+i*4) Reset Value: Table 134



Field	Bits	Type	Description
TSF_S	23:0	rw	**Time stamp field of STATE** Time stamp value of each active STATE event. There are 2* (SNU+1) entries.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 134 Reset Values of DPLL_TSF_Si (i=0-63)

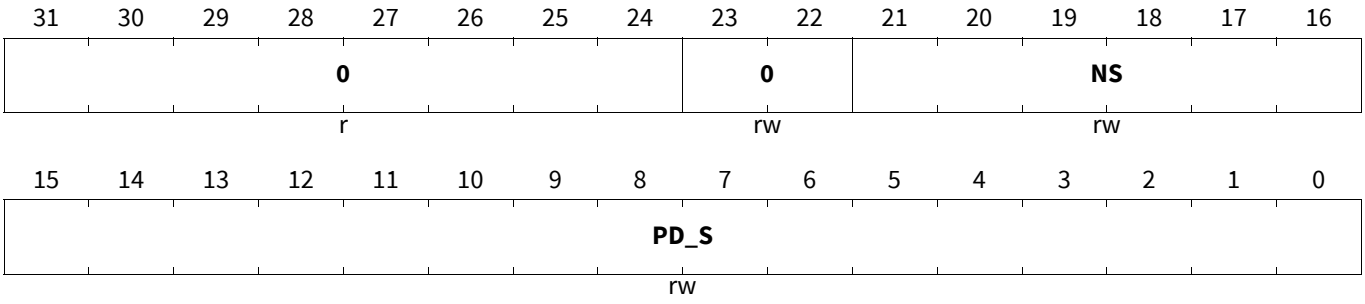
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.90Memory DPLL_ADT_S[i]

DPLL Adapt and Profile Values of the STATE i Increments in FULL_SCALE

Note: If DPLL_CTRL_11.STATE_EXT is set, this memory range is not used by the DPLL, but emulated outside the DPLL. The DPLL will access the MCS to DPLL interface 18.15 and will expect data to be correctly stored there. This means in fact, that the handling of the ADT_S values has to be done outside, in the MCS integrated in the same cluster.

DPLL_ADT_Si (i=0-63)
DPLL Adapt and Profile Values of the STATE i Increments in FULL_SCALE(028800$_H$+i*4) Reset Value: Table 135



Field	Bits	Type	Description
PD_S	15:0	rw	**Physical deviation of STATE** Adapt values for each *STATE* increment in FULL_SCALE (sint16); This value represents the number of pulses to be added to the correspondent nominal increment. The absolute value of a negative PD_S must not exceed MLS1 or MLS2 respectively. The PD value does mean the number of SUB_INC1 pulses per nominal tooth to be added to NS*((MLS1/2+1) + PD_S);
NS	21:16	rw	**Number of STATEs** Number of nominal STATE parts in the corresponding increment. There are 2*(SNU+1-SYN_NS) entries for SYSF=0 or 2*(SNU+1)-SYN_NS entries for SYSF=1 respectively.
0	23:22	rw	**Not used** Must be written to zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 135 Reset Values of DPLL_ADT_Si (i=0-63)

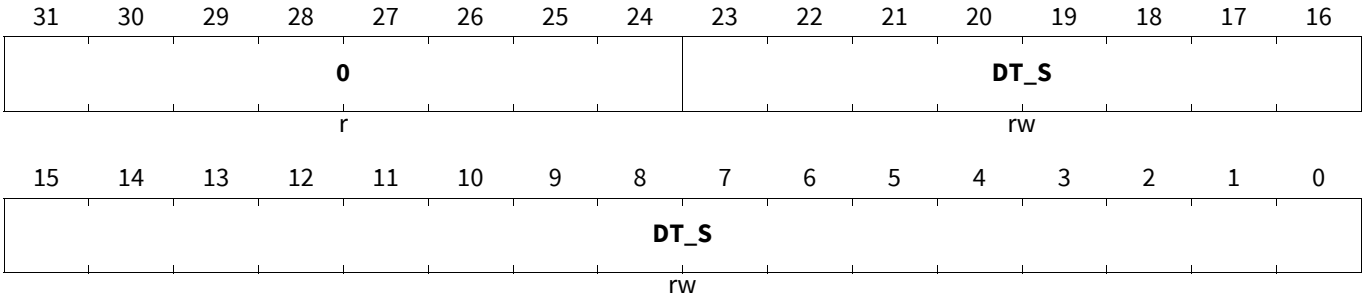
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.91Memory DPLL_DT_S[i]

DPLL Nominal STATE i Increment Duration in FULL_SCALE

Note: If DPLL_CTRL_11.STATE_EXT is set, this memory range is not used by the DPLL, but emulated outside the DPLL. The DPLL will access the MCS to DPLL interface chapter "MCS to DPLL Register description" and will expect data to be correctly stored there. This means in fact, that the handling of the DT_S values has to be done outside, in the MCS integrated in the same cluster.

DPLL_DT_Si (i=0-63)
DPLL Nominal STATE i Increment Duration in FULL_SCALE(028900$_H$+i*4) **Reset Value: Table 136**



Field	Bits	Type	Description
DT_S	23:0	rw	**Difference time of STATE** Nominal increment duration values for each *STATE* increment in FULL_SCALE (considering no gap). There are 2*(SNU+1-SYN_NS) entries for SYSF=0 or 2*(SNU+1)-SYN_NS entries for SYSF=1 respectively.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 136 Reset Values of DPLL_DT_Si (i=0-63)

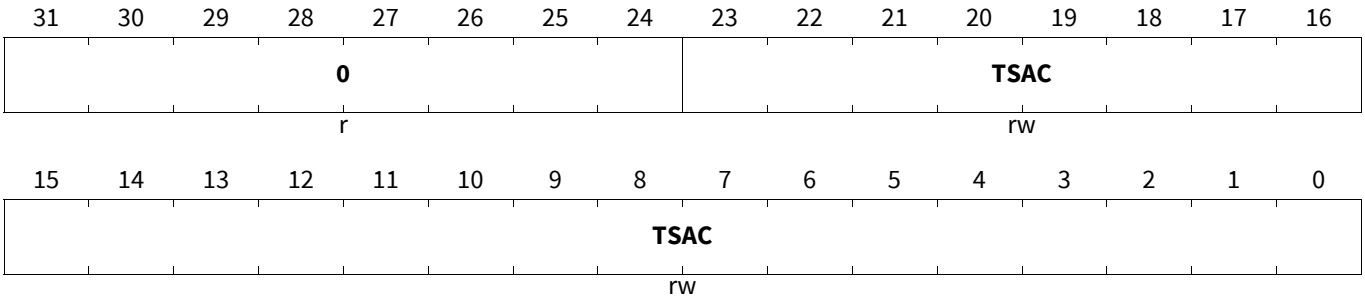
Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1BC	--00 0000$_H$	Cleared by state machine

28.20.12.92Register DPLL_TSAC[z]

DPLL Calculated Time Value to start Action z Register

DPLL_TSACz (z=0-31)

DPLL Calculated Time Value to start Action z Register(028E00$_H$+z*4) Application Reset Value: 007F FFFF$_H$



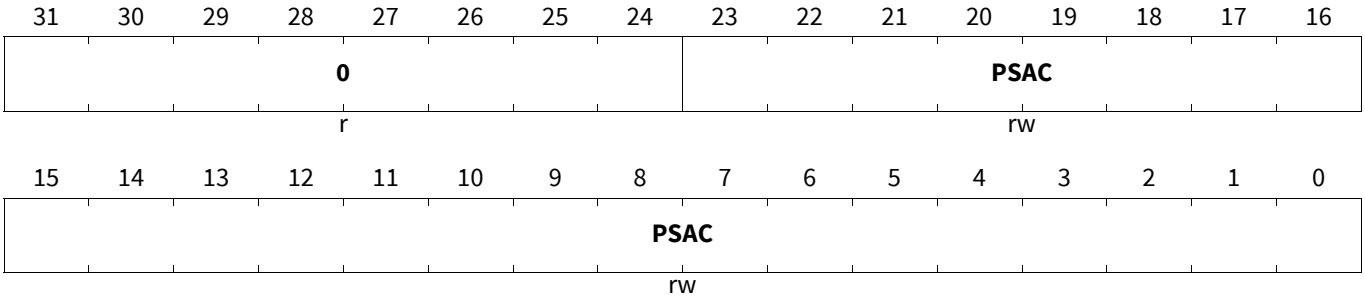
Field	Bits	Type	Description
TSAC	23:0	rw	**Calculated time stamp for ACTION_z** This value can only be written when the DPLL is disabled.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.93Register DPLL_PSAC[z]

DPLL ACTION Position/Value Action z Request Register

DPLL_PSACz (z=0-31)

DPLL ACTION Position/Value Action z Request Register(028E80$_H$+z*4) Application Reset Value: 007F FFFF$_H$



Field	Bits	Type	Description
PSAC	23:0	rw	**Calculated position value for the start of ACTION_z in normal or emergency mode according to equations DPLL-17 or DPLL-20, respectively** This value can only be written when the DPLL is disabled.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.94Register DPLL_ACB_[z]

DPLL Control Bits Register z for up to 32 Actions

DPLL_ACB_z (z=0-7)

DPLL Control Bits Register z for up to 32 Actions(028F00$_H$+z*4) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**ACB_3**					**0**			**ACB_2**				
r			rw					r			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**ACB_1**					**0**			**ACB_0**				
r			rw					r			rw				

Field	Bits	Type	Description
ACB_0	4:0	rw	**Action Control Bits of ACTION_z** Reflects ACT_D[i](52:48), i=4*z. When DPLL_CTRL_11.ACBU = '0': ACB_0[4:0] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation. When DPLL_CTRL_11.ACBU = '1': ACB_0[4:2] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation. ACB_0[1]= '1' is used as input signal to control if "action in past" shall be checked based on position information. ACB_0[1] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far. ACB_0[0] is used as input signal to control if "action in past" shall be checked based on time information. ACB_0[0] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far. This value can only be written via AEI-interface when the DPLL is disabled.

Field	Bits	Type	Description
ACB_1	12:8	rw	**Action Control Bits of ACTION_(i + 1)** Reflects ACT_D[i+1](52:48), i=4*z. When DPLL_CTRL_11.ACBU = '0': ACB_1[4:0] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation. When DPLL_CTRL_11.ACBU = '1': ACB_1[4:2] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation.ACB_1[1]= '1' is used as input signal to control if "action in past" shall be checked based on position information. ACB_1[1] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.ACB_1[0] is used as input signal to control if "action in past" shall be checked based on time information. ACB_1[0] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far.This value can only be written via AEI-interface when the DPLL is disabled.
ACB_2	20:16	rw	**Action Control Bits of ACTION_(i + 2)** Reflects ACT_D[i+2](52:48), i=4*z. When DPLL_CTRL_11.ACBU = '0': ACB_2[4:0] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation. When DPLL_CTRL_11.ACBU = '1': ACB_2[4:2] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation. ACB_2[1]= '1' is used as input signal to control if "action in past" shall be checked based on position information. ACB_2[1] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far. ACB_2[0] is used as input signal to control if "action in past" shall be checked based on time information. ACB_2[0] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far. This value can only be written via AEI-interface when the DPLL is disabled.

Field	Bits	Type	Description
ACB_3	28:24	rw	**Action Control Bits of ACTION_(i + 3)** Reflects ACT_D[i+3](52:48), i=4*z. When DPLL_CTRL_11.ACBU = '0': ACB_3[4:0] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation. When DPLL_CTRL_11.ACBU = '1': ACB_3[4:2] are taken as received by ARU interface and are transmitted unchanged as result of action (PMT) calculation. ACB_3[1]= '1' is used as input signal to control if "action in past" shall be checked based on position information. ACB_3[1] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far. ACB_3[0] is used as input signal to control if "action in past" shall be checked based on time information. ACB_3[0] is written to '1' if action channel has reached "action in past" condition after action has been calculated, written to '0' if action has not reached "past" so far. This value can only be written via AEI-interface when the DPLL is disabled.
0	7:5, 15:13, 23:21, 31:29	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.95Register DPLL_CTRL_11

DPLL Control Register 11

DPLL_CTRL_11
DPLL Control Register 11 **(028F20$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WACBU	**WSTATE_EXT**	**WPCMF2_INCCNT_B**	**WINCF2**	**WFSYL2**	**WPCMF2**	**WERZ2**	**WSIP2**	**WADS**	**WADT**	**WPCMF1_INCCNT_B**	**WINCF1**	**WFSYL1**	**WPCMF1**	**WERZ1**	**WSIP1**
rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACBU	**STATE_EXT**	**PCMF2_INCCNT_B**	**INCF2**	**FSYL2**	**PCMF2**	**ERZ2**	**SIP2**	**ADS**	**ADT**	**PCMF1_INCCNT_B**	**INCF1**	**FSYL1**	**PCMF1**	**ERZ1**	**SIP1**
rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SIP1	0	rw	**Simplified increment prediction in normal mode and for the first engine in the case SMC=1** For the first increment after setting SIP1 from 0 to 1, the value of DT_T_ACT is replaced by the value of the DT_T_START register. This results in a CDT_TX value which is equal to DT_T_START. Please notice that this DT_T_Start value must be always > 256. *Note: The value of SIP1 influences only the increment prediction CDT_TX and when NUTE-VTN=1. The calculation of QDT_T itself is not influenced by the SIP1 bit. The value of SIP1 can be only be written when WSIP1=1.* *Note: When SIP1=1 is set, the first pulses of the subincrement generator are not generated with highest frequency for the first increment (DPLL_STATUS.FTD = 0, DPLL_CTRL_1.SGE1=1).* 0_B Increment prediction calculation; the current increment duration is calculated using the relation between increment durations in the past, like explained by the corresponding equations 1_B Increment prediction continuation; in this mode, for the increment prediction value calculation of CDT_TX the value of QDT_T is replaced by 1 for all calculations when NUTE-VTN=1; in the other case, the value of SIP1 is ignored, and the calculation is performed like for SIP1=0
ERZ1	1	rw	**Error is assumed as zero in normal mode and for the first engine for SMC=1** The calculation of EDT_T and MEDT_T is performed independently from the ERZ1 value in all modes without any influence to the MEDT_T value itself. The ERZ1 value influences the use of MEDT_T in normal mode and for SMC=1. The value of ERZ1 can be only written when WERZ1=1. 0_B The MEDT_T and MEDT_S values are considered as provided in the corresponding equations 1_B Instead of using MEDT_T, the value '0' is used in the corresponding equations

Field	Bits	Type	Description
PCMF1	2	rw	**Pulse correction mode fast for INC_CNT1** The fast pulse generation is performed immediately within the current increment. MPVAL1 must be positive integers for the fast pulse correction mode - in the case of negative values, the correction is suppressed, and the FPCE (fast pulse correction error) bit in the DPLL_STATUS register is set, causing the EI (error interrupt) when enabled. The setting of PCMF1 prevents the transfer of control bits PCM1 to the corresponding shadow registers with an active input event, and prevents therefore the distribution of the MPVAL1 values over the current or next increment. The MPVAL1 pulses are sent with the fast clock CMU_CLK0 by the rapid pulse generator RPCUx (see chapter 18.8.3.6 of specification v3.0) triggered in the state 6/26 or 18/38 of the state machines (see **Section 28.20.8.6.1**), respectively. The INC_CNT1 is incremented by MPVAL1, respectively. When taken the MPVAL1 value to RPCUx and INC_CNT1, the PCM1 bit is reset immediately, and after that, also the PCMF1 bit. The value of PCMF1 can be only written when WPCMF1=1. Be careful when using the fast pulse correction during a direction change. Because of sending the correction pulses before, during or after the direction change recognition, the result is typically unpredictable. No automatic correction of the fast correction pulses is provided. The necessary corrections must be performed on responsibility of the user. 0_B No fast update of pulses, provided by MPVAL1 1_B When PCM1 is set while PCMF1=1, the pulses provided by MPVAL1 are sent using the rapid pulse generator RPCUx, without waiting for a new input event
FSYL1	3	rw	**Force Synchronization Loss of LOCK1** The synchronization loss resets SYT/SYS and prevents the use of profiles, respectively. The above described effect for FSYL1=1 is only active when WFSYL1=1 simultaneously. 0_B No force of synchronization loss 1_B Reset LOCK1 and reset SYT in normal mode, and for SMC=1, reset SYS in emergency mode

Field	Bits	Type	Description
INCF1	4	rw	**INC_CNT1 fast correction** The calculation of ADD_IN for the SUB_INC generation is performed without adding the 0.5 value to NMB_T/S in equations DPLL_25 ff. The signal RESET_SIGx of the pulse generator is activated for each new active input slope, in order to reset the register values. *Note: The INCF1 value can be only written when WINCF1=1.* *Note: The INCF1 bit should only be written when DPLL_CTRL_1.DEN = '0' (DPLL disabled) to prevent generation of wrong number of sub increments.* 0_B The calculation of a new INC_CNT1 is performed after an active slope was detected and the plausibility check was performed 1_B The calculation of a new INC_CNT1 is prepared before an active slope is detected; the plausibility check is supported by an additional HW checker in order to get the decision earlier, and after this decision, the pulse generator for SUB_INC1 starts immediately sending out pulses
PCMF1_INCCNT_B	5	rw	**No increment of INC_CNT1 when PCMF1 active (automatic end mode)** The PCMF1_INCCNT_B value can be only written when WPCMF1_INCCNT_B =1. 0_B When fast pulse correction is done by PCM1, PCMF1, the MPVAL1 value is as well added to the INC_CNT1 1_B Do not add MPVAL1 value to the INC_CNT1 register when fast pulse correction is done by PCM1 or PCMF1. This means that just fast pulses are done by decrementing current content of INC_CNT1 register as long as INC_CNT1 is not zero (automatic end mode). The number of pulses (MPVAL1) shall be sufficiently smaller than INC_CNT1 when MPVAL1 is written.
ADT	6	r	**Correction of DT_T_ACTUAL,CDT_TX_nom_corr by PD_T** 0_B No correction of DT_T_ACTUAL, CDT_TX_nom_corr by physical deviation (PD_T) defined in profile of TRIGGER processing unit 1_B Correction of DT_T_ACTUAL, CDT_TX_nom_corr by physical deviation (PD_T) defined in profile of TRIGGER processing unit
ADS	7	r	**Correction of DT_S_ACTUAL, CDT_SX_nom_corr by PD_S** 0_B No correction of DT_S_ACTUAL, CDT_SX_nom_corr by physical deviation (PD_S) defined in profile of STATE processing unit 1_B Correction of DT_S_ACTUAL, CDT_SX_nom_corr by physical deviation (PD_S) defined in profile of STATE processing unit

Field	Bits	Type	Description
SIP2	8	rw	**Simplified increment prediction in emergency mode and for the second engine in the case RMO=1** For the first increment after setting SIP2 from 0 to 1, the value of DT_S_ACT is replaced by the value of the DT_S_START register. This results in a CDT_SX value which is equal to DT_S_START. Please notice that this DT_S_START value must be always > 256. The value of SIP2 influences only the increment prediction and error accumulation when NUSE-VSN=1. The calculation of QDT_S itself is not influenced by the SIP2 bit. The value of SIP2 can be only written when WSIP2=1. 0_B Increment prediction calculation; the current increment duration CDT_SX is calculated using the relation between increments duration in the past, like explained by the corresponding equations 1_B Increment prediction value calculation CDT_SX; the value of QDT_S is replaced by 1 for all calculations when NUSE-VSN=1; in the other case, the value of SIP2 is ignored, and the calculation is performed like for SIP2=0
ERZ2	9	rw	**Error is assumed as zero in emergency mode and for the second engine for SMC=1** The calculation of EDT_S and MEDT_S is performed independently from the ERZ2 value in all modes, without any influence to the MEDT_S value itself. The ERZ2 value influences the use of MEDT_S in emergency mode and for SMC=1 with RMO=1. The value of ERZ2 can be only written when WERZ2=1. 0_B The MEDT_S value is considered as provided in the corresponding equations 1_B Instead of using MEDT_S, the value '0' is used in the corresponding equations

Field	Bits	Type	Description
PCMF2	10	rw	**Pulse correction mode fast for INC_CNT2** The fast pulse generation is performed immediately within the current increment. MPVAL2 must be positive integers for the fast pulse correction mode - in the case of negative values, the correction is suppressed, and the FPCE (fast pulse correction error) bit in the DPLL_STATUS register is set, causing the EI (error interrupt) when enabled. The setting of PCMF2 prevents the transfer of control bits PCM2 to the corresponding shadow registers with an active input event, and therefore prevents the distribution of the MPVAL1 values over the current or next increment. The MPVAL2 pulses are sent with the fast clock CMU_CLK0 by the rapid pulse generator RPCUx (of specification v3.0) triggered in the state 6/26 or 18/38 of the state machines. The INC_CNT2 is incremented by MPVAL2, respectively. When taken the MPVAL2 value to RPCUx and INC_CNT2, the PCM2 bit is reset immediately, and after that also the PCMF2 bit. The value of PCMF2 can be only written when WPCMF2=1. Be careful when using the fast pulse correction during a direction change. Because of sending the correction pulses before, during or after the direction change recognition, the result is typically unpredictable. No automatic correction of the fast correction pulses is provided. The necessary corrections must be performed on responsibility of the user. 0_B No fast update of pulses provided by MPVAL2 1_B When PCM2 is set while PCMF2=1, the pulses provided by MPVAL2 are sent using the rapid pulse generator RPCUx, without waiting for a new input event
FSYL2	11	rw	**Force Synchronization Loss of LOCK2** The synchronization loss resets SYS and prevents the use of profiles respectively. The above described effect for FSYL2=1 is only active when WFSYL2=1 simultaneously. 0_B No force of synchronization loss 1_B Reset LOCK2 and reset SYS in emergency mode and for SMC=1
INCF2	12	rw	**INC_CNT2 fast** The INCF2 value can be only written when WINCF2=1. 0_B The calculation of a new INC_CNT2 is performed after an active slope was detected and the plausibility check was performed 1_B The calculation of a new INC_CNT2 is prepared before an active slope is detected; the plausibility check is supported by an additional HW checker in order to get the decision earlier, and after this decision, the pulse generator for SUB_INC2 starts immediately sending out pulses. The calculation of ADD_IN for the SUB_INC generation is performed without adding the 0.5 value to NMB_S in equations DPLL_25 ff. The signal RESET_SIGx of the pulse generator is activated for each new active input slope in order to reset the register values.

Field	Bits	Type	Description
PCMF2_INCCNT_B	13	rw	**No increment of INC_CNT2 when PCMF2 active (automatic end mode)** The PCMF2_INCCNT_B value can be only written when WPCMF2_INCCNT_B=1. 0_B Add MPVAL2 value is as well added to the INC_CNT2 when fast pulse correction is done by PCM2 or PCMF2 1_B Do not add MPVAL2 value to the INC_CNT2 register when fast pulse correction is done by PCM2 or PCMF2. This means, that just fast pulses are done by decrementing current content of INC_CNT2 register as long as INC_CNT2 is not zero (automatic end mode). The number of pulses (MPVAL2) shall be sufficiently smaller than INC_CNT2 when MPVAL2 is written.
STATE_EXT	14	rw	**Use of STATE engine extension** The STATE_EXT value can be only written when WSTATE_EXT=1 and the DPLL is disabled. See 18.10 for a further explanation. If this bit shall be modified during operation, a software reset of the DPLL module is strongly recommended. A RAM initialisation should also be considered depending on the given application case. 0_B STATE extension is not considered 1_B STATE extension is enabled for up to 128 STATE events
ACBU	15	rw	**ACB use; the ACB values of PMTR are used to decide if an action is in the past** Return ACB values together with actions as zero, when the actions are in the future; Set ACB[1]=1, when calculated position value is in the past and the ACB[1] of PMTR was 1. Set ACB[0]=1, when calculated time value is in the past and the ACB[0] of PMTR was 1. The value of ACBU can be only written when WACBU=1. 0_B ACB values of PMTR are not considered in DPLL; the decision, whether an action is in the past, is made considering the calculated time value 1_B ACB values of PMTR are considered in DPLL as follows: If ACB[1] = 1, consider whether the calculated position value of the corresponding action is in the past if ACB[0] = 1; consider whether the calculated time value of the corresponding action is in the past; ACB[1] and ACB[0] can be set also simultaneously to 1
WSIP1	16	rw	**Write enable for simplified increment prediction 1** Enable writing. 0_B Writing to SIP1 is not enabled 1_B Writing to SIP1 is enabled
WERZ1	17	rw	**Write enable for error zero 1** Enable writing. 0_B Writing to ERZ1 is not enabled 1_B Writing to ERZ1 is enabled

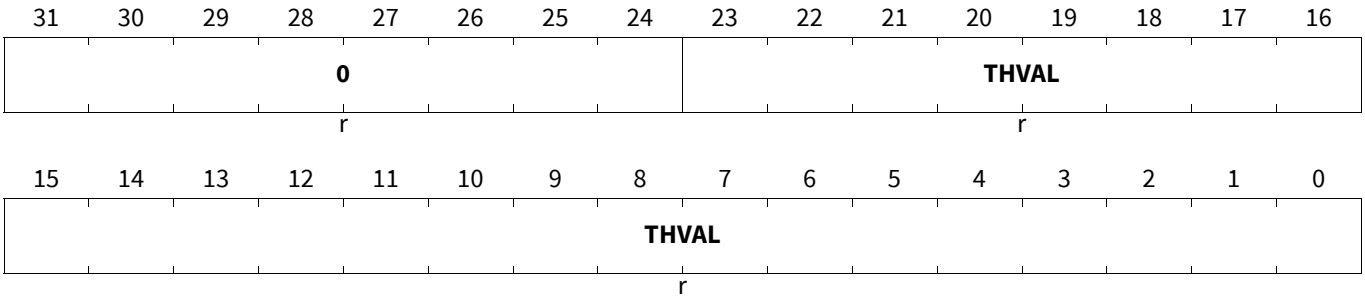
Field	Bits	Type	Description
WPCMF1	18	rw	**Write enable for pulse correction mode fast 1** Enable writing. 0_B Writing to PCM1 is not enabled 1_B Writing to PCM1 is enabled
WFSYL1	19	rw	**Write enable for force synchronization loss 1** Enable writing. 0_B Writing to FSYL1 is not enabled 1_B Writing to FSYL1 is enabled
WINCF1	20	rw	**Write enable for INC_CNT1 fast** Enable writing. 0_B Writing to INCF1 is not enabled 1_B Writing to INCF1 is enabled
WPCMF1_INCCNT_B	21	rw	**Write enable of PCMF1_INCCNT_B** Enable writing. 0_B Writing to PCMF1_INCCNT_B is not enabled 1_B Writing to PCMF1_INCCNT_B is enabled
WADT	22	r	**Write enable of ADT** Enable writing. 0_B Writing to ADT is not enabled 1_B Writing to ADT is enabled
WADS	23	r	**Write enable of ADS** Enable writing. 0_B Writing to ADS is not enabled 1_B Writing to ADS is enabled
WSIP2	24	rw	**Write enable for simplified increment prediction 2** Enable writing. 0_B Writing to SIP2 is not enabled 1_B Writing to SIP2 is enabled
WERZ2	25	rw	**Write enable for error zero 2** Enable writing. 0_B Writing to ERZ2 is not enabled 1_B Writing to ERZ2 is enabled
WPCMF2	26	rw	**Write enable for pulse correction mode fast 2** Enable writing. 0_B Writing to PCMF2 is not enabled 1_B Writing to PCMF2 is enabled
WFSYL2	27	rw	**Write enable for force synchronization loss 2** Enable writing. 0_B Writing to FSYL2 is not enabled 1_B Writing to FSYL2 is enabled
WINCF2	28	rw	**Write enable for INC_CNT2 fast** Enable writing. 0_B Writing to INCF2 is not enabled 1_B Writing to INCF2 is enabled

Field	Bits	Type	Description
WPCMF2_INCCNT_B	29	rw	**Write enable of PCMF2_INCCNT_B** 0_B Writing to PCMF2_INCCNT_B is not enabled 1_B Writing to PCMF2_INCCNT_B is enabled
WSTATE_EXT	30	rw	**Write enable of STATE_EXT** 0_B Writing to STATE_EXT is not enabled 1_B Writing to STATE_EXT is enabled
WACBU	31	rw	**Write enable for ACB use** The ACB values of PMTR are used to decide whether an action is in the past. Enable writing. 0_B Writing to ACBU is not enabled 1_B Writing to ACBU is enabled

28.20.12.96Register DPLL_THVAL2

DPLL Immediate THVAL Value Register

DPLL_THVAL2
DPLL Immediate THVAL Value Register (028F24$_H$) Application Reset Value: 0000 0000$_H$



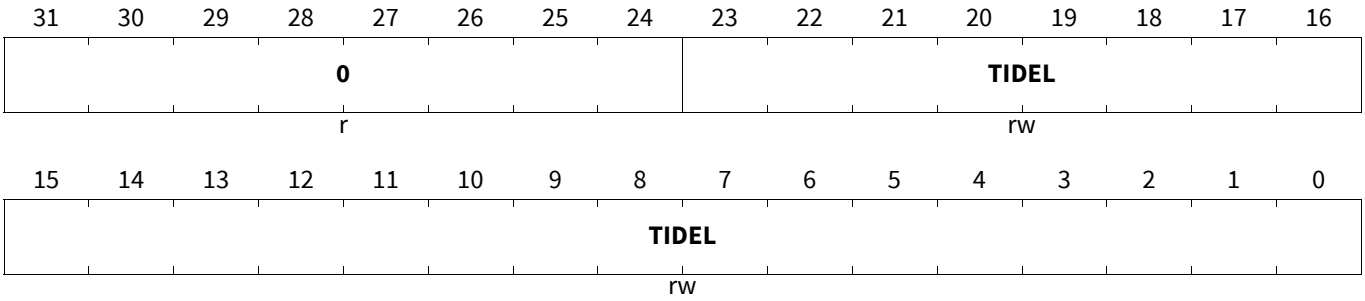
Field	Bits	Type	Description
THVAL	23:0	r	**Measured last pulse time from active to inactive slope of TRIGGER after correction of input slope filter delays** This value is available immediately after the inactive slope of TRIGGER. The measured value considers all input slope filter delays. From the received input the corresponding filter delays are subtracted before the time stamp difference of active and inactive slope is calculated.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.97Register DPLL_TIDEL

DPLL Additional TRIGGER Input Delay Register

DPLL_TIDEL

DPLL Additional TRIGGER Input Delay Register (028F28$_H$) **Application Reset Value: 0000 0000$_H$**



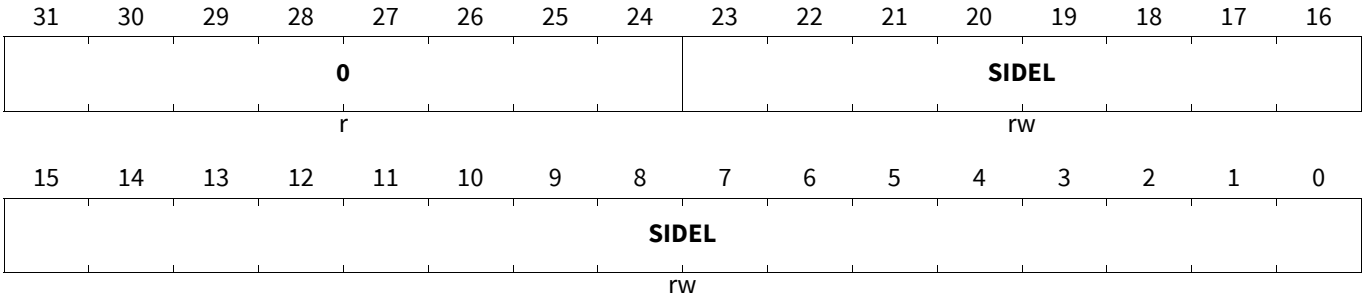
Field	Bits	Type	Description
TIDEL	23:0	rw	**TRIGGER input delay** Transmit this value with each active TRIGGER slope into a shadow register. Subtract this shadow register value from each TRIGGER time stamp (active and inactive slope). This feature is always active and cannot be disabled by a control bit.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.98Register DPLL_SIDEL

DPLL Additional STATE Input Delay Register

DPLL_SIDEL

DPLL Additional STATE Input Delay Register (028F2C$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SIDEL	23:0	rw	**STATE input delay** Transmit this value with each active STATE slope into a shadow register. Subtract this shadow register value from each STATE time stamp (active and inactive slope). This feature is always active and cannot be disabled by a control bit.

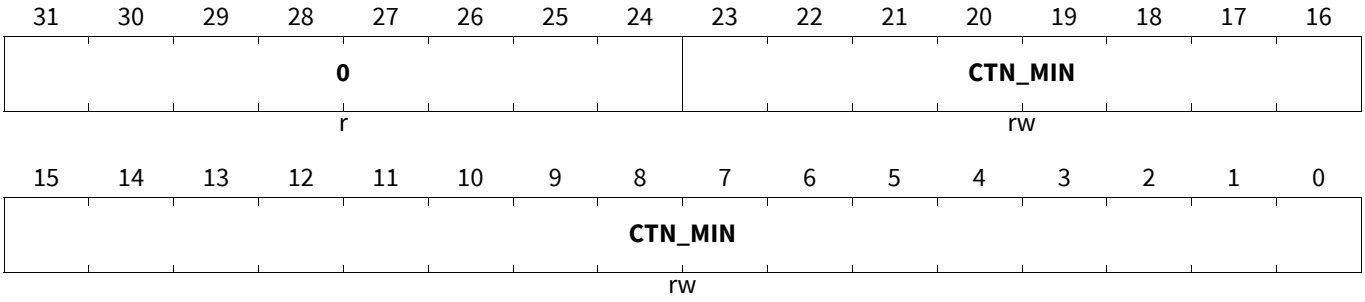
Field	Bits	Type	Description
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.99Register DPLL_CTN_MIN

DPLL Minimum CDT_T Nominal Value Register

DPLL_CTN_MIN

DPLL Minimum CDT_T Nominal Value Register (028F6C$_H$) **Application Reset Value: 0000 0000$_H$**



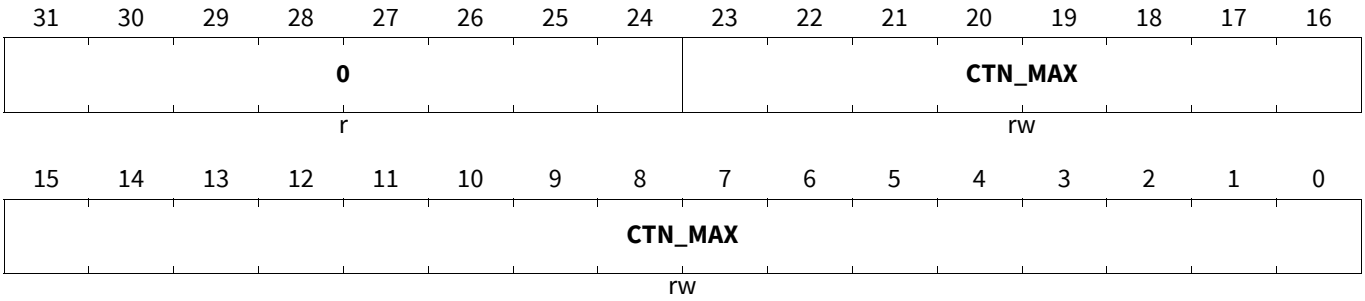
Field	Bits	Type	Description
CTN_MIN	23:0	rw	**CDT_T_NOM min value** Use this register value as CDT_T_NOM value when the calculated value for the nominal increment prediction of TRIGGER is less than the register value.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.100Register DPLL_CTN_MAX

DPLL Maximum CDT_T Nominal Value Register

DPLL_CTN_MAX

DPLL Maximum CDT_T Nominal Value Register (028F70$_H$) **Application Reset Value: 00FF FFFF$_H$**



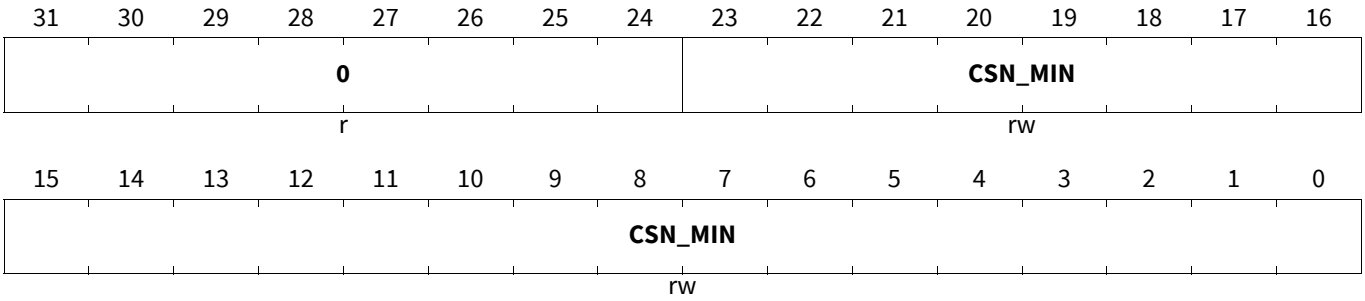
Field	Bits	Type	Description
CTN_MAX	23:0	rw	**CDT_T_NOM max value** Use this register value as CDT_T_NOM value when the calculated value for the nominal increment prediction of TRIGGER is greater than the register value.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.101Register DPLL_CSN_MIN

DPLL Minimum CDT_S Nominal Value Register

DPLL_CSN_MIN

DPLL Minimum CDT_S Nominal Value Register (028F74$_H$) **Application Reset Value: 0000 0000$_H$**



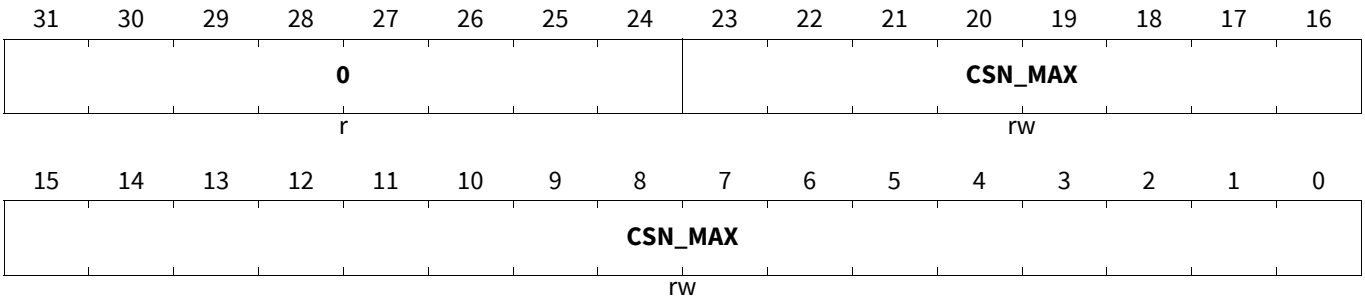
Field	Bits	Type	Description
CSN_MIN	23:0	rw	**CDT_SX_NOM min value** Use this register value as CDT_SX_NOM value when the calculated value for the nominal increment prediction of STATE is less than the register value.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.102Register DPLL_CSN_MAX

DPLL Maximum CDT_S Nominal Value Register

DPLL_CSN_MAX

DPLL Maximum CDT_S Nominal Value Register (028F78$_H$) **Application Reset Value: 00FF FFFF$_H$**



Field	Bits	Type	Description
CSN_MAX	23:0	rw	**CDT_SX_NOM max value** Use this register value as CDT_SX_NOM value when the calculated value for the nominal increment prediction of STATE is greater than the register value.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.103Register DPLL_STA

DPLL Status of the State Machine States Register

DPLL_STA

DPLL Status of the State Machine States Register(028F40$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								CNT_S			0	STA_S			
r								r			r	r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STA_S				CNT_T			0	STA_T							
r				r			r	r							

Field	Bits	Type	Description
STA_T	7:0	r	**Status of TRIGGER state machine; state binary coded** This bit field reflects the status of the TRIGGER state machine. The decimal step number 1 to 20 of the state machine is binary coded from 0x01 to 0x14 respectively using the upper 5 bits (8:4). The lower 4 bits (3:0) show substates of the corresponding state machine. When the DPLL is disabled, this field is 0x000. Table STA_T
CNT_T	11:9	r	**Count TRIGGER** This reflects the count of active TRIGGER slopes (mod8). This value shows the number of active TRIGGER slopes (mod8). This value allows distinguishing if the above state machine status is consistent to other status values read before or after it.

Field	Bits	Type	Description
STA_S	19:12	r	**Status of STATE state machine** State binary coded. This bit field reflects the status of the STATE state machine. The decimal step number 21 to 40 of the state machine is binary coded from 0x01 to 0x14 respectively using the upper 5 bits (20:16) after subtraction of 20 to the decimal value. The lower 4 bits (15:12) show substates of the corresponding state machine. When the DPLL is disabled, this field is 0x000. Table_STA_S
CNT_S	23:21	r	**Count STATE** This reflects the count of active STATE slopes (mod8). This value shows the number of active STATE slopes (mod8). This value allows distinguishing if the above state machine status is consistent to other status values read before or after it.
0	8, 20, 31:24	r	**Reserved** Read as zero, shall be written as zero.

Bit description of DPLL_STA

0	1	Wait, DEN=0
0	2	Calculation of 1/mlt+1, mls1, mls2.
0	3	Calculation of direction change issues (pointers and profile update)
0	4	--
0	5	--
0	6	APS_1c3 was incremented
0	7	APS_1c3 was incremented. Update of pointers is finished, perform change of direction operations
1	0	pvt-check
1	1	update of RAM: write RDT_S; DT_S; TSF_S
1	2	loading of profile (syn_s, update syn_s_old) from ADT_S
1	3	SASI-irq, store FTV into RAM1b
1	4	Write PSSC; modify aps, aps_1c2; aps_1c3 (if synchronized); Start fast pulse updates if necessary; Update inc_cnt1/2;
2	0	Write TS_S to ram1b, calculate dt_s_actual
3	0	update cdsi-irq
3	1	calculate EDT_S, MEDT_S, RDT_S_actual
4	0	calculate cdt_sx_nom, cdt_sx
5	0	calculate PSSM, rcdt_s, nmb_s_tar, start fast correction of missing pulses (if necessary).
6	0	calculate nmb_s for rmo=1 or smc=1, dmo=0, coa=0.
7	0	calculate nmb_s for rmo=1 or smc=1, dmo=0, coa=1.
8	0	calculate nmb_t for rmo=1 or smc=1, dmo=1.
9	0	
10	0	calculate add_in_cal1
10	1	write of add_in_cal1 finished, all subincrement calulations done for last active input event
11	0	calculate ts_s_check (MSI-irq), r_add_caln (prepare time stamp calculation(TS_S)) for IDT=IFP=1.
12	0	set caip1,2, action masking bits , action calculation loop
13	0	calculate NA(i),
14	0	calculate PDT_S(i)
14	1	calculate DTA(i)
15	0	calculate TSAC(i)
15	1	calculate PSAC(i)
15	2	action(i) in past condition occured: assignment of output data.
15	3	action loop control
16	0	wait for new action calculation

Figure 130 DPLL_STA.STA_T

STA_T(7:3)	STA_T(2:0)	Description/ Monitored action
0	0	Reset state
0	1	Wait, DEN=0
0	2	Calculation of 1/mlt+1, mls1, mls2.
0	3	calculation of direction change issues (pointers and profile update)
0	4	APT_2C was incremented
0	5	APT_2C was incremented
0	6	APT_2C was incremented
0	7	APT_2C was incremented. Update of pointers is finished, perform change of direction operations
1	0	pvt-check
1	1	update of RAM: write RDT_T; DT_T; TSF_T
1	2	loading of profile (syn_t, update syn_t_old) from ADT_T
1	3	TASI-irq, store FTV into RAM1b
1	4	Write PSTC; modify apt, apt_2b; apt_2c (if synchronized); Start fast pulse updates if necessary; Update inc_cnt1;
2	0	Write TS_T to ram1b, calculate dt_t_actual
3	0	update nti_cnt, cdti-irq if nti_cnt=0;
3	1	calculated EDT_T, MEDT_T, RDT_T_actual
4	0	calculate cdt_tx_nom, cdt_tx
5	0	calculate PSTM, rcdt_t, nmb_t_tar, start fast correction of missing pulses (if necessary) for rmo=0 or smc=1.
6	0	calculate nmb_t for rmo=0 or smc=1, dmo=0, coa=0.
7	0	calculate nmb_t for rmo=0 or smc=1, dmo=0, coa=1.
8	0	calculate nmb_t for rmo=0 or smc=1, dmo=1.
9	0	
10	0	calculate add_in_cal1
10	1	write of add_in_cal1 finished, all subincrement calulations done for last active input event
11	0	calculate ts_t_check (MTI-irq), r_add_caln (prepare time stamp calculation(TS_T)) for IDT=IFP=1.
12	0	set caip1,2, action masking bits , action calculation loop control.
13	0	calculate NA(i),
14	0	calculate PDT_T(i)
14	1	calculate DTA(i)
15	0	calculate TSAC(i)
15	1	calculate PSAC(i)
15	2	action(i) in past condition occured: assignment of output data.
15	3	action loop control
16	0	wait for new action calculation

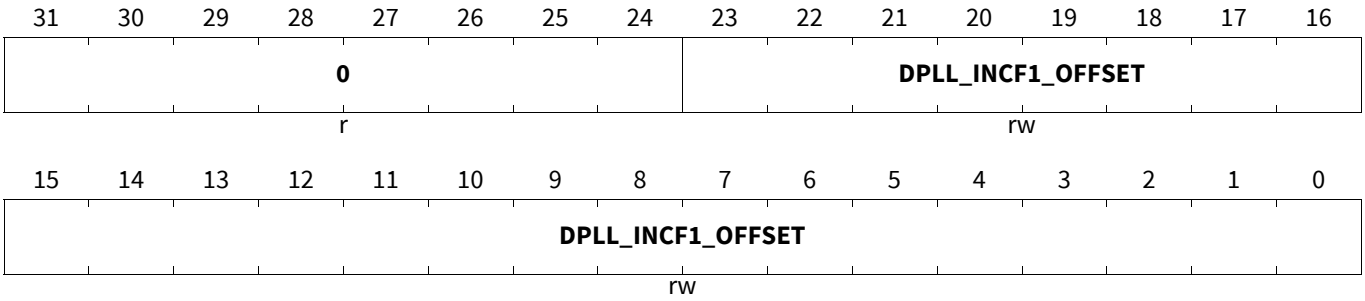
Figure 131 DPLL_STA.STA_S

28.20.12.104Register DPLL_INCF1_OFFSET

DPLL Start Value of the ADD_IN_ADDER1 Register

DPLL_INCF1_OFFSET

DPLL Start Value of the ADD_IN_ADDER1 Register(028F44$_H$) **Application Reset Value: 0000 0000$_H$**



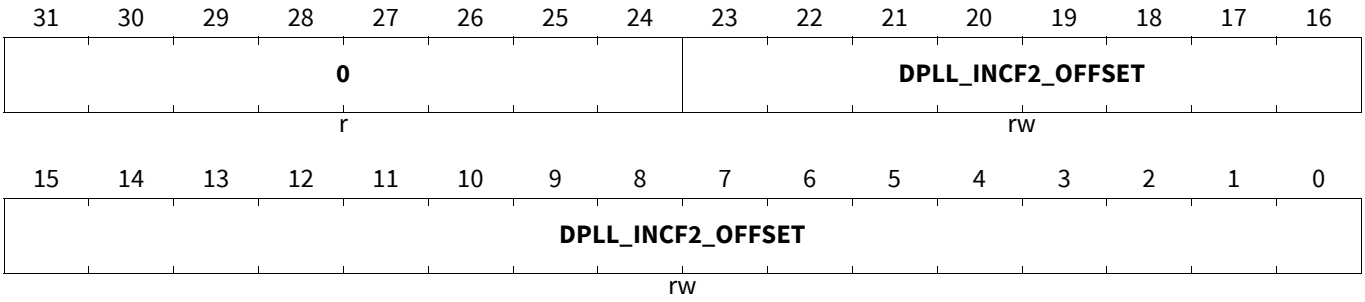
Field	Bits	Type	Description
DPLL_INCF1_OFFSET	23:0	rw	**Start value of the ADD_IN_ADDER1** In the case of set DPLL_CTRL_11-INCF1 the ADD_IN_ADDER1 starts always after an active new input event (TRIGGER in normal mode or STATE in emergency mode respectively) with this offset value. In the case of choosing DPLL_INCF1_OFFSET= 0xFFFFFF the generation of the first SUB_INC1 pulse is performed with the next TS_CLK. In the case of DPLL_INCF1_OFFSET= 0x000000 the first pulse is delayed by a full SUB_INC1 period and in the case of DPLL_INCF1_OFFSET= 0x7FFFFF the first pulse is delayed by a half SUB_INC1 period. Any other value is possible.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.105Register DPLL_INCF2_OFFSET

DPLL Start Value of the ADD_IN_ADDER2 Register

DPLL_INCF2_OFFSET

DPLL Start Value of the ADD_IN_ADDER2 Register(028F48$_H$) **Application Reset Value: 0000 0000$_H$**

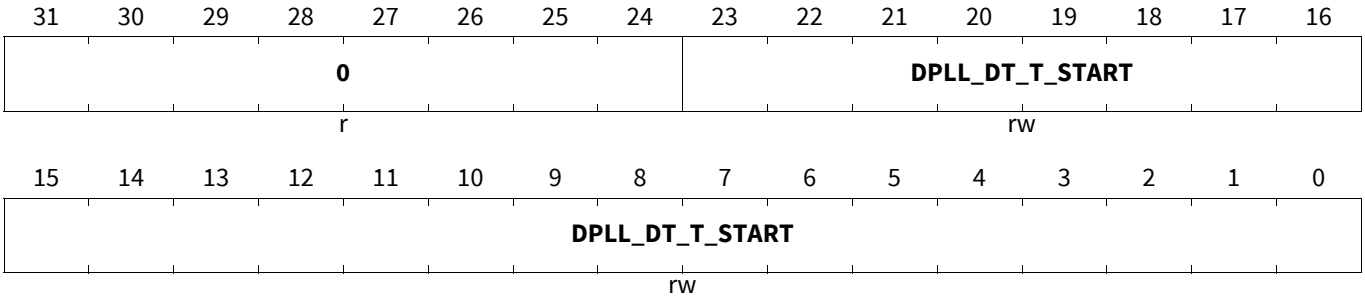


Field	Bits	Type	Description
DPLL_INCF2_OFFSET	23:0	rw	**Start value of the ADD_IN_ADDER2** In the case of set DPLL_CTRL_11-INCF2 the ADD_IN_ADDER2 starts always after an active new input event (STATE) with this offset value. In the case of choosing DPLL_INCF2_OFFSET= 0xFFFFFF the generation of the first SUB_INC2 pulse is performed with the next TS_CLK. In the case of DPLL_INCF2_OFFSET= 0x000000 the first pulse is delayed by a full SUB_INC2 period and in the case of DPLL_INCF2_OFFSET= 0x7FFFFF the first pulse is delayed by a half SUB_INC2 period. Any other value is possible.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.106Register DPLL_DT_T_START

DPLL Start Value of DPLL_DT_T_ACT for the First Increment after SIP1 is Set to 1

DPLL_DT_T_START
DPLL Start Value of DPLL_DT_T_ACT for the First Increment after SIP1 is Set to 1(028F4C$_H$) **Application**
Reset Value: 0000 0101$_H$

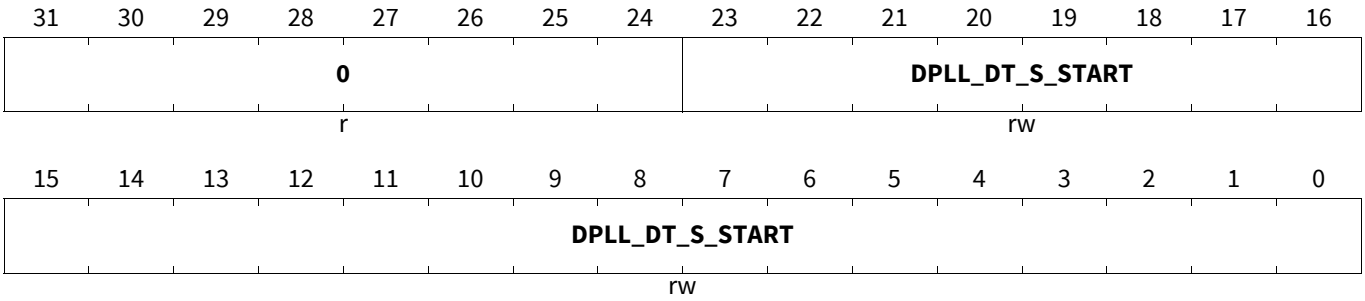


Field	Bits	Type	Description
DPLL_DT_T_START	23:0	rw	**Start value of DPLL_DT_T_ACT for the first increment after SIP1 is set to 1** For the first increment after setting SIP1 from 0 to 1, the value of DPLL_DT_T_START is taken instead of the calculated DPLL_DT_T_ACT for the current increment duration. This value should be always > 256 in order to avoid an overflow during the calculation of DPLL_RDT_T_ACT.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.107Register DPLL_DT_S_START

DPLL Start Value of DPLL_DT_S_ACT for the First Increment after SIP2 is Set to 1

DPLL_DT_S_START
DPLL Start Value of DPLL_DT_S_ACT for the First Increment after SIP2 is Set to 1(028F50$_H$) Application Reset Value: 0000 0101$_H$

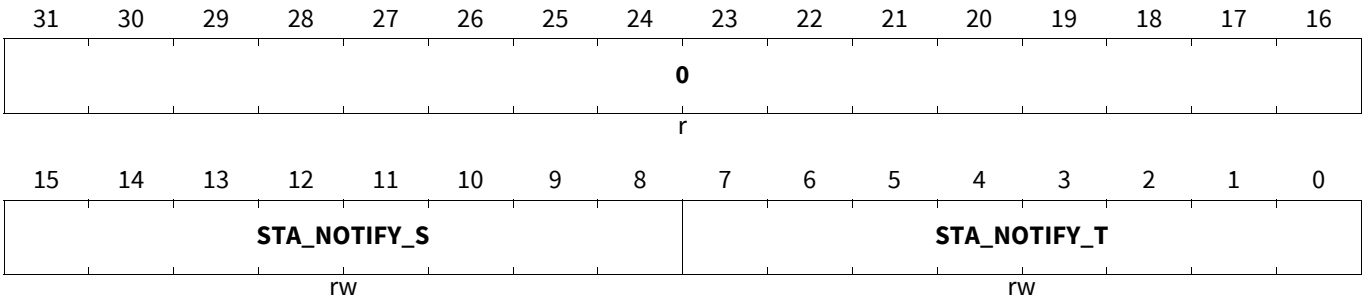


Field	Bits	Type	Description
DPLL_DT_S_START	23:0	rw	**Start value of DPLL_DT_S_ACT for the first increment after SIP2 is set to 1** For the first increment after setting SIP2 from 0 to 1, the value of DPLL_DT_S_START is taken instead of the calculated DPLL_DT_S_ACT for the current increment duration. This value should be always > 256 in order to avoid an overflow during the calculation of DPLL_RDT_S_ACT.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.108Register DPLL_STA_MASK

DPLL Trigger Masks for Signals DPLL_STA_T and DPLL_STA_S

DPLL_STA_MASK
DPLL Trigger Masks for Signals DPLL_STA_T and DPLL_STA_S(028F54$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
STA_NOTIFY_T	7:0	rw	**Notify value for STA_T of register DPLL_STA** The STA_NOTIFY_T is representing a trigger mask of DPLL_STA.STA_T. When DPLL_STA.STA_T reaches the value of STA_NOTIFY_T the flag DPLL_STA_FLAG.STA_FLAG_T is set to '1' when DPLL_STA.STA_T is leaving the state STA_NOTIFY_T.The signal is visible to MCS0 sub module as part of the special function register.
STA_NOTIFY_S	15:8	rw	**Notify value for STA_S of register DPLL_STA** The STA_NOTIFY_S is representing a trigger mask of DPLL_STA.STA_S. When DPLL_STA.STA_S reaches the value of STA_NOTIFY_S the flag DPLL_STA_FLAG.STA_FLAG_S is set to '1' when DPLL_STA.STA_S is leaving the state STA_NOTIFY_S.
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.109Register DPLL_STA_FLAG

DPLL STA Flag Register

DPLL_STA_FLAG
DPLL STA Flag Register **($028F58_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					INC_CNT2_FLAG	INC_CNT1_FLAG	STA_FLAG_S	0							STA_FLAG_T
r					rw	rw	rw	r							rw

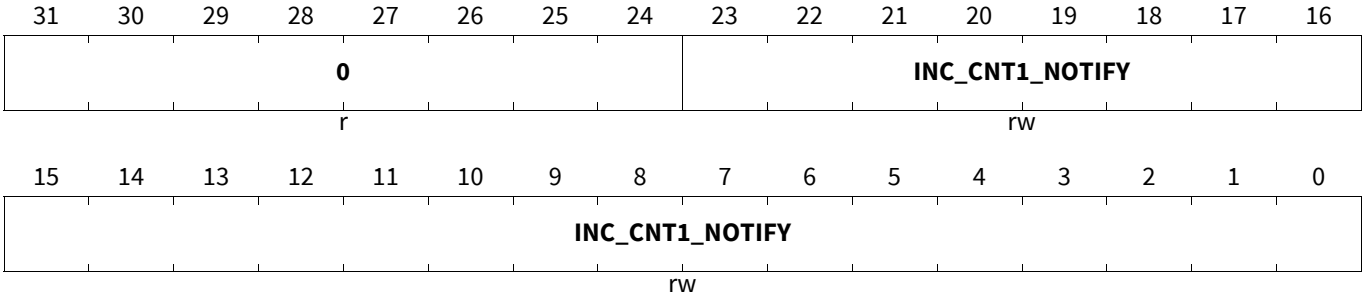
Field	Bits	Type	Description
STA_FLAG_T	0	rw	**Flag according to DPLL_MASK.STA_NOTIFY_T** The STA_FLAG_T is set to '1' indicating that the signal DPLL_STA.STA_T has left the state defined by the trigger mask of DPLL_STA_MASK.STA_NOTIFY_T. The Flag is reset when this bit of the register is written to '1'. The signal is visible to MCS0 sub module as part of the special function register.

Field	Bits	Type	Description
STA_FLAG_S	8	rw	**Flag according to DPLL_STA_MASK.STA_NOTIFY_S** The STA_FLAG_S is set to '1' indicating that the signal DPLL_STA.STA_S has left the state defined by the trigger mask of DPLL_STA_MASK.STA_NOTIFY_S. The Flag is reset when this bit of the register is written to '1'. The signal is visible to MCS0 sub module as part of the special function register.
INC_CNT1_FLAG	9	rw	**Flag according to DPLL_INC_CNT1_MASK.INC_CNT1_NOTIFY** The INC_CNT1_FLAG is set to '1' indicating that the signal DPLL_INC:CNT1.INC_CNT1 has left the state defined by the trigger mask of DPLL_INC_CNT1_MASK.INC_CNT1_NOTIFY. The Flag is reset when this bit of the register is written to '1'. The signal is visible to MCS0 sub module as part of the special function register.
INC_CNT2_FLAG	10	rw	**Flag according to DPLL_INC_CNT2_MASK.INC_CNT2_NOTIFY** The INC_CNT2_FLAG is set to '1' indicating that the signal DPLL_INC_CNT2.INC_CNT2 has left the state defined by the trigger mask of DPLL_INC_CNT2_MASK.INC_CNT2_NOTIFY. The Flag is reset when this bit of the register is written to '1'. The signal is visible to MCS0 sub module as part of the special function register.
0	7:1, 31:11	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.110Register DPLL_INC_CNT1_MASK

DPLL INC_CNT1 Trigger Mask

DPLL_INC_CNT1_MASK
DPLL INC_CNT1 Trigger Mask **($028F5C_H$)** **Application Reset Value: $0000\ 0000_H$**



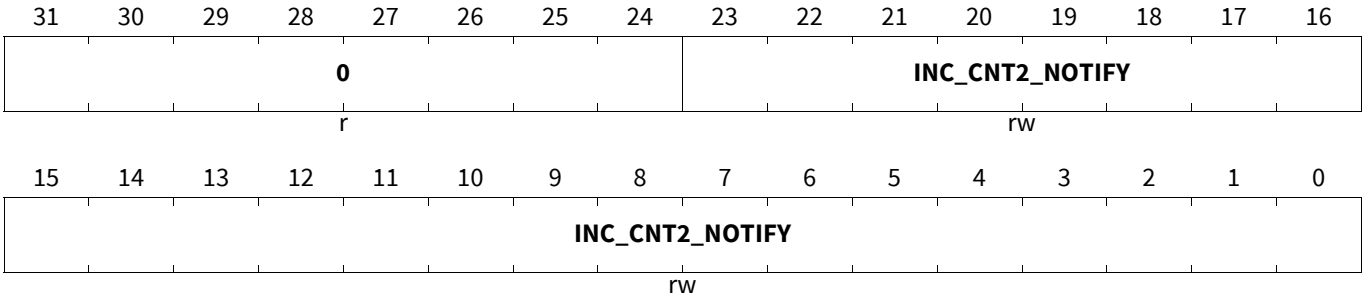
Field	Bits	Type	Description
INC_CNT1_NOTIFY	23:0	rw	**Notify value for INC_CNT1 of register DPLL_INC_CNT1** The INC_CNT1_NOTIFY is representing a trigger mask of DPLL_INC_CNT1.INC_CNT1. When DPLL_INC_CNT1.INC_CNT1 reaches the value of INC_CNT1_NOTIFY the flag DPLL_STA_FLAG.INC_CNT1_FLAG is set to '1' when DPLL_INC_CNT1.INC_CNT1 is leaving the state INC_CNT1_NOTIFY.

Field	Bits	Type	Description
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.111Register DPLL_INC_CNT2_MASK

DPLL INC_CNT2 Trigger Mask

DPLL_INC_CNT2_MASK
DPLL INC_CNT2 Trigger Mask **(028F60$_H$)** **Application Reset Value: 0000 0000$_H$**

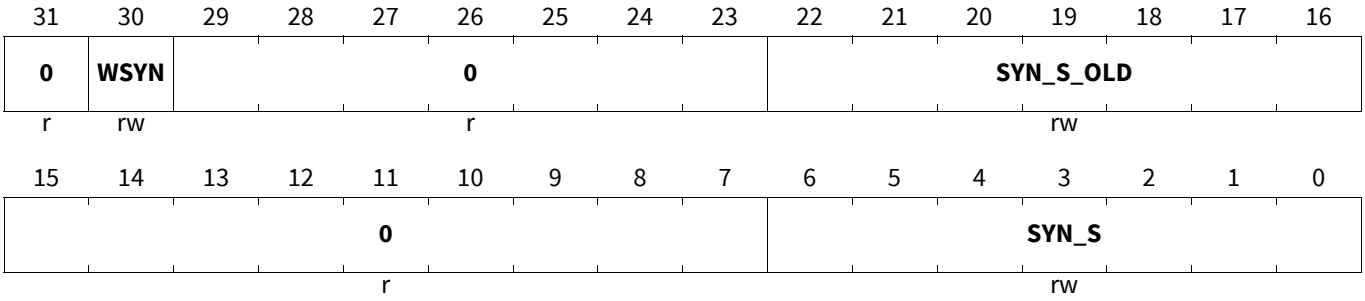


Field	Bits	Type	Description
INC_CNT2_NOTIFY	23:0	rw	**Notify value for INC_CNT2 of register DPLL_INC_CNT2** The INC_CNT2_NOTIFY is representing a trigger mask of DPLL_INC_CNT2.INC_CNT2. When DPLL_INC_CNT2.INC_CNT2 reaches the value of INC_CNT2_NOTIFY the flag DPLL_STA_FLAG.INC_CNT2_FLAG is set to '1' when DPLL_INC_CNT2.INC_CNT2 is leaving the state INC_CNT2_NOTIFY.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.112Register DPLL_NUSC_EXT1

DPLL Extension Register Number 1 for DPLL_NUSC 4

DPLL_NUSC_EXT1
DPLL Extension Register Number 1 for DPLL_NUSC 4(028F64$_H$) **Application Reset Value: 0001 0001$_H$**



Field	Bits	Type	Description
SYN_S	6:0	rw	**Number of real and virtual events to be considered for the current increment** This value reflects the NS value of the last valid increment, stored in ADT_S[i]; to be updated after all calculations in step 37 of Table **Section 28.20.8.6.6**. This value can only be written when the WSYN bit in this register is set.
SYN_S_OLD	22:16	rw	**Number of real and virtual events to be considered for the last increment** This value reflects the NS value of the last but one valid increment, stored in ADT_S[i]; is updated automatically when writing SYN_S. This value is updated by the SYN_S value when the WSYN bit in this register is set.
WSYN	30	rw	**Write control bit for SYN_S and SYN_S_OLD** Read as zero. 0_B The SYN_S value is not writeable 1_B The SYN_S value is writeable
0	15:7, 29:23, 31	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.113Register DPLL_NUSC_EXT2

DPLL Extension Register Number 2 for DPLL_NUSC 4

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set, any read/write access to this register will return AEI_STATUS = 0b10.

DPLL_NUSC_EXT2

DPLL Extension Register Number 2 for DPLL_NUSC 4($028F68_H$) **Application Reset Value: $0000\ 0001_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WVSN	**0**	**WNUS**	**0**						**VSN**						
rw	r	rw	r						rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FSS	**0**								**NUSE**						
rw	r								rw						

Field	Bits	Type	Description
NUSE	6:0	rw	**Number of recent STATE events used for SUB_INCx calculations modulo 2*(SNUmax+1)** No gap is considered in that case for this value, but in the VSN value (see below): This register is set by the CPU but reset automatically to “1” by a change of direction or loss of LOCK. Each other value can be set by the CPU, maybe Full_SCALE, HALF_SCALE or parts of them. The relation values QDT_Sx are calculated using NUSE values in the past with its maximum value of 2*SNU+1. This value can only be written when the WNUS bit is set.
FSS	15	rw	**This value is to be set, when NUSE is set to FULL_SCALE** This value is set by the CPU, but reset automatically to '0' by a change of direction or loss of LOCK. This value can only be written when the WNUS bit is set. 0_B The NUSE value is less then FULL_SCALE 1_B The NUSE value is equal to FULL_SCALE
VSN	22:16	rw	**Number of virtual state increments in the current NUSE region** This value reflects the number of virtual increments in the current NUSE region; for NUSE=1 this value is zero, when the CPU sets NUSE to a value > 1 or zero(2^7 modulo 2^7) , it must also set VSN to the correspondent value;the VSN value is subtracted from the NUSE value in order to get the corresponding APS value for the past; the VSN value is not used for the APS_1C2 pointer. VSN is to be updated by the CPU when a new gap is to be considered for NUSE or a gap is leaving the NUSE region; for this purpose the SASI interrupt can be used; no further update of VSN is necessary when NUSE is set to FULL_SCALE. This value can only be written when the WVSN bit is set.
WNUS	29	rw	**Write control bit for NUSE** Read as zero. 0_B The NUSE value is not writeable 1_B The NUSE value is writeable
WVSN	31	rw	**Write control bit for VSN** Read as zero. 0_B The VSN value is not writeable 1_B The VSN value is writeable
0	14:7, 28:23, 30	r	**Reserved** Read as zero, shall be written as zero.

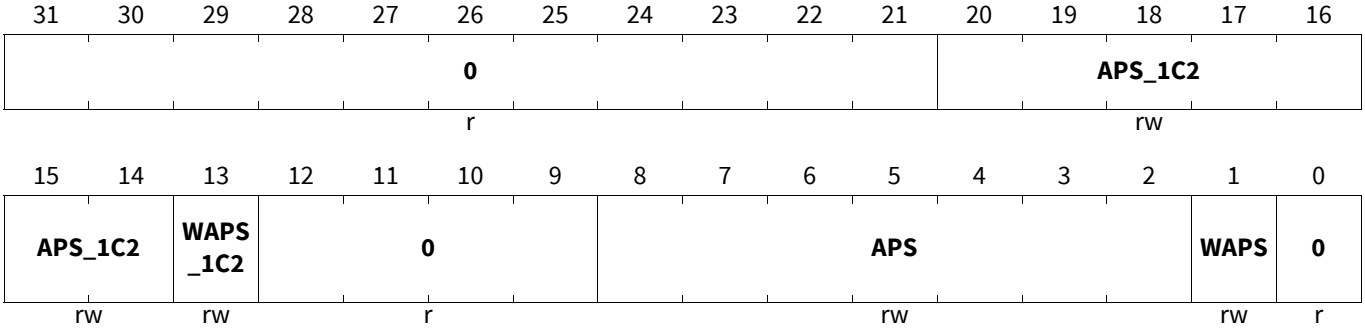
28.20.12.114Register DPLL_APS_EXT

DPLL Extension Register for DPLL_APS

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set, any read/write access to this register will return AEI_STATUS = 0b10.

DPLL_APS_EXT

DPLL Extension Register for DPLL_APS (028F38$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
WAPS	1	rw	**Write bit for address pointer APS** Read as zero. 0_B The APS is not writeable 1_B The APS is writeable
APS	8:2	rw	**Actual RAM pointer address value for DT_S[i] and RDT_S[i]** Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 128 *STATE* events but limited to 2*(SNU+1-SYN_NS) in normal and emergency mode for SYSF=0 or to 2*(SNU+1)-SYN_NS for SYSF=1 respectively; See **Section 28.20.10**. APS is incremented (decremented) by one for each active *STATE* event and DIR2=0 DIR2=1). The APS offset value is added in the above shown bit position with the subsection offset of the RAM region. The APS pointer value is directed to the RAM position, in which the data values are to be written, which correspond to the last increment. The APS value is not to be changed, when the direction (shown by DIR2) changes, because it points always to a storage place after the considered increment. Changing of DIR2 takes place always after an active *STATE* event and the resulting increment/decrement. This value can only be written when the WAPS bit is set.
WAPS_1C2	13	rw	**Write bit for address pointer APS_1C2** Read as zero. 0_B The APS_1C2 is not writeable 1_B The APS_1C2 is writeable

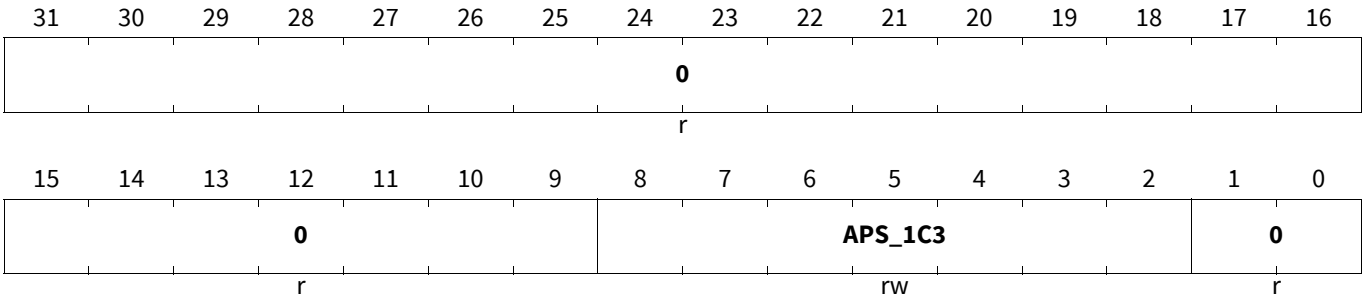
Field	Bits	Type	Description
APS_1C2	20:14	rw	**Actual RAM pointer address value for TSF_S[i]** Initial value: zero (0x00). Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 64 *STATE* events but limited to 2*(SNU+1) in normal and emergency mode; this pointer is used for the RAM region 1c2. For SYS=1: APS_1C2 is incremented (decremented) by SYN_S_OLD for each active *STATE* event and DIR2=0 (DIR2=1). For SYS=0: APT_1c2 is incremented or decremented by 1 respectively. The APS_1C2 offset value is added in the above shown bit position with the subsection offset of the RAM region. In addition when the APS_1C3 value is written by the CPU - in order to synchronize the DPLL- with the next active *STATE* event the APS_1C2_EXT value is added/subtracted (while APS_1C2_STATUS is one; see DPLL_APT_SYNC register at **Section 28.20.12.25**). This value can only be written when the WAPS_1C2 bit is set
0	0, 12:9, 31:21	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.115Register DPLL_APS_1C3_EXT

DPLL Extension Register for DPLL_APS_1C3

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set, any read/write access to this register will return AEI_STATUS = 0b10.

DPLL_APS_1C3_EXT
DPLL Extension Register for DPLL_APS_1C3 (028F3C$_H$) Application Reset Value: 0000 0000$_H$



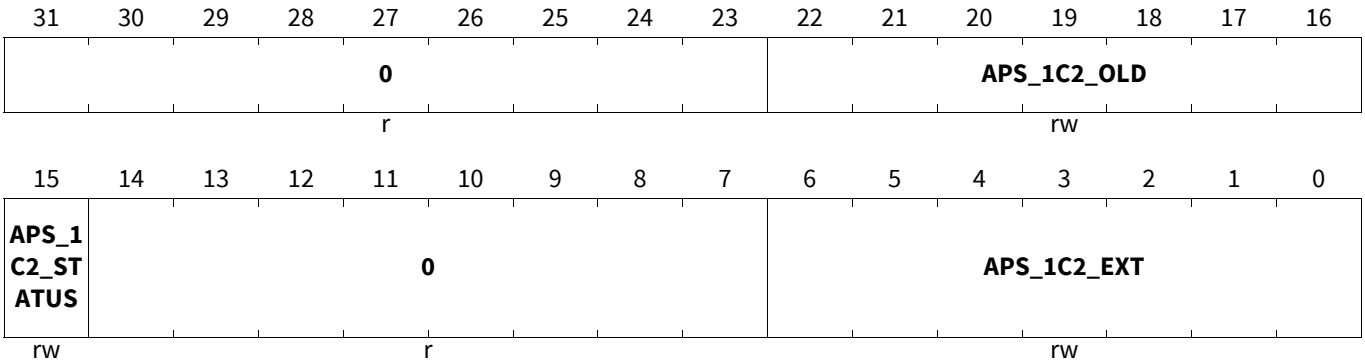
Field	Bits	Type	Description
APS_1C3	8:2	rw	**Actual RAM pointer address value for ADT_S[i]** Initial value: zero (0x00). Actual RAM pointer and synchronization position/value of *STATE* events in FULL_SCALE for up to 128 *STATE* events but limited to 2*(SNU+1-SYN_NS) in normal and emergency mode for SYSF=0 or to 2*(SNU+1)-SYN_NS for SYSF=1 respectively; this pointer is used for the RAM region 1c3. See **Section 28.20.10**. The RAM pointer is set by the CPU accordingly, when the synchronization condition was detected. The APS_1C3 pointer value is directed to the RAM position of the profile element in RAM region 1c2, which corresponds to the current increment. When changing the direction DIR1 or DIR2 respectively, this is always known before an active *STATE* event is processed. This is because of the pattern recognition in SPE (for PMSM) or because of the direction change recognition by TRIGGER. This direction change results in an automatic increment (forwards) or decrement (backwards) when the input event occurs in addition with a 2 times correction. The APS_1C3_x offset value is added in the above shown bit position with the subsection address offset of the corresponding RAM region.
0	1:0, 31:9	r	**Reserved** Read as zero, shall be written as zero.

28.20.12.116Register DPLL_APS_SYNC_EXT

DPLL Extension Register for DPLL_APS_SYNC

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set, any read/write access to this register will return AEI_STATUS = 0b10.

DPLL_APS_SYNC_EXT
DPLL Extension Register for DPLL_APS_SYNC (028F30$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
APS_1C2_EXT	6:0	rw	**Address pointer 1c2 extension** This offset value determines, by which value the APS_1C2 is changed at the synchronization time; set by CPU before the synchronization is performed. This offset value is the number of virtual increments to be inserted in the TSF for an imminent intended synchronization; the CPU sets its value dependent on the gaps until the synchronization time taking into account the considered NUSE value to be set and including the next future increment (when SYN_S_OLD is still 1). When the synchronization takes place, this value is to be added to the APS_1C2 address pointer (for forward direction, DIR2=0) and the APT_1c2_status bit is cleared after it. For backward direction subtract APS_1C2_EXT accordingly. When the synchronization is intended and the NUSE value is to be set to FULL_SCALE after it, the APS_1C2_EXT value must be set to SYN_NS (for SYSF=1) or 2*SYN_NS (for SYSF=0) in order to be able to fill all gaps in the extended TSF_S with the corresponding values by the CPU. When still not all values for FULL_SCALE are available, the APS_1C2_EXT value considers only a share according to the NUSE value to be set after the synchronization.
APS_1C2_STATUS	15	rw	**Address pointer 1c2 status** Set by CPU before the synchronization is performed. The value is cleared automatically when the APS_1C2_OLD value is written. 0_B APS_1C2_EXT is not to be considered 1_B APS_1C2_EXT has to be considered for time stamp field extension
APS_1C2_OLD	22:16	rw	**Address pointer STATE for RAM region 1c2 at synchronization time** This value is set by the current APS_1C2 value when the synchronization takes place for the first active STATE event after writing APS_1C3 but before adding the offset value APS_1C2_EXT (that means: when APS_1C2_STATUS=1). Address pointer APS_1C2 value at the moment of synchronization, before the offset value is added, that means the pointer with this value points to the last value before the additional inserted gap.
0	14:7, 31:23	r	**Reserved** Read as zero, shall be written as zero.

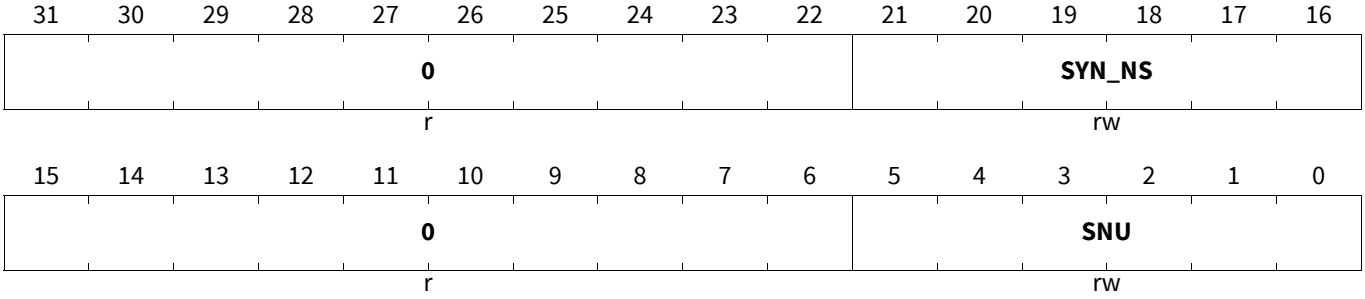
28.20.12.117Register DPLL_CTRL_EXT

DPLL Extension Register for DPLL_CTRL

Note: This register is only used when DPLL_CTRL_11.STATE_EXT is set. If DPLL_CTRL_11.STATE_EXT is not set, any read/write access to this register will return AEI_STATUS = 0b10.

DPLL_CTRL_EXT

DPLL Extension Register for DPLL_CTRL (028F34$_H$) Application Reset Value: 0000 0017$_H$



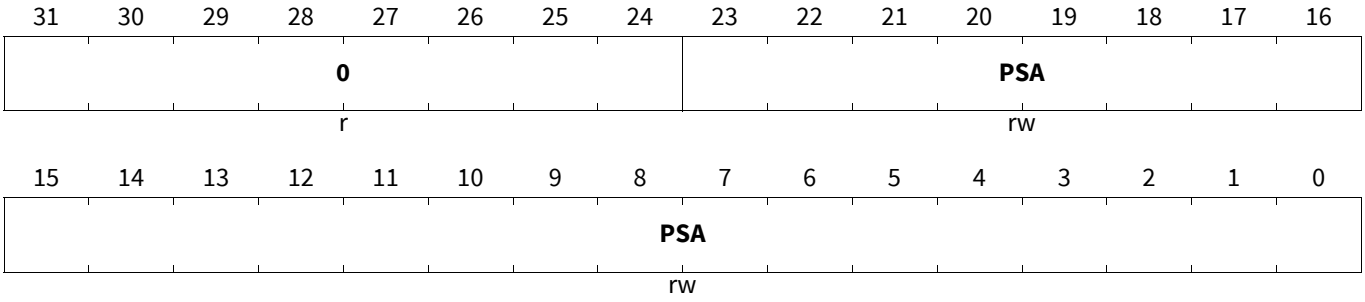
Field	Bits	Type	Description
SNU	5:0	rw	**STATE number** This bit can only be written when the DPLL is disabled. The number of nominal *STATE* events is the decimal value plus 1. This value can only be written when (RMO=0 and SMC=0) or DEN=0. Set SSL=00 before changing this value and set RMO=1 only after FULL_SCALE with SSL>0.
SYN_NS	21:16	rw	**Synchronization number of STATE** Summarized number of virtual increments in HALF_SCALE. Sum of all systematic missing *STATE* events in HALF_SCALE (for SYSF=0) or FULL SCALE (for SYSF=1) ; the SYN_NS missing *STATES* can be divided up to an arbitrary number of blocks. The pattern of events and missing events in FULL_SCALE is shown in RAM region 1c3 as value NS in addition to the adapted values. The number of stored increments in FULL_SCALE must be equal to 2*(SNU+1-SYN_NS) for SYSF=0 or 2*(SNU+1)-SYN_NS for SYSF=1 . This pattern is written by the CPU beginning from a fixed reference point (maybe beginning of the FULL_SCALE region). The relation to the actual increment is established by setting of the profile RAM pointer APS_1C3 in an appropriate relation to the RAM pointer APS of the actual increment by the CPU. This value can only be written when the DPLL is disabled. This value can only be written when (RMO=0 and SMC=0) or DEN=0. Set SSL=00 before changing this value and set RMO=1 only after FULL_SCALE with SSL>0.
0	15:6, 31:22	r	**Reserved** Read as zero, shall be written as zero.

28.20.13 DPLL RAM Region 1a value description

28.20.13.1Memory DPLL_PSA[i]

DPLL ACTION_i Position/Value Request

DPLL_PSAi (i=0-31)
DPLL ACTION_i Position/Value Request **(028200_H+i*4)** **Reset Value: Table 137**



Field	Bits	Type	Description
PSA	23:0	rw	**Position information of a desired action i** This value can only be written when the DPLL is disabled. The PSA values for actions 24...31 are not available for all devices but depends on specific product configuration.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

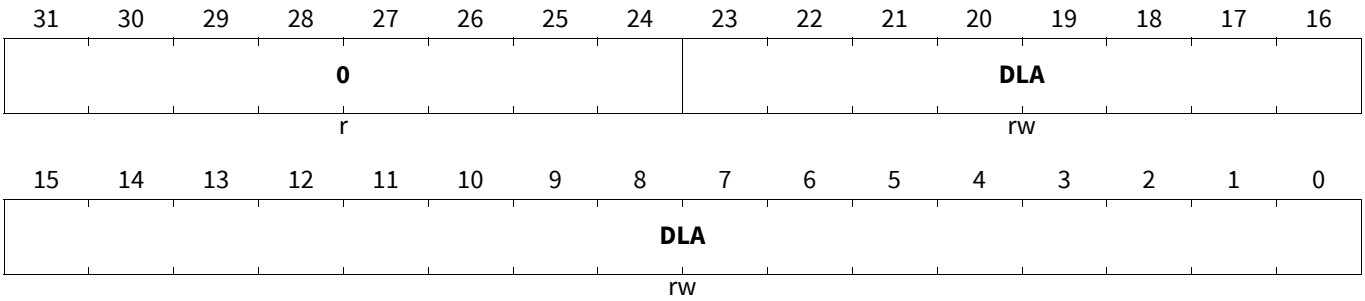
Table 137 Reset Values of DPLL_PSAi (i=0-31)

Reset Type	Reset Value	Note
PowerOn Reset	XXXX $XXXX_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1A	0000 0000_H	Cleared by state machine

28.20.13.2Memory DPLL_DLA[i]

DPLL ACTION_i Time to React before PSAi

DPLL_DLAi (i=0-31)
DPLL ACTION_i Time to React before PSAi **(028280_H+i*4)** **Reset Value: Table 138**



Field	Bits	Type	Description
DLA	23:0	rw	**Time to react before the corresponding position value of a desired action i is reached** In the case of LOW_RES=1 (see **Table 70**), this delay value must be also given as low resolution value. This value can only be written when the DPLL is disabled.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

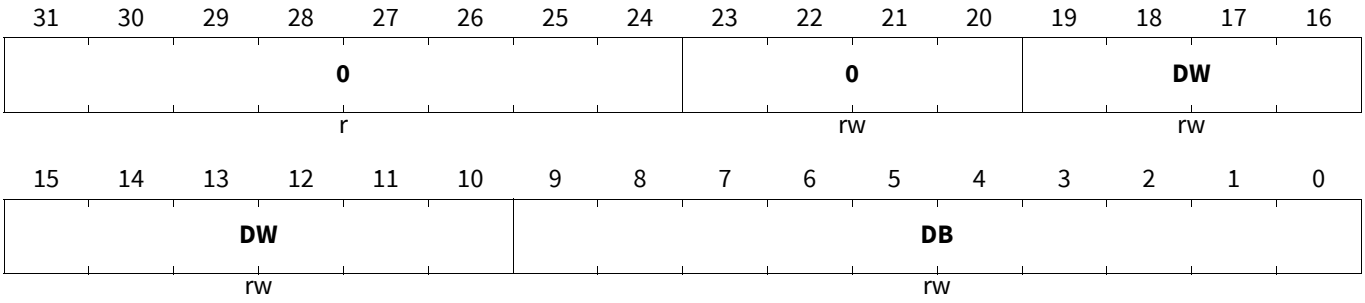
Table 138 Reset Values of DPLL_DLAi (i=0-31)

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1A	0000 0000$_H$	Cleared by state machine

28.20.13.3Memory DPLL_NA[i]

DPLL Calculated Number of TRIGGER/STATE Increments to ACTION_i

DPLL_NAi (i=0-31)
DPLL Calculated Number of TRIGGER/STATE Increments to ACTION_i(028300$_H$+i*4) Reset Value: Table 139



Field	Bits	Type	Description
DB	9:0	rw	**Number of events to Action_i (fractional part)** The NA values for actions 24...31 are only available for device 4 or 5. This value can only be written when the DPLL is disabled.
DW	19:10	rw	**Number of events to Action_i (integer part)** Use the maximum value for NA_DW=0x3FF in the case of a calculated value which exceeds the represent able value. This value can only be written when the DPLL is disabled.
0	23:20	rw	**Reserved** Read as zero, shall be written as zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

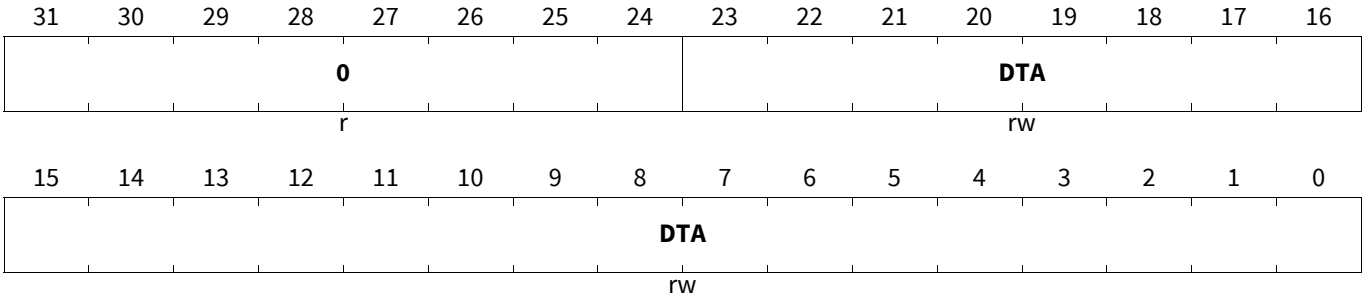
Table 139 Reset Values of DPLL_NAi (i=0-31)

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1A	0000 0000$_H$	Cleared by state machine

28.20.13.4Memory DPLL_DTA[i]

DPLL Calculated Relative TIME to ACTION_i

DPLL_DTAi (i=0-31)
DPLL Calculated Relative TIME to ACTION_i (028380$_H$+i*4) **Reset Value: Table 140**



Field	Bits	Type	Description
DTA	23:0	rw	**Calculated relative time to ACTION_i** This value can only be written when the DPLL is disabled. The DTA value is a positive integer value. When calculations using equations DPLL-12 or DPLL-14 result in a negative value, it is replaced by zero.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

Table 140 Reset Values of DPLL_DTAi (i=0-31)

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_1A	0000 0000$_H$	Cleared by state machine

28.20.14 DPLL RAM Region 2 value description

Bits 31 to 24 of RAM region 2 are not implemented and therefore always read as zero (reserved). Other bits which are declared as reserved are not protected against writing. Unused address regions are not protected against writing when implemented.

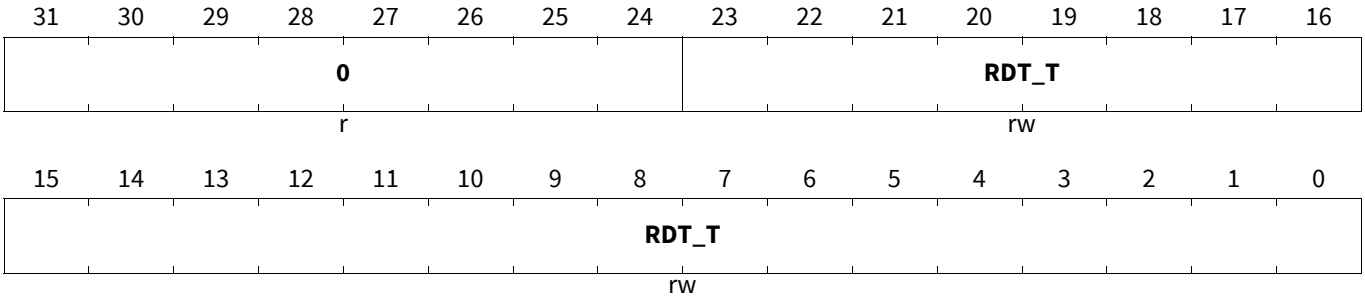
28.20.14.1Memory DPLL_RDT_T[i]

Region 2a. Reciprocal value of the corresponding successive increment i, for each true nominal increment.

Reciprocal Values of the Nominal TRIGGER Increments Duration in FULL_SCALE

Note: The starting index for Memory DPLL_RDT_T[i] in RAM2 is defined by the parameter AOSV_2A in DPLL_AOSV2 Register

DPLL_RDT_Ti
Region 2a. Reciprocal value of the corresponding successive increment i, for each true nominal increment.
($02C000_H$) **Reset Value: Table 141**



Field	Bits	Type	Description
RDT_T	23:0	rw	**Reciprocal difference time of TRIGGER; 2* (TNU+1- SYN_NT) stored values nominal reciprocal value of the number of time stamp clocks measured in the corresponding increment (which is divided by the number of nominal increments); multiplied by *232 while only the lower 24 bits are used; the LSB is rounded up, when the next truncated bit is 1.** RDT_T: Reciprocal difference time of TRIGGER; 2* (TNU+1- SYN_NT) stored values nominal reciprocal value of the number of time stamp clocks measured in the corresponding increment (which is divided by the number of nominal increments); multiplied by *232 while only the lower 24 bits are used; the LSB is rounded up, when the next truncated bit is 1. **Note:** There are 2* (TNU+1- SYN_NT) entries. The maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.
0	31:24	r	

Table 141 Reset Values of DPLL_RDT_Ti

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_2	--00 0000$_H$	Cleared by state machine

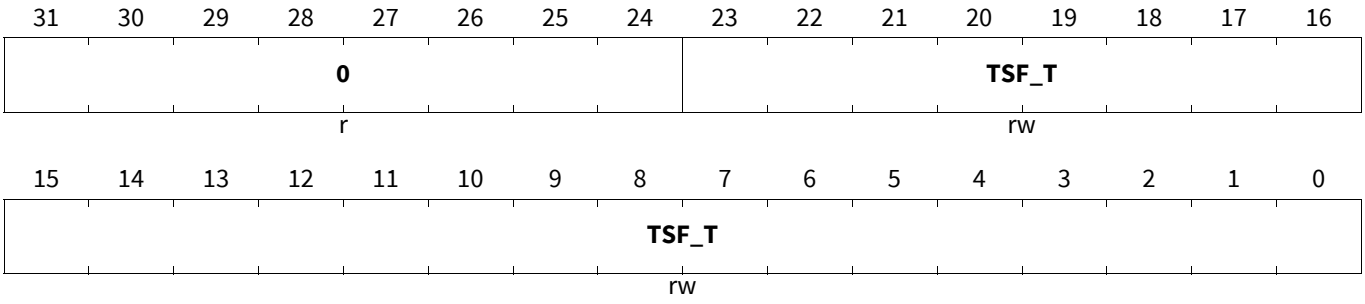
28.20.14.2Memory DPLL_TSF_T[i]

Region 2b. Time Stamp Field for TRIGGER event i, for each true nominal increment plus each virtual increment.

Time Stamp Values of the Nominal TRIGGER Increments in FULL_SCALE

Note: The starting index for Memory DPLL_TSF_T[i] in RAM2 is defined by the parameter AOSV_2C in DPLL_AOSV2 Register

DPLL_TSF_Ti
Region 2b. Time Stamp Field for TRIGGER event i, for each true nominal increment plus each virtual increment. **(02C000$_H$)** **Reset Value: Table 142**



Field	Bits	Type	Description
TSF_T	23:0	rw	**Time stamp field of active TRIGGER slopes** TSF_T: Time stamp field of active TRIGGER slopes **Note:** There are 2* (TNU+1) entries. The maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.
0	31:24	r	

Table 142 Reset Values of DPLL_TSF_Ti

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_2	--00 0000$_H$	Cleared by state machine

28.20.14.3Memory DPLL_ADT_T[i]

Region 2c. Adapt values for the current TRIGGER increment i, for each true nominal increment.

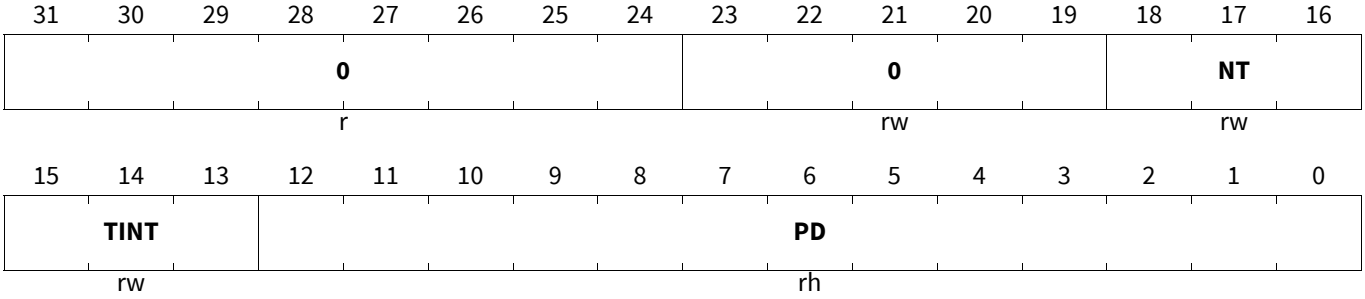
Adapt and Profile Values of the TRIGGER Increments in FULL_SCALE

Note: The starting index for Memory DPLL_ADT_T[i] in RAM2 is defined by the parameter AOSV_2C in DPLL_AOSV2 Register

DPLL_ADT_Ti

Region 2c. Adapt values for the current TRIGGER increment i, for each true nominal increment.($02C000_H$)

Reset Value: Table 143



Field	Bits	Type	Description
PD	12:0	rh	**Physical deviation; Adapt values for each nominal TRIGGER increment in FULL_SCALE (sint13);** PD: Physical deviation; Adapt values for each nominal TRIGGER increment in FULL_SCALE (sint13); The PD value does mean the number of SUB_INC1 pulses to be added to NT*((MLT+1) + PD); the absolute value of a negative PD must not exceed (MLT+1) or MLS1 respectively;systematic missing *TRIGGER* events must be considered for the value of PD;
TINT	15:13	rw	**TRIGGER Interrupt information** Depending on the value, up to 7 different interrupts can be generated. In the current version, the 5 interrupts TE0_IRQ ... TE4_IRQ are supported by TINT=001_B, 010_B, 011_B, 100_B and 101_B, respectively. For the values 000_B, 110_B and 111_B, no interrupt is generated and no other reaction is performed. The corresponding interrupt is activated when the TINT value is read by the DPLL together with the other values (PD, NT) according to the profile.
NT	18:16	rw	**Number of TRIGGERs; number of nominal TRIGGER parts in the corresponding increment.** NT: Number of TRIGGERs; number of nominal TRIGGER parts in the corresponding increment. **Note:** There are 2* (TNU+1- SYN_NT) entries. The maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.
0	23:19	rw	**Not used - NOT_USED** Not used Note: must be written to zero.
0	31:24	r	

Table 143 Reset Values of DPLL_ADT_Ti

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_2	--00 0000$_H$	Cleared by state machine

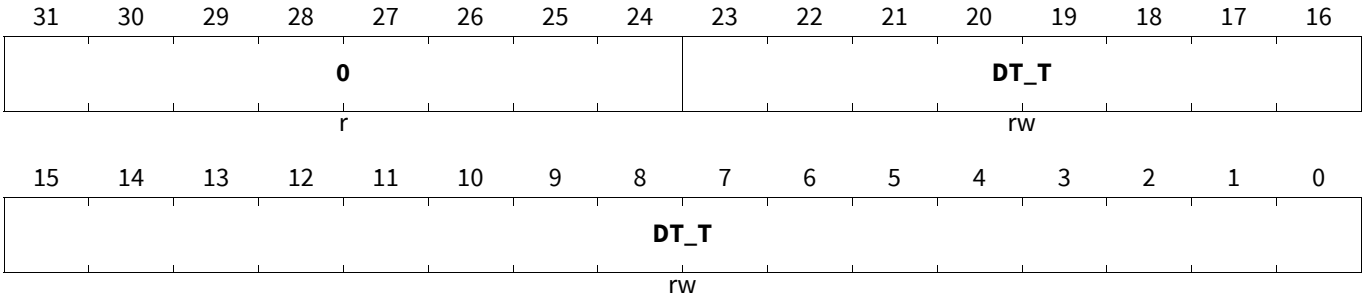
28.20.14.4Memory DPLL_DT_T[i]

Region 2d. Uncorrected last increment value of TRIGGER i, for each true nominal increment.

Nominal TRIGGER Increments Duration in FULL_SCALE

Note: The starting index for Memory DPLL_DT_T[i] in RAM2 is defined by the parameter AOSV_2D in DPLL_AOSV2 Register

DPLL_DT_Ti
Region 2d. Uncorrected last increment value of TRIGGER i, for each true nominal increment.(02C000$_H$)
Reset Value: Table 144



Field	Bits	Type	Description
DT_T	23:0	rw	**Difference time of TRIGGER; increment duration values for each TRIGGER increment in FULL_SCALE divided by the number of nominal increments (nominal value).** DT_T: Difference time of TRIGGER; increment duration values for each TRIGGER increment in FULL_SCALE divided by the number of nominal increments (nominal value). **Note:** There are 2* (TNU+1- SYN_NT) entries. The maximum number of entries is restricted to a value corresponding to the OSS value in the DPLL_OSW register.
0	31:24	r	

Table 144 Reset Values of DPLL_DT_Ti

Reset Type	Reset Value	Note
PowerOn Reset	XXXX XXXX$_H$	Undefined after Power On
DPLL_RAM_INI.INIT_2	--00 0000$_H$	Cleared by state machine

28.20.15 MCS to DPLL Register description

28.20.15.1Register MCS2DPLL_DEB0

MCS to DPLL Data Exchange Buffer 0

READ access from MCS: Duration of the last increment DT_S_ACT (**Update of RAM in Normal and Emergency Mode**). This value is updated by the DPLL during the update of ram (STA_S = 0b0000_1001) and is ready to be read when STA_S is modified to 0b0000_1010.WRITE access from MCS: The DPLL expects DT_S[p-1] (**Equation DPLL-8 to calculate the error of last prediction**) or DT_S[p+1](**Equation DPLL-8a to calculate the error of the last prediction**) during the increment prediction (STA_S = 0b0001_0000).

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

MCS2DPLL_DEB0

MCS to DPLL Data Exchange Buffer 0 (007800_H) PowerOn Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								DATA							
r								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															
rw															

Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 0. Actual content depends on whether it is a Read or Write operation from MCS.** DATA: Data exchange buffer 0. Actual content depends on whether it is a Read or Write operation from MCS.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.15.2Register MCS2DPLL_DEB1

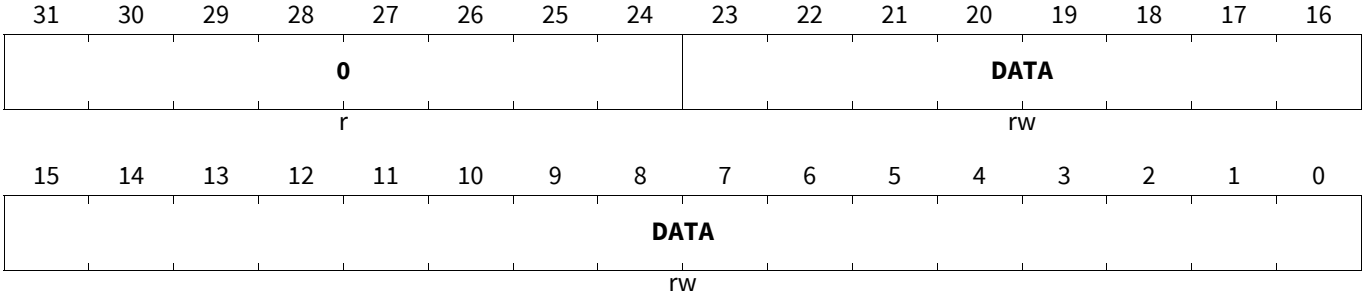
MCS to DPLL Data Exchange Buffer 1

READ access from MCS: Reads as 0.WRITE access from MCS: The DPLL expects RDT_S[p-1] (**Equation DPLL-7a1 to calculate QDT_S_ACT**) or RDT_S[p+1](**Equation DPLL-7a2 to calculate QDT_S_ACT backwards**) during the increment prediction (STA_S = 0b0001_0000) and RDT_S[t-1] (**Action calculations for STATE forwards**) or RDT_S[t+1](**Action calculations for STATE backwards**) during the action calculation (STA_S = 0b0111_0000)

Note: In both cases, the data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

MCS2DPLL_DEB1

MCS to DPLL Data Exchange Buffer 1 **(007804_H)** **PowerOn Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 1.** DATA: Data exchange buffer 1.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.15.3Register MCS2DPLL_DEB2

MCS to DPLL Data Exchange Buffer 2

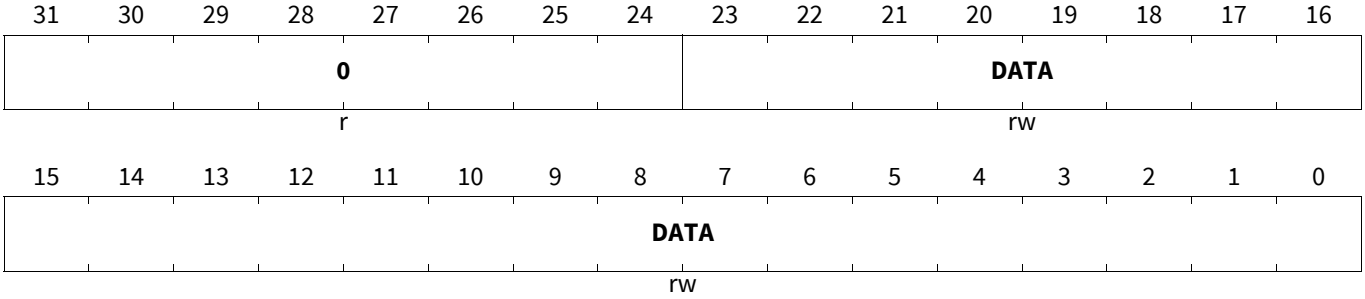
READ access from MCS: TS_Sx (**Equation DPLL-6a4 to update the time stamp values for STATE**). Use to compute/update the time stamp values for STATE during the update of ram (STA_S = 0b0000_1001)WRITE access from MCS: The DPLL expects RDT_S[p-q-1] (**Equation DPLL-8 to calculate the error of last prediction**) or RDT_S[p+q+1](**Equation DPLL-8a to calculate the error of the last prediction**) during the increment prediction (STA_S = 0b0001_0000).

Note: The data read from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

MCS2DPLL_DEB2

MCS to DPLL Data Exchange Buffer 2 **(007808_H)** **PowerOn Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 2. Actual content depends on whether it is a Read or Write operation from MCS.** DATA: Data exchange buffer 2. Actual content depends on whether it is a Read or Write operation from MCS.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.15.4Register MCS2DPLL_DEB3

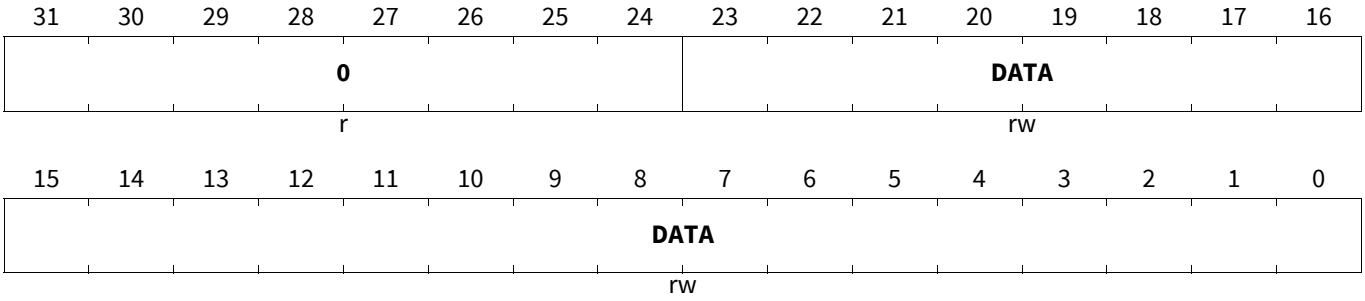
MCS to DPLL Data Exchange Buffer 3

READ access from MCS: DT_Sx (**Equation DPLL-6a4 to update the time stamp values for STATE**). Use to compute/update the time stamp values for STATE during the update of ram (STA_S = 0b0000_1001)WRITE access from MCS: The DPLL expects DT_S[p-q] (**Equation DPLL-8 to calculate the error of last prediction**) or DT_S[p+q](**Equation DPLL-8a to calculate the error of the last prediction**) during the increment prediction (STA_S = 0b0001_0000).

Note: The data read from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

MCS2DPLL_DEB3

MCS to DPLL Data Exchange Buffer 3 (00780C$_H$) PowerOn Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 3. Actual content depends on whether it is a Read or Write operation from MCS.** DATA: Data exchange buffer 3. Actual content depends on whether it is a Read or Write operation from MCS.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.15.5Register MCS2DPLL_DEB4

MCS to DPLL Data Exchange Buffer 4

READ access from MCS: SYN_S_OLD (**Equation DPLL-6a4 to update the time stamp values for STATE**). Use to compute/update the time stamp values for STATE during the update of ram (STA_S = 0b0000_1001)WRITE access from MCS: The DPLL expects RDT_S[p-q] (**Equations DPLL-10 to calculate the current increment (nominal value)**) or RDT_S[p+q](**Equations DPLL-10 to calculate the current increment value**) during the increment

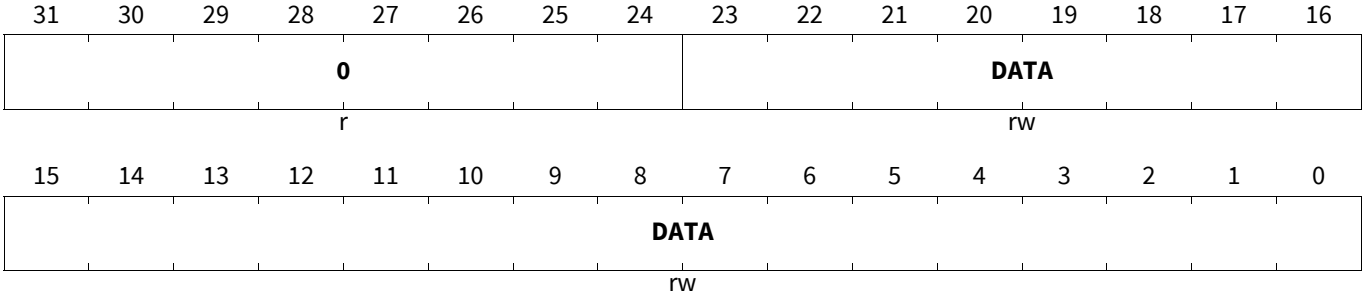
prediction (STA_S = 0b0001_0000) and RDT_S[t-q] (**Action calculations for STATE forwards**) or RDT_S[t+q](**Action calculations for STATE backwards**) during the action calculation (STA_S = 0b0111_0000)

Note: The data read from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

MCS2DPLL_DEB4

MCS to DPLL Data Exchange Buffer 4 (007810$_H$) PowerOn Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 4. Actual content depends on whether it is a Read or Write operation from MCS.** DATA: Data exchange buffer 4. Actual content depends on whether it is a Read or Write operation from MCS.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.15.6Register MCS2DPLL_DEB5

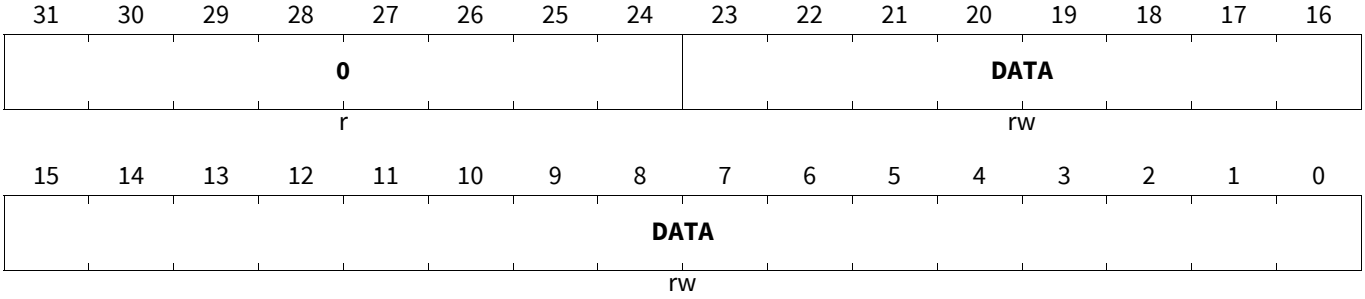
MCS to DPLL Data Exchange Buffer 5

READ access from MCS: M_DW (m in **Action calculations for STATE forwards** and **Action calculations for STATE backwards**). Use to provide the proper Time Stamp Field value during the action calculation (STA_S = 0b0111_0000)WRITE access from MCS: The DPLL expects DT_S[p-q+1] (**Equations DPLL-10 to calculate the current increment (nominal value)**) or DT_S[p+q-1](**Equations DPLL-10 to calculate the current increment value**) during the increment prediction (STA_S = 0b0001_0000) and DT_S[t-q+1] (**Action calculations for STATE forwards**) or DT_S[t+q-1](**Action calculations for STATE backwards**) during the action calculation (STA_S = 0b0111_0000)

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

MCS2DPLL_DEB5

MCS to DPLL Data Exchange Buffer 5 (007814_H) PowerOn Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 5. Actual content depends on whether it is a Read or Write operation from MCS.** DATA: Data exchange buffer 5. Actual content depends on whether it is a Read or Write operation from MCS.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.15.7Register MCS2DPLL_DEB6

MCS to DPLL Data Exchange Buffer 6

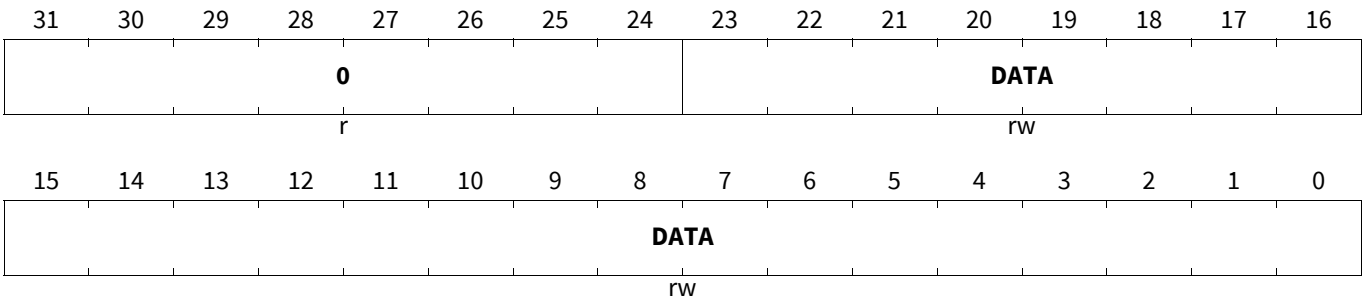
READ access from MCS: Reads as 0.WRITE access from MCS: The DPLL expects ADT_S[APS_1C2] during the update of RAM (STA_S = 0b0000_1001) and change of direction (STA_S = 0b0000_0100 and STA_S = 0b0000_0110)

Note: In both cases, the current ADT_S[APS_1C2] value should be stored in the register before unlocking the state machine the second time, i.e.: between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

Note: The format of ADT_S should match the defined in **ADDR**

MCS2DPLL_DEB6

MCS to DPLL Data Exchange Buffer 6 (007818_H) PowerOn Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 6.** DATA: Data exchange buffer 6.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

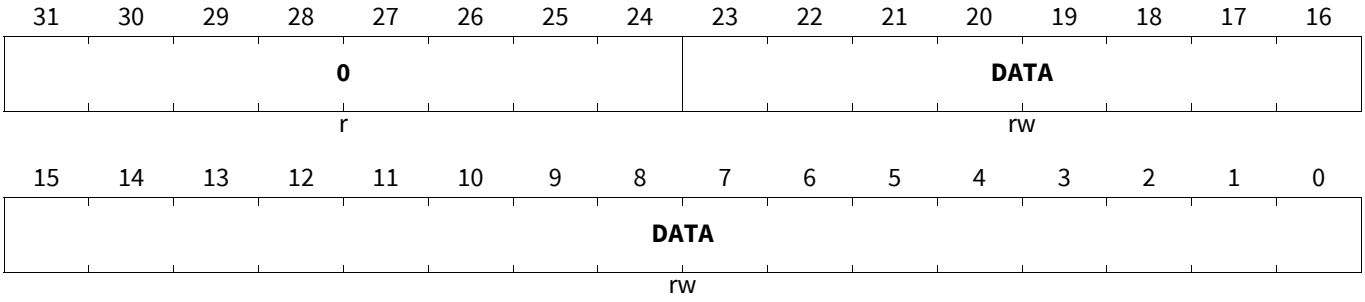
28.20.15.8Register MCS2DPLL_DEB7

MCS to DPLL Data Exchange Buffer 7

READ access from MCS: Duration of the reciprocal of the last increment RDT_S_ACT (**Update of RAM in Normal and Emergency Mode**). This value is written by the DPLL during the update of ram (STA_S = 0b0000_1001) and is ready to be read when STA_S is modified to 0b0000_1010.WRITE access from MCS: The DPLL expects the reciprocal of the last increment RDT_S[APS] (**Update of RAM in Normal and Emergency Mode**) before it is overwritten with RDT_S_ACT during the update of ram (STA_S = 0b0000_1001). For the action calculation, this value is needed as well as RDT_S[t] in **Action calculations for STATE forwards** and **Action calculations for STATE backwards** .

Note: During an update of ram, perform the write before unlocking the state machine (STA_S = 0b0000_1001), i.e.: after the first write to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO) but before the second write. During the action calculation, the data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

MCS2DPLL_DEB7
MCS to DPLL Data Exchange Buffer 7 (00781C$_H$) PowerOn Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 7. Actual content depends on whether it is a Read or Write operation from MCS.** DATA: Data exchange buffer 7. Actual content depends on whether it is a Read or Write operation from MCS.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.15.9Register MCS2DPLL_DEB8

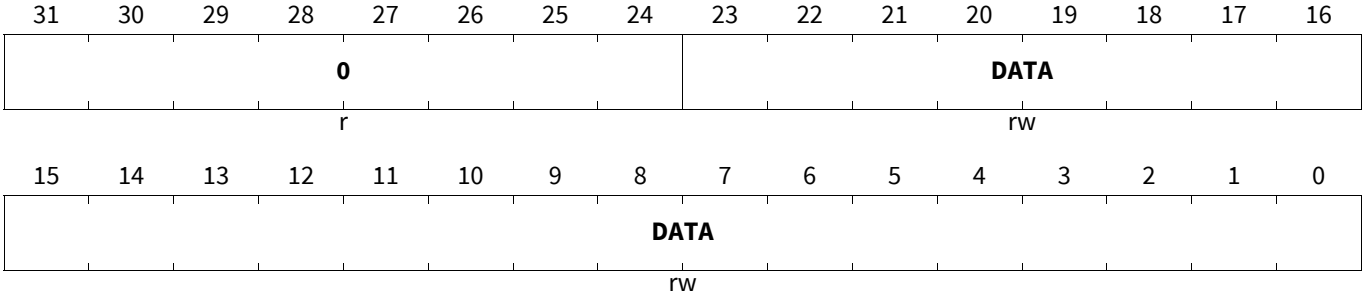
MCS to DPLL Data Exchange Buffer 8

READ access from MCS: Reads as 0.WRITE access from MCS: The DPLL expects TSF_S[p] (**Action calculations for STATE forwards** and **Action calculations for STATE backwards**) during the action calculation (STA_S = 0b0111_0000)

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

MCS2DPLL_DEB8

MCS to DPLL Data Exchange Buffer 8 (007820_H) PowerOn Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 8.** DATA: Data exchange buffer 8.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.15.10Register MCS2DPLL_DEB9

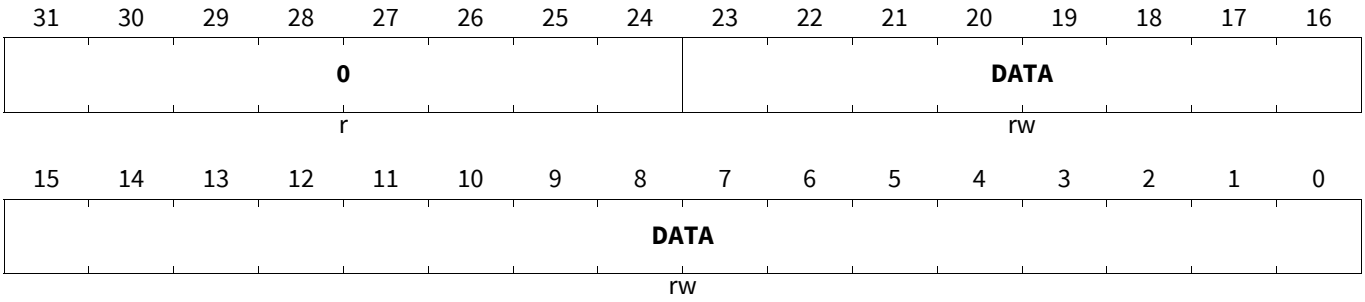
MCS to DPLL Data Exchange Buffer 9

READ access from MCS: Reads as 0.WRITE access from MCS: The DPLL expects TSF_S[p-n] (**Action calculations for STATE forwards**) or TSF_S[p+n](**Action calculations for STATE backwards**) during the action calculation (STA_S = 0b0111_0000)

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

MCS2DPLL_DEB9

MCS to DPLL Data Exchange Buffer 9 (007824_H) PowerOn Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 9.** DATA: Data exchange buffer 9.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

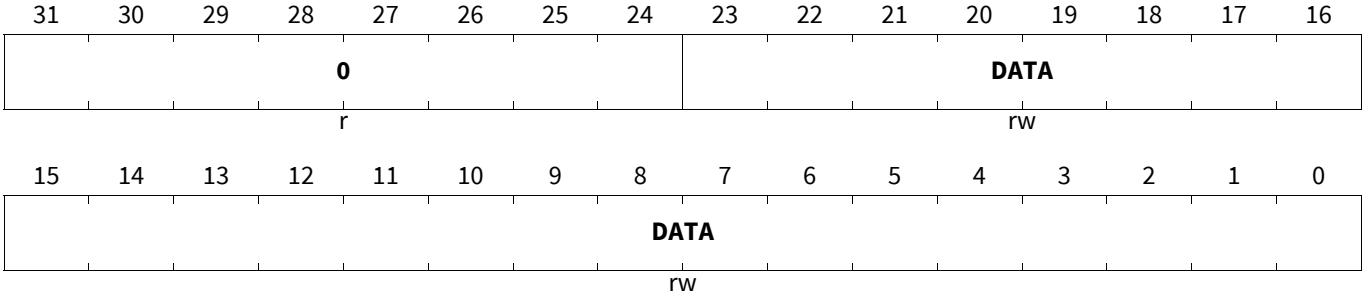
28.20.15.11Register MCS2DPLL_DEB10

MCS to DPLL Data Exchange Buffer 10

READ access from MCS: Reads as 0.WRITE access from MCS: The DPLL expects TSF_S[p+m-n] (**Action calculations for STATE forwards**) or TSF_S[p-m+n](**Action calculations for STATE backwards**) during the action calculation (STA_S = 0b0111_0000)

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

MCS2DPLL_DEB10
MCS to DPLL Data Exchange Buffer 10 (007828_H) PowerOn Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 10.** DATA: Data exchange buffer 10.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

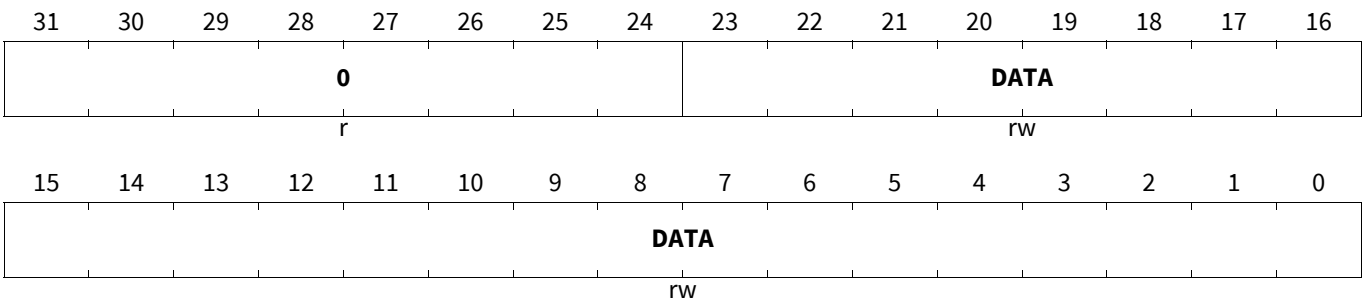
28.20.15.12Register MCS2DPLL_DEB11

MCS to DPLL Data Exchange Buffer 11

READ access from MCS: Reads as 0.WRITE access from MCS: The DPLL expects TSF_S[p+m] (**Action calculations for STATE forwards**) or TSF_S[p-m](**Action calculations for STATE backwards**) during the action calculation (STA_S = 0b0111_0000)

Note: The data write from MCS should be performed between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO)

MCS2DPLL_DEB11
MCS to DPLL Data Exchange Buffer 11 ($00782C_H$) PowerOn Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 11.** DATA: Data exchange buffer 11.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.20.15.13Register MCS2DPLL_DEB12

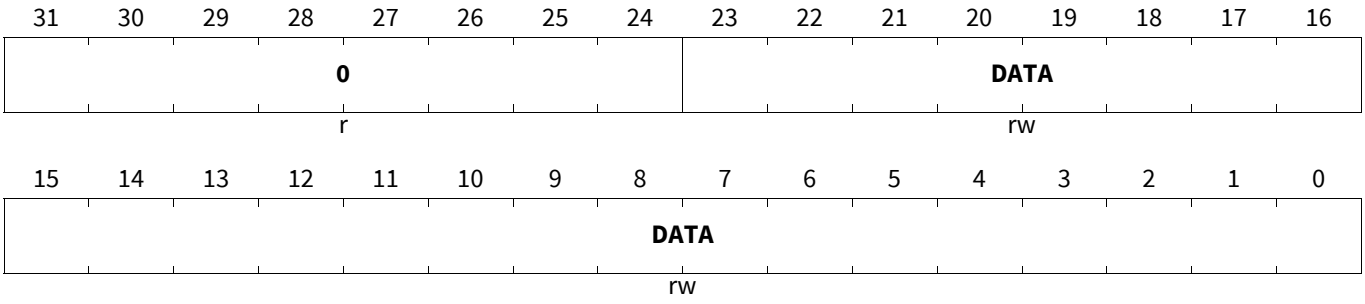
MCS to DPLL Data Exchange Buffer 12

READ access from MCS: Reads as 0.WRITE access from MCS: The DPLL expects the future adapt information ADT_S[APS_1C2+1] (when in forwards) or ADT_S[APS_1C2-1] (when in backwards) (**Equation DPLL-8 to calculate the error of last prediction**) during the update of RAM (STA_S = 0b0000_1001) and change of direction (STA_S = 0b0000_0100 and STA_S = 0b0000_0110)

Note: In both cases, the current ADT_S[APS_1C2+1] or ADT_S[APS_1C2-1] value should be stored in the register before unlocking the state machine the second time, i.e.: between the two writes to MCS2DPLL_DEB15 (MCS2DPLL_STATUS_INFO).

Note: The format of ADT_S should match the defined in **ADDR**

MCS2DPLL_DEB12
MCS to DPLL Data Exchange Buffer 12 **(007830_H)** **PowerOn Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 12.** DATA: Data exchange buffer 12.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

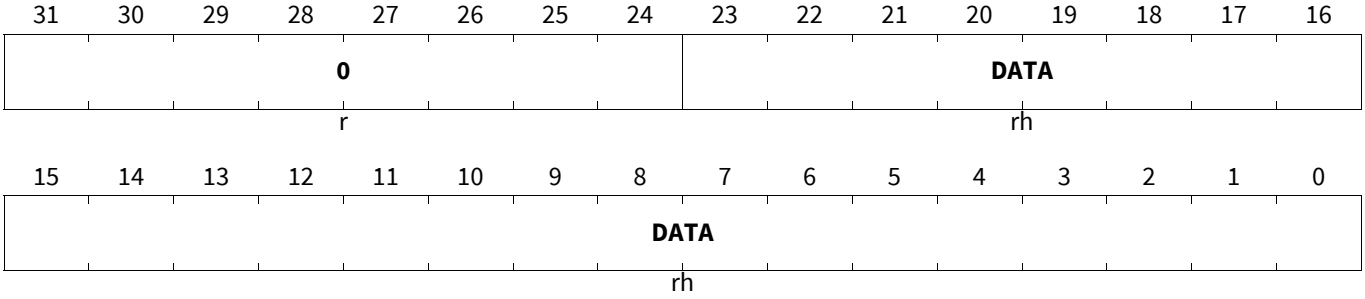
28.20.15.14Register MCS2DPLL_DEB13

MCS to DPLL Data Exchange Buffer 13

READ access from MCS: Reads as 0.WRITE access from MCS: Ignored during DPLL processing.

MCS2DPLL_DEB13

MCS to DPLL Data Exchange Buffer 13 (007834_H) PowerOn Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
DATA	23:0	rh	**Data exchange buffer 13.** DATA: Data exchange buffer 13.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

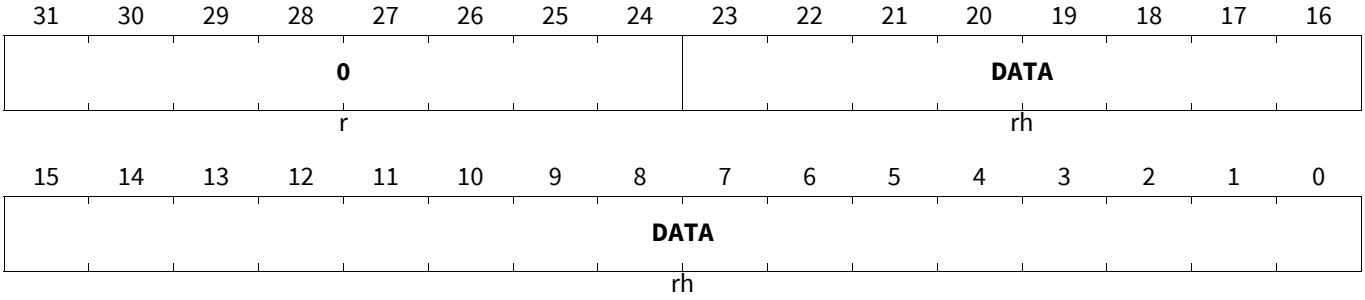
28.20.15.15Register MCS2DPLL_DEB14

MCS to DPLL Data Exchange Buffer 14

READ access from MCS: Reads as 0.WRITE access from MCS: Ignored during DPLL processing.

MCS2DPLL_DEB14

MCS to DPLL Data Exchange Buffer 14 (007838_H) PowerOn Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
DATA	23:0	rh	**Data exchange buffer 14.** DATA: Data exchange buffer 14.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

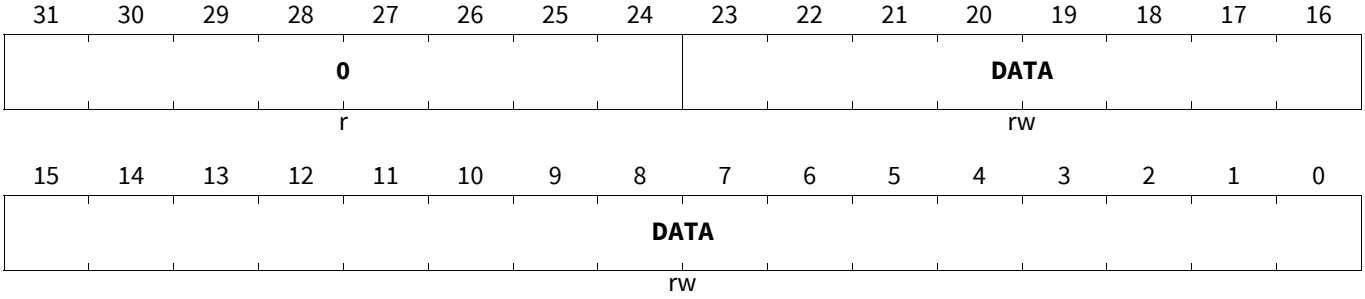
28.20.15.16Register MCS2DPLL_DEB15

MCS to DPLL Data Exchange Buffer 15

READ access from MCS: Reads as 0.WRITE access from MCS: Unlocks the DPLL STATE state machine. See **General functionality** .

MCS2DPLL_DEB15

MCS to DPLL Data Exchange Buffer 15 **(00783C$_H$)** **PowerOn Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
DATA	23:0	rw	**Data exchange buffer 15.** DATA: Data exchange buffer 15.
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.21 Sensor Pattern Evaluation (SPE)

28.21.1 Overview

The Sensor Pattern Evaluation (SPE) submodule can be used to evaluate three hall sensor inputs and together with the TOM module to support the drive of BLDC engines. Thus, the input signals are filtered already in the connected TIM channels. In addition, the SPE submodule can be used as an input stage to the MAP submodule if the DPLL should be used to calculate the rotation speed of one or two electric engine(s). The integration of the SPE submodule into the overall GTM architecture concept is shown in **Figure 132**.

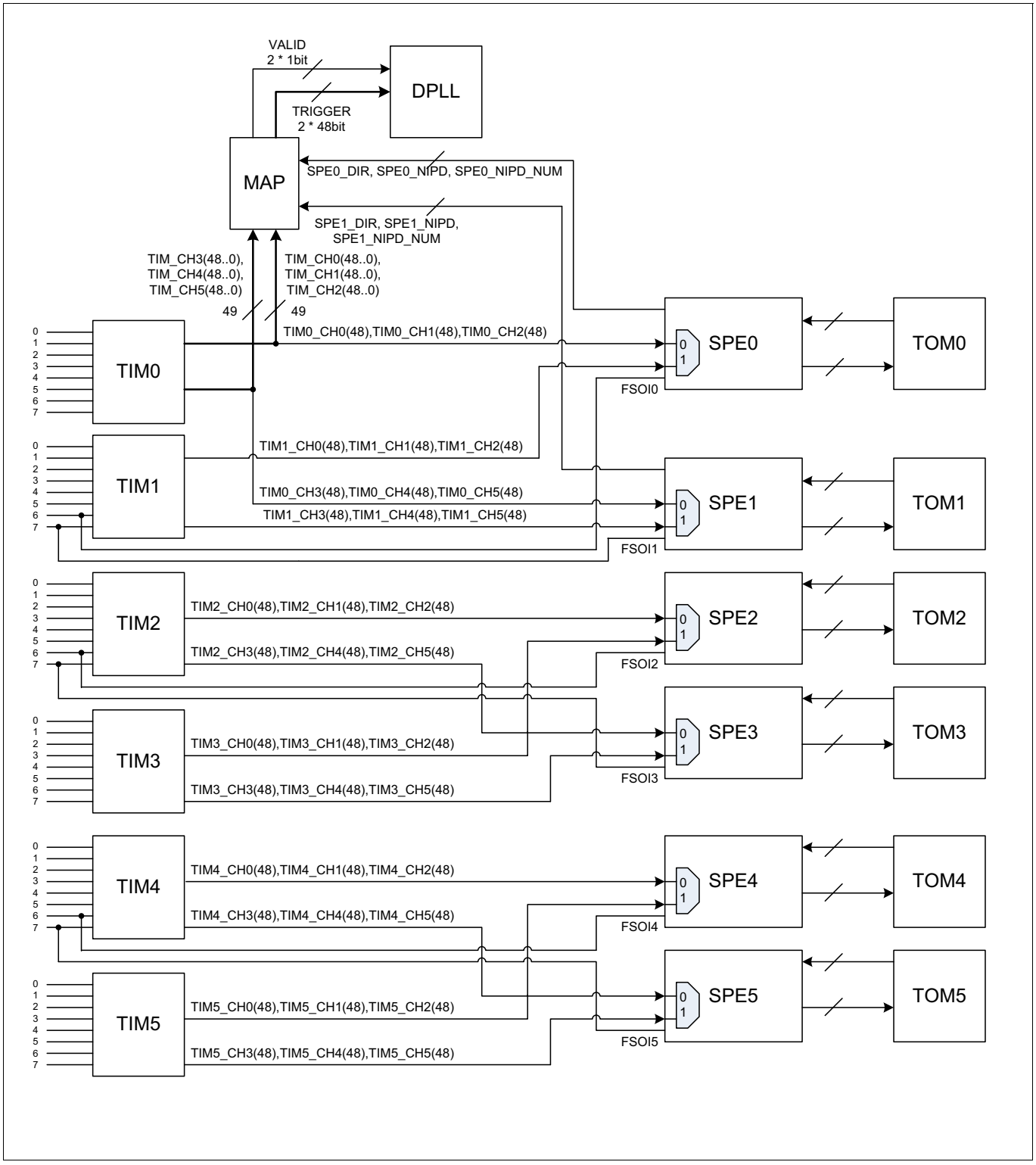


Figure 132 SPE Submodule integration concept into GTM

The SPE submodule can determine a rotation direction out of the combined *TIM[i]_CHx(48)*, *TIM[i]_CHy(48)* and *TIM[i]_CHz(48)* signals. On this input signals a pattern match algorithm is applied to generate the *SPEx_DIR* signal on behalf of the temporal relation between these input patterns. A possible sample pattern of the three input signals is shown in **Figure 133**. In general, the input pattern is programmable within the SPE submodule.

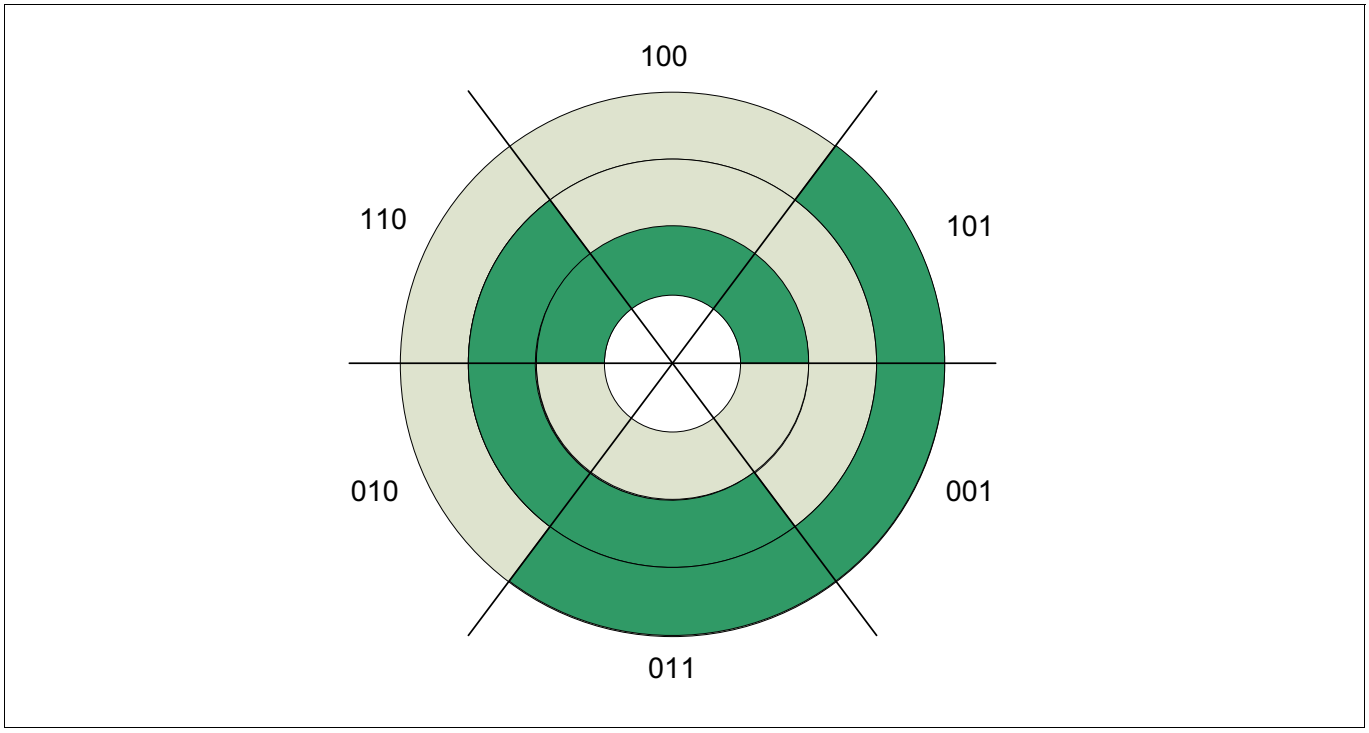


Figure 133 SPE Sample input pattern for TIM[i]_CH[x,y,z](48)

In **Figure 133** the input signals define the pattern from the input sensors which have a 50% high and 50% low phase. The pattern according to **Figure 133** is as follows:

100 – 110 – 010 – 011 – 001 – 101 – 100

where the first bit (smallest circle) represents *TIM[i]_CH[x](48)*, the second bit represents *TIM[i]_CH[y](48)*, and the third bit (greatest circle) represents *TIM[i]_CH[z](48)*.

Note that the SPE module expects that with every new pattern only one of the three input signals changes its value.

28.21.2 SPE Submodule description

The SPE submodule can handle sensor pattern inputs. Every time if one of the input signals *TIM[i]_CH[x](48)*,*TIM[i]_CH[y](48) or TIM[i]_CH[z](48)* changes its value, a sample of all three input signals is made. Derived from the sample of the three inputs the encoded rotation direction and the validity of the input pattern sequence is determined and signaled. When a valid input pattern is detected, the SPE submodule can control the outputs of a dedicated connected TOM submodule. This connection is shown in **Figure 134**.

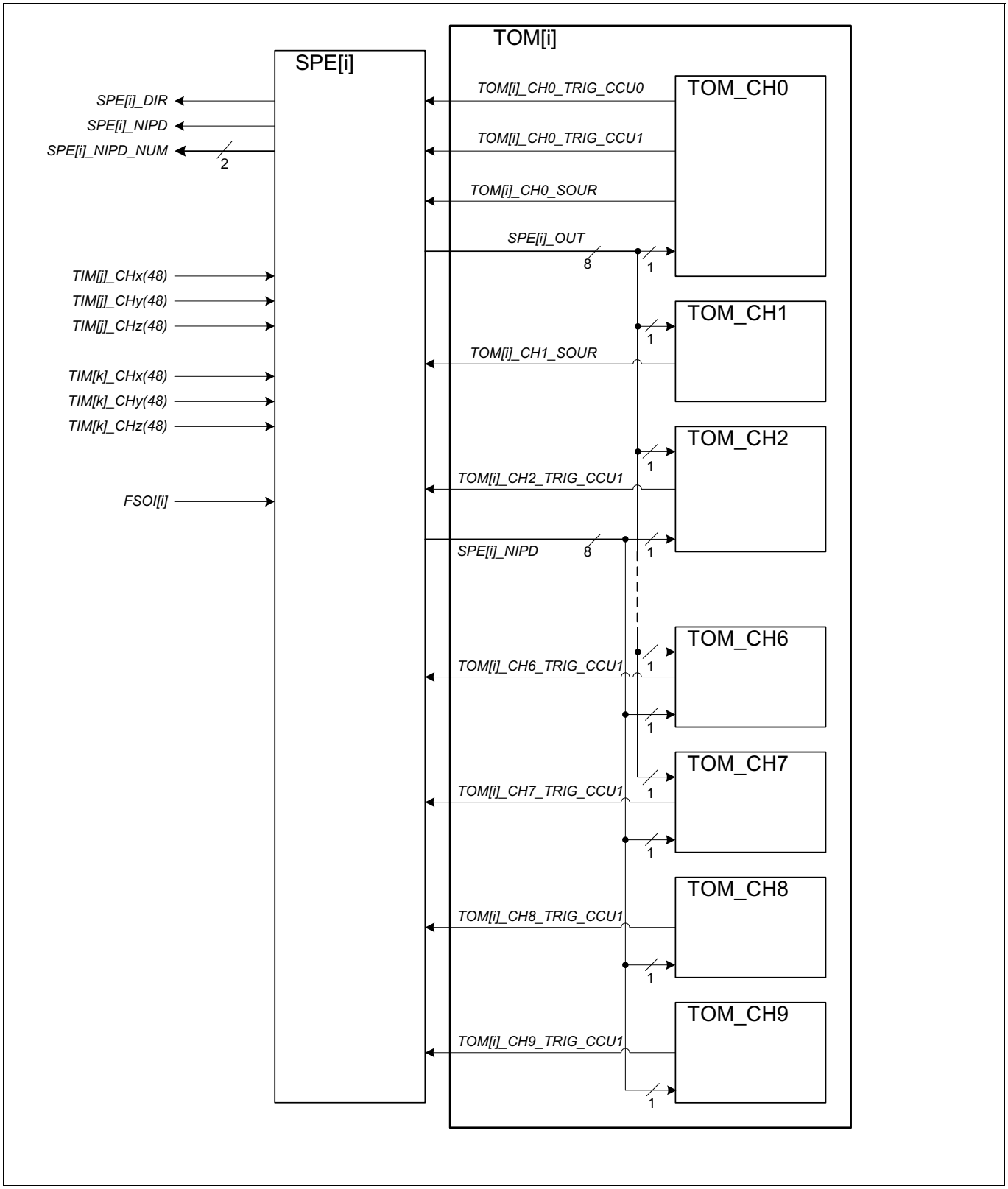


Figure 134 SPE to TOM Connections

The *TOM[i]_CH0_TRIG_CCU[x]* and *TOM[i]_CH[x]_SOUR* signal lines are used to evaluate the current state of the TOM outputs, whereas the *SPE[i]_OUT* output vector is used to control the TOM output depending on the new input pattern. The *SPE[i]_OUT* output vector is defined inside the SPE submodule in a pattern definition table **SPE[i]_OUT_PAT[x]**. The internal SPE submodule architecture is shown in **Figure 135**.

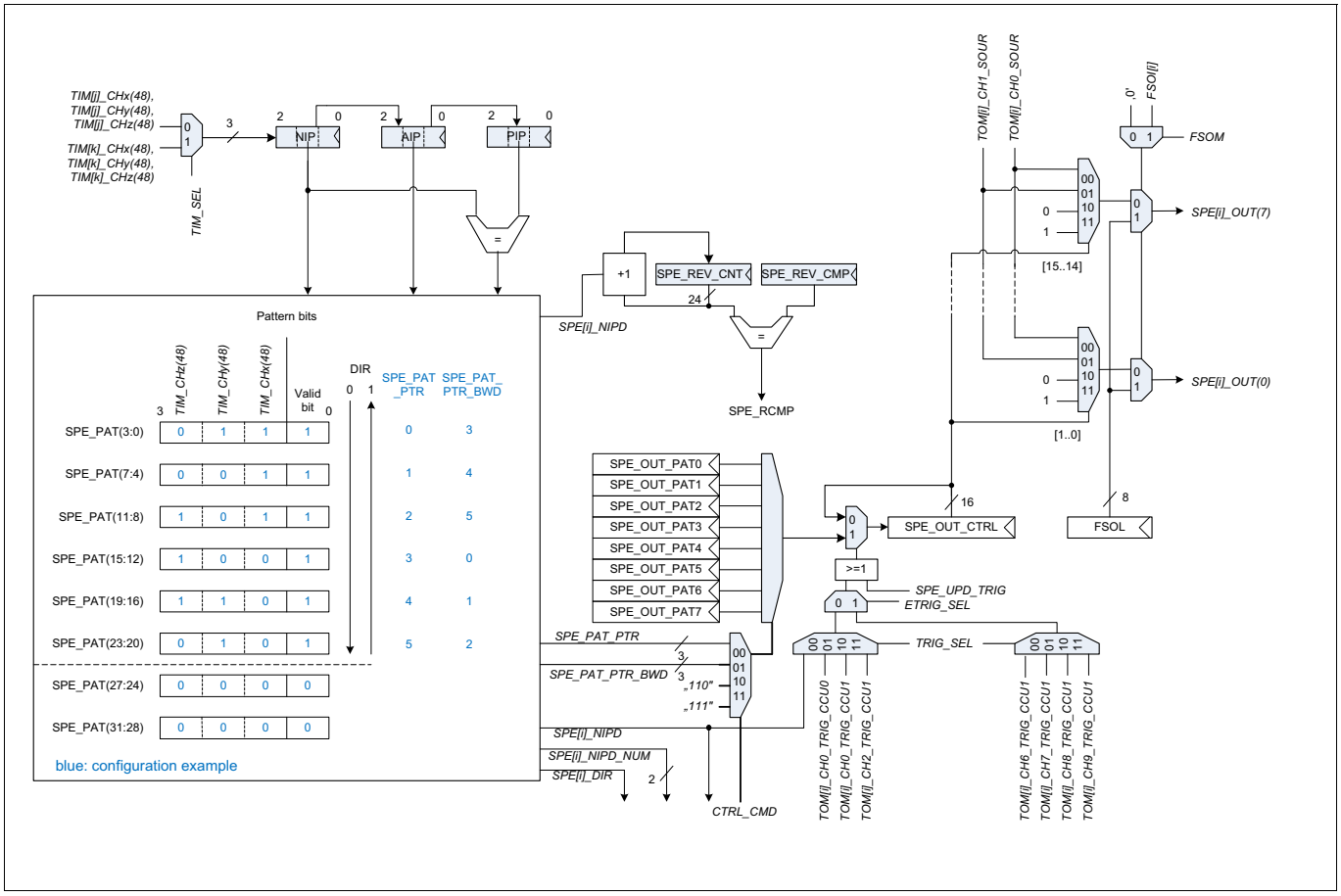


Figure 135 SPE Submodule architecture

The **SPE[i]_PAT** register holds the valid input pattern for the three input patterns *TIM[i]_CH[x](48)*, *TIM[i]_CH[y](48)* and *TIM[i]_CH[z](48)*. The input pattern is programmable. The valid bit shows if the programmed pattern is a valid one. **Figure 135** shows the programming of the **SPE[i]_PAT** register for the input pattern defined in **Figure 133**.

The rotation direction is determined by the order of the valid input pattern. This rotation direction defines if the *SPE_PAT_PTR* is incremented (DIR = 0) or decremented (DIR = 1). Whenever a valid input pattern is detected, the *NIPD* signal is raised, the *SPE_PAT_PTR* is incremented/decremented and a new output control signal *SPE[i]_OUT(x)* is send to the corresponding TOM submodule.

To command directly the forward or backward rotation the SPE provides with **SPE[i]_APT_PTR** and **SPE[i]_PAT_PTR_BWD** two pointers to array **SPE[i]_OUT_PAT[z].** Both can point to different values of **SPE[i]_OUT_PAT[z]** at the same point in time. **SPE[i]_APT_PTR** is intended to point to the pattern for forward commanding and **SPE[i]_PAT_PTR_BWD** is intended to point to the pattern for backward commanding.On startup both pointers have to be configured to an initial value that corresponds to different direction depending start pattern of **SPE[i]_OUT_PAT[z]**.With each valid new input pattern indicated by *SPE_NIPD* both pointers will be incremented or decremented according to the detected direction. Switching from command forward to command backward can then be done by changing the selected pointer to **SPE[i]_OUT_PAT[z]** array, i.e. changing **SPE_CTRL_CMD** in register **SPE[i]_CMD** from selecting **SPE[i]_PAT_PTR** to selecting **SPE[i]_PAT_PTR_BWD** or vice versa. The intended behavior is depicted in the following figure.

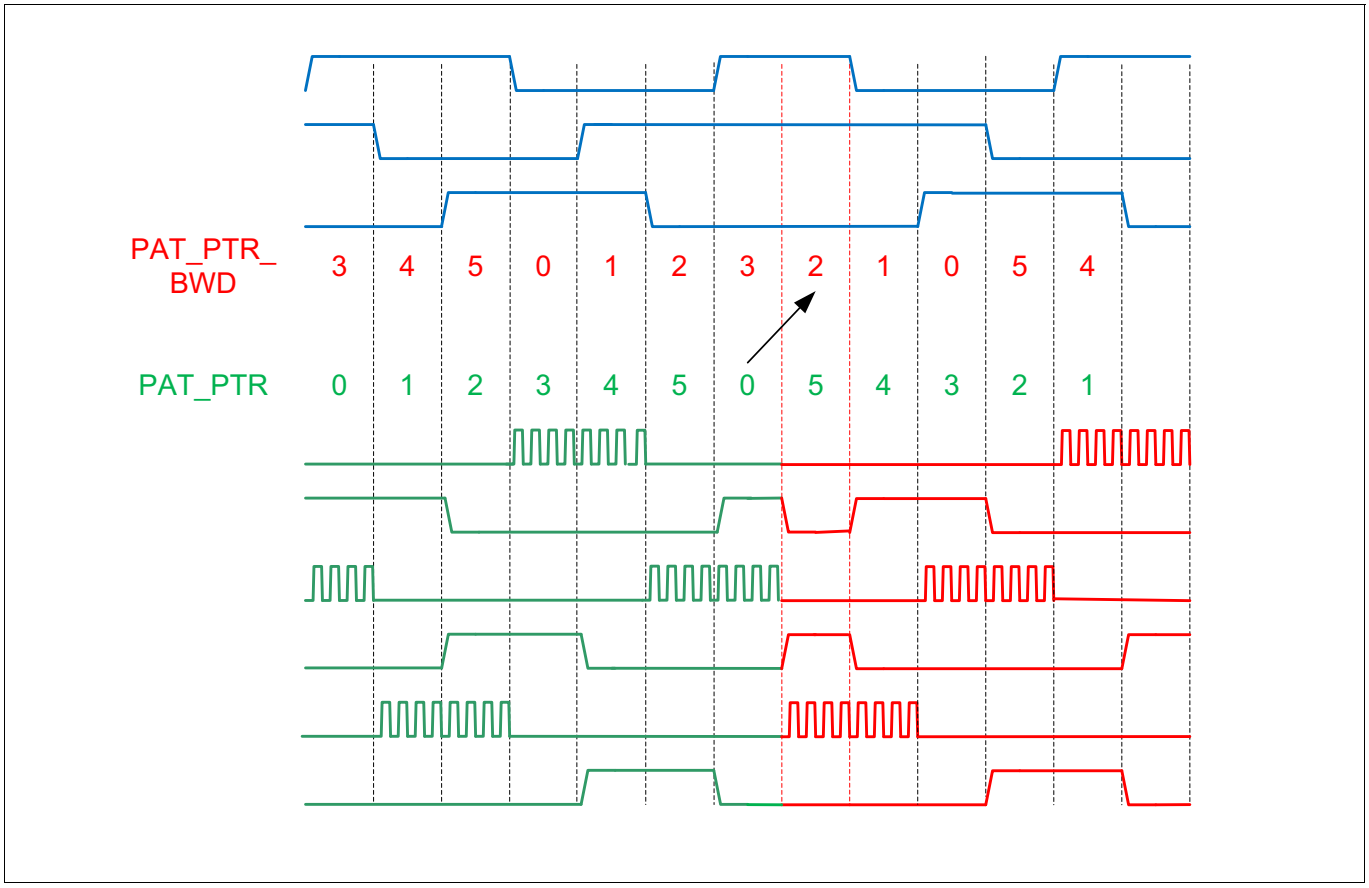


Figure 136 SPE forward - backward commanding

With command **SPE_CTRL_CMD** = 0b10 or 0b11 a dedicated configurable output pattern configured to the pattern **SPE_OUT_PAT6** or **SPE_OUT_PAT7** can be commanded to the outputs.

An example is the introduction of a SW dead time if switching from pointer **SPE[i]_PAT_PTR** to **SPE[i]_PAT_PTR_BWD** or vice versa. E.g. if in **SPE[i]_OUT_PAT6** the value 0b10 for each output (i.e. set *SPE_OUT(n)* to 0) is programmed, this can be used as an intermediate step to introduce this 'all off' when switching between **SPE[i]_PAT_PTR** to **SPE[i]_PAT_PTR_BWD** or vice versa.

Selectable by **TRIG_SEL** and **ETRIG_SEL** the CCU1 trigger of either the TOM channel 2,6,7,8 and 9 can be used together with the SPE module to trigger a delayed update of the **SPE_OUT_CTRL** register after new input pattern detected by SPE (signaled by *SPE[i]_NIPD*).

To do this, the TOM channel z=2,6,7,8 or 9 has to be configured to work in one-shot mode (set bit **OSM** in register **TOM[i]_CH[z]_CTRL**). The SPE trigger of this channel has to be enabled, too (set description of bit **SPEM** and bit **SPE_TRIG** in register **TOM[i]_CH[z]_CTRL**). The SPE module has to be configured to update **SPE_OUT_CTRL** on *TOM[i]_CH[z]_TRIG_CCU1* (set in **SPE[i]_CTRL_STAT** bits **TRIG_SEL** to 0b11). Then, on new input detected by SPE, the signal *SPE[i]_NIPD* triggers the start of the TOM channel z to generates one PWM period by resetting **CN0** to 0. On second PWM edge triggered by CCU1 of TOM channel z, the signal *TOM[i]_CH[z]_TRIG_CCU1* triggers the update of **SPE_OUT_CTRL**.

The update of **SPE[i]_OUT_CTRL** with the content of one of the **SPE_OUT_PAT[z]** register can be triggered at any time by writing a 1 to bit **SPE_UPD_TRIG** in register **SPE[i]_CMD**.

The regular trigger for update of **SPE[i]_OUT_CTRL** (commutation trigger) is selected by **TRIG_SEL** and **ETRIG_SEL**.

According to **Figure 135**, the two input patterns 0b000 and 0b111 are not allowed combinations and will end in a *SPE[i]_PERR* interrupt. These two patterns can be used to determine a sensor input error. A *SPE[i]_PERR*

interrupt will also be raised, if the input patterns occur in a wrong order, e.g. if the pattern 0b010 does not follow the pattern 0b110 or 0b011.

The register **SPE[i]_IN_PAT** bit field inside the **SPE[i]_CTRL_STAT** register is implemented, where the input pattern history is stored by the SPE submodule. The CPU can determine a broken sensor when the SPE[i]_PERR interrupt occurs by analyzing the bit pattern readable via bit field **NIP** inside the **SPE[i]_CTRL_STAT** register. The input pattern in the **SPE[i]_CTRL_STAT** register is updated whenever a valid edge is detected on one of the input lines *TIM[i]_CH[x](48)*, *TIM[i]_CH[y](48)* or *TIM[i]_CH[z](48)*. The pattern bit fields are then shifted. The input pattern history generation inside the **SPE[i]_CTRL_STAT** register is shown in **Figure 137**.

Additionally to the sensor pattern evaluation the SPE module also provides the feature of fast shutoff for all TOM channels controlled by the SPE module. The feature is enabled by setting bit **FSOM** in register **SPE[i]_CTRL_STAT**. The fast shutoff level itself is defined in the bit field **FSOL** of register **SPE[i]_CTRL_STAT**. The TIM input used to trigger the fast shutoff is either TIM channel 6 or TIM channel 7 depending on the TIM instance connected to the SPE module. For details of connections please refer to **Figure 132**.

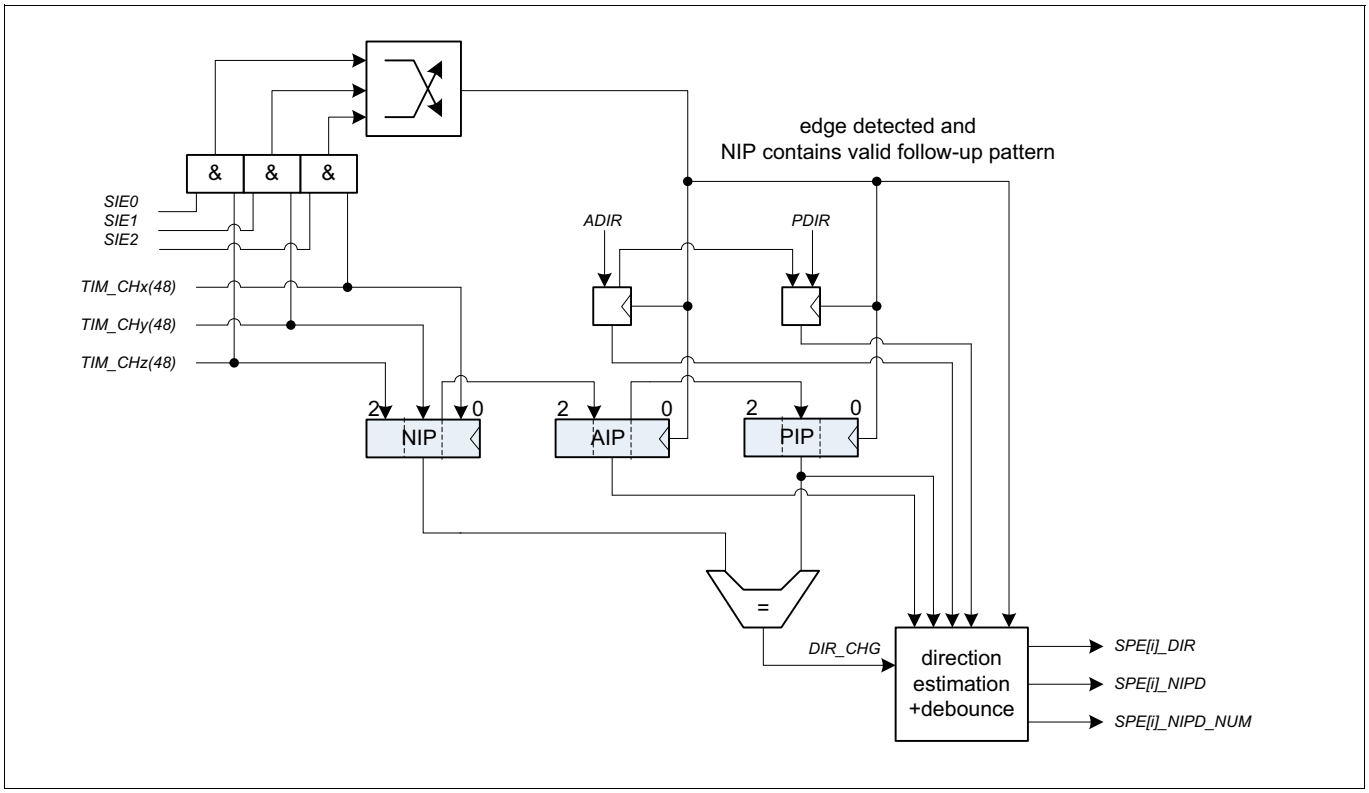


Figure 137 SPE[i]_IN_PAT register representation

The CPU can disable one of the three input signals, e.g. when a broken input sensor was detected, by disabling the input with the three input enable bits **SIE** inside the **SPE[i]_CTRL_STAT** register.

Whenever at least one of the input signal *TIM[i]_CH[x](48)*, *TIM[i]_CH[y](48)* or *TIM[i]_CH[z](48)* changes the SPE submodule stores the new bit pattern in an internal register NIP (New Input Pattern). If the current input pattern in **NIP** is the same as in the Previous Input Pattern (**PIP**) the direction of the engine changed, the *SPE[i]_DCHG* interrupt is raised, the direction change is stored internally and the pattern in the **PIP** bit field is filled with the **AIP** bit field and the **AIP** bit field is filled with the **NIP** bit field. The **SPE[i]_ADIR** bit inside the **SPE[i]_CTRL_STAT** register is toggled and the *SPE[i]_DIR* signal is changed.

If the SPE encounters that with the next input pattern detected new input pattern **NIP** the direction change again, the input signal is categorized as bouncing and the bouncing input signal interrupt *SPE[i]_BIS* is raised.

Immediately after update of register **NIP**, when the new detected input pattern doesn't match the **PIP** pattern (i.e. no direction change was detected), the SPE shifts the value of register **AIP** to register **PIP** and the value of register **NIP** to register **AIP**. The *SPE[i]_NIPD* interrupt is raised.

The number of the channel that has been changed and thus leads to the new input pattern is encoded in the signal *SPE[i]_NIPD_NUM*.

If a sensor error was detected, the CPU has to define upon the pattern in the **SPE[i]_CTRL_STAT** register, which input line comes from the broken sensor. The faulty signal line has to be masked by the CPU and the SPE submodule determines the rotation direction on behalf of the two remaining *TIM[i]_CH[x]* input lines.

The pattern history can be determined by the CPU by reading the two bit fields AIP and PIP of the **SPE[i]_CTRL_STAT** register. The **AIP** register field holds the actual detected input pattern at *TIM[i]_CH[x](48)*, *TIM[i]_CH[y](48)* and *TIM[i]_CH[z](48)* and the PIP holds the previous detected pattern.

After reset the register **NIP**, **AIP** and **PIP** as well as the register **SPE[i]_PAT_PTR** and **SPE[i]_OUT_CTRL** will not contain valid startup values which would allow correct behavior after enabling SPE and detecting the first input patterns.Thus, it is necessary to initialize these register to correct values.To do this, before enabling the SPE, the bit field **NIP** of register **SPE[i]_CTRL_STAT** can be read and depending on this value the initialization values for the register **AIP**, **PIP**, **SPT_PAT_PTR** and **SPE[i]_OUT_CTRL** can be determined.

28.21.2.1 SPE Revolution detection

The SPE submodule is able to detect and count the number of valid input patterns detected at the specified input ports. This is done with a 24 bit revolution counter **SPE_REV_CNT**. The counter is incremented by a value of one (1) when a new valid input pattern indicating forward direction is detected. The counter is decremented by a value of one (1) when a new valid input pattern indicating backward direction is detected.

In addition there exists a 24 bit **SPE_REV_CMP** register. The user can initialize this register with a compare value, where an interrupt *SPE[i]_RCMP* is raised, when the revolution counter equals the compare value either in forward or backward direction.

Both register may be written by software at any time.

28.21.3 SPE Interrupt signals

Table 145 SPE Interrupt signals

Signal	Description
SPE[i]_NIPD	SPE New valid input pattern detected.
SPE[i]_DCHG	SPE Rotation direction change detected on behalf of input pattern.
SPE[i]_PERR	SPE Invalid input pattern detected.
SPE[i]_BIS	SPE Bouncing input signal detected at input.
SPE[i]_RCMP	SPE Revolution counter compare value reached.

28.21.4 SPE Register overview

Table 146 SPE Register overview

Register name	Description	see Page
SPE[i]_CTRL_STAT	SPEi Control status register	670
SPE[i]_PAT	SPEi Input pattern definition register.	671
SPE[i]_OUT_PAT[z]	SPEi Output definition register.	672
SPE[i]_OUT_CTRL	SPEi output control register	673
SPE[i]_REV_CNT	SPEi input revolution counter	674
SPE[i]_REV_CMP	SPEi Revolution counter compare value	674
SPE[i]_IRQ_NOTIFY	SPEi Interrupt notification register.	675
SPE[i]_IRQ_EN	SPEi Interrupt enable register.	676
SPE[i]_EIRQ_EN	SPEi Error interrupt enable register.	678
SPE[i]_IRQ_FORCINT	SPEi Interrupt generation by software.	677
SPE[i]_IRQ_MODE	SPEi Interrupt mode configuration register	678
SPE[i]_CTRL_STAT2	SPEi Control status register 2	679
SPE[i]_CMD	SPEi Command register	680

28.21.5 SPE Register description

28.21.5.1 Register SPE[i]_CTRL_STAT

SPEi Control Status Register

SPEi_CTRL_STAT (i=0-5)
SPEi Control Status Register **(000800_H+i*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FSOL								ETRIG_SEL	NIP			PDIR	PIP		
rw								rw	r			rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADIR	AIP			0	SPE_PAT_PTR			FSOM	TIM_SEL	TRIG_SEL		SIE2	SIE1	SIE0	EN
rw	rw			r	rw			rw	rw	rw		rw	rw	rw	rw

Field	Bits	Type	Description
EN	0	rw	**SPE Submodule enable** 0_B SPE disabled 1_B SPE enabled
SIE0	1	rw	**SPE Input enable for TIM_CHx(48)** When the input is disabled, a 0 signal is sampled for this input. However, the bit field NIP of this register shows the true value of the input signal. 0_B SPE Input is disabled 1_B SPE Input is enabled
SIE1	2	rw	**SPE Input enable for TIM_CHy(48)** Coding see bit 1.
SIE2	3	rw	**SPE Input enable for TIM_CHz(48)** Coding see bit 1.
TRIG_SEL	5:4	rw	**Select trigger input signal** ETRIG_SEL = 0 / ETRIG_SEL = 1: *Note: In case of ETRIG_SEL=1, according to selected TOM_CH[x]_TRIG_CCU1 signal the configuration bits SPE_TRIG and OSM of register TOM_CH[x]_CTRL have to be set in same TOM channel x to enable the trigger signal generation in one-shot mode.* 00_B SPE[i]_NIPD selected / TOM_CH6_TRIG_CCU1 selected 01_B TOM_CH0_TRIG_CCU0 selected / TOM_CH7_TRIG_CCU1 selected 10_B TOM_CH0_TRIG_CCU1 selected / TOM_CH8_TRIG_CCU1 selected 11_B TOM_CH2_TRIG_CCU1 selected / TOM_CH9_TRIG_CCU1 selected

Field	Bits	Type	Description
TIM_SEL	6	rw	**Select TIM input signal** The GTM supports up to 6 SPE modules.SPE0: 0_B SPE0: TIM0_CH0..2 / SPE1: TIM0_CH3..5 / SPE2: TIM2_CH0..2 / SPE3: TIM2_CH3..5 / SPE4: TIM4_CH0..2 / SPE5: TIM4_CH3..5 1_B SPE0: TIM1_CH0..2 / SPE1: TIM1_CH3..5 / SPE2: TIM3_CH0..2 / SPE3: TIM3_CH3..5 / SPE4: TIM5_CH0..2 / SPE5: TIM5_CH3..5
FSOM	7	rw	**Fast Shutoff Mode** 0_B Fast Shutoff mode disabled 1_B Fast Shutoff mode enabled
SPE_PAT_PTR	10:8	rw	**Pattern selector for TOM output signals** Actual index into the SPE[i]_OUT_PAT[x] register table. Each register SPE[i]_OUT_PAT[x] is fixed assigned to one bit field IPx_PAT of register SPE[i]_PAT. Thus, the pointer SPE[i]_PAT_PTR represents an index to the selected SPE[i]_OUT_PAT[x] register as well as the actual detected input pattern IPx_PAT. 000_B SPE[i]_OUT_PAT0 selected
AIP	14:12	rw	**Actual input pattern that was detected by a regular input pattern change**
ADIR	15	rw	**Actual rotation direction** 0_B Rotation direction is 0 according to SPE[i]_PAT register 1_B Rotation direction is 1 according to SPE[i]_PAT register
PIP	18:16	rw	**Previous input pattern that was detected by a regular input pattern change**
PDIR	19	rw	**Previous rotation direction** 0_B Rotation direction is 0 according to SPE[i]_PAT register 1_B Rotation direction is 1 according to SPE[i]_PAT register
NIP	22:20	r	**New input pattern that was detected** This bit field mirrors the new input pattern. SPE internal functionality is triggered on each change of this bit field.
ETRIG_SEL	23	rw	**Extended TRIG_SEL** Extended trigger selection. Details described in TRIG_SEL bit field. ETRIG_SEL = 0 / ETRIG_SEL = 1: 00_B = SPE[i]_NIPD selected / TOM_CH6_TRIG_CCU1 selected 01_B = TOM_CH0_TRIG_CCU0 selected / TOM_CH7_TRIG_CCU1 selected 10_B = TOM_CH0_TRIG_CCU1 selected / TOM_CH8_TRIG_CCU1 selected 11_B = TOM_CH2_TRIG_CCU1 selected / TOM_CH9_TRIG_CCU1 selected
FSOL	31:24	rw	**Fast Shutoff Level for TOM[i] channel 0 to 7**
0	11	r	**Reserved** Read as zero, shall be written as zero.

28.21.5.2 Register SPE[i]_PAT

SPEi Input Pattern Definition Register

Only the first block of valid input patterns defines the commutation. All input pattern following the first marked invalid input pattern are ignored.

SPEi_PAT (i=0-5)

SPEi Input Pattern Definition Register (000804$_H$+i*80$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IP7_PAT			IP7_VAL	IP6_PAT			IP6_VAL	IP5_PAT			IP5_VAL	IP4_PAT			IP4_VAL
rw			rw	rw			rw	rw			rw	rw			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IP3_PAT			IP3_VAL	IP2_PAT			IP2_VAL	IP1_PAT			IP1_VAL	IP0_PAT			IP0_VAL
rw			rw	rw			rw	rw			rw	rw			rw

Field	Bits	Type	Description
IPz_VAL (z=0-7)	4*z	rw	**Input pattern z is a valid pattern** 0_B Pattern invalid 1_B Pattern valid
IPz_PAT (z=0-7)	4*z+3:4*z+1	rw	**Input pattern z** Bit field defines the input pattern z of the SPE input signals. Bit 1 defines the TIM[i]_CHx(48) input signal. Bit 2 defines the TIM[i]_CHy(48) input signal. Bit 3 defines the TIM[i]_CHz(48) input signal.

28.21.5.3 Register SPE[i]_OUT_PAT[z]

SPEi Output Definition Register z

Register SPE_OUT_PAT[z] defines the output selection for TOM[i]_CH0 to TOM[i]_CH7 depending on actual input pattern IP[z]_PAT.

SPEi_OUT_PATz (i=0-5;z=0-7)

SPEi Output Definition Register z (000808$_H$+i*80$_H$+z*4) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPE_OUT_PAT															
rw															

Field	Bits	Type	Description
SPE_OUT_PAT	15:0	rw	**SPE output control value for TOM_CH0 to TOM_CH7** SPE_OUT_PAT[n+1:n] defines output select signal of TOM[i]_CH[n] with n:0...7 Current output control selection for SPE[i]_OUT(0..7). 0000_HSet SPE_OUT(n) to TOM_CH0_SOUR 0001_HSet SPE_OUT(n) to TOM_CH1_SOUR 0002_HSet SPE_OUT(n) to 0 0003_HSet SPE_OUT(n) to 1
0	31:16	r	**Reserved** Read as zero, shall be written as zero

28.21.5.4 Register SPE[i]_OUT_CTRL

SPEi Output Control Register

SPEi_OUT_CTRL (i=0-5)

SPEi Output Control Register **(000828_H+i*80_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

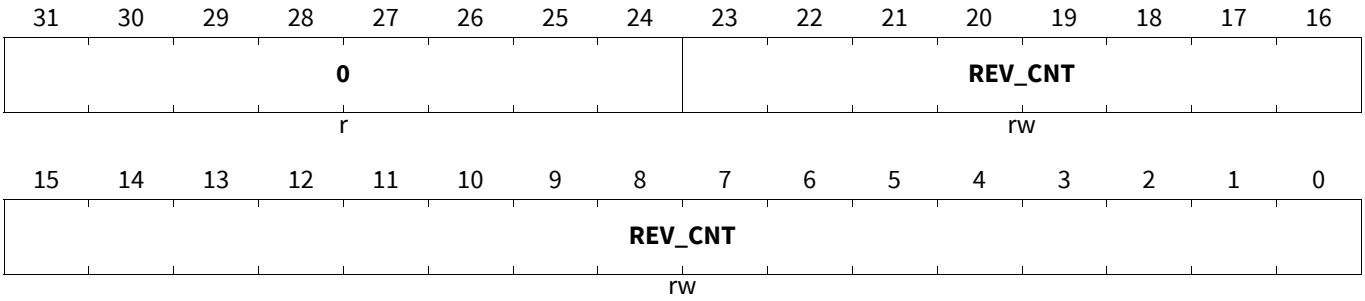
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPE_OUT_CTRL															
rw															

Field	Bits	Type	Description
SPE_OUT_CTRL	15:0	rw	**SPE output control value for TOM_CH0 to TOM_CH7** SPE_OUT_CTRL[n+1:n] defines output select signal of TOM_CHn. Current output control selection for SPE[i]_OUT(0..7). 0000_HSet SPE_OUT(n) to TOM_CH0_SOUR 0001_HSet SPE_OUT(n) to TOM_CH1_SOUR 0002_HSet SPE_OUT(n) to 0 0003_HSet SPE_OUT(n) to 1 with n: 0..7
0	31:16	r	**Reserved** Read as zero, shall be written as zero.

28.21.5.5 Register SPE[i]_REV_CNT

SPEi Input Revolution Counter

SPEi_REV_CNT (i=0-5)
SPEi Input Revolution Counter **(000840_H+i*80_H)** **Application Reset Value: 0000 0000_H**

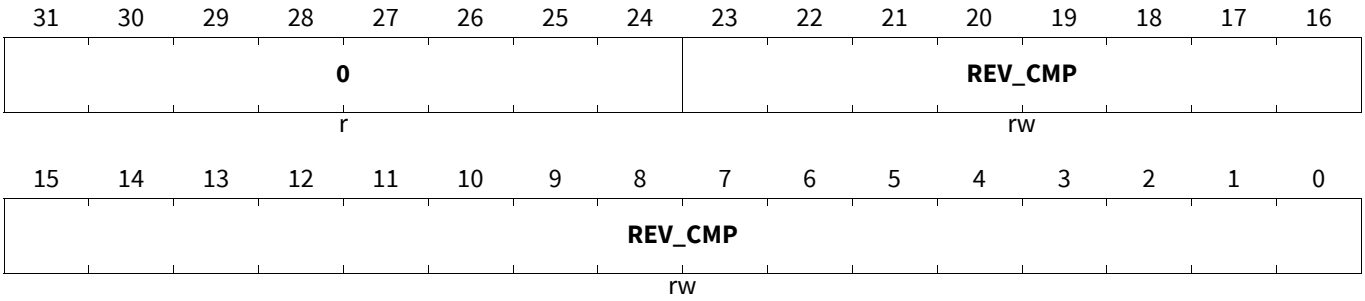


Field	Bits	Type	Description
REV_CNT	23:0	rw	**Input signal revolution counter** The counter is running if SPE module is enabled (bit SPE_EN). REV_CNT is incrementing if SPE_PAT_PTR is incrementing. REV_CNT is decrementing if SPE_PAT_PTR is decrementing-
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.21.5.6 Register SPE[i]_REV_CMP

SPEi Revolution Counter Compare Value

SPEi_REV_CMP (i=0-5)
SPEi Revolution Counter Compare Value **(000844_H+i*80_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
REV_CMP	23:0	rw	**Input signal revolution counter compare value** The interrupt *SPE[i]_RCMP* is raised when the SPE[i]_REV_CNT value equals the SPE[i]_REV_CMP register. It should be noted that *SPE[i]_RCMP* is only raised if an incrementation or decrementation of SPE[i]_REV_CNT is applied, due to an input signal change. Any update of SPE[i]_REV_CNT or SPE[i]_REV_CMP via AEI does not raise a *SPE[i]_RCMP* interrupt.

Field	Bits	Type	Description
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.21.5.7 Register SPE[i]_IRQ_NOTIFY

SPEi Interrupt Notification Register

SPEi_IRQ_NOTIFY (i=0-5)

SPEi Interrupt Notification Register **($00082C_H$+i*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											SPE_RCMP	SPE_BIS	SPE_PERR	SPE_DCHG	SPE_NIPD
r											rw	rw	rw	rw	rw

Field	Bits	Type	Description
SPE_NIPD	0	rw	**New input pattern interrupt occurred** This bit will be cleared on a CPU write access of value 1. (As the bit is rw, otherwise no clear.) A read access leaves the bit unchanged. 0_B No interrupt occurred 1_B New input pattern detected interrupt occurred
SPE_DCHG	1	rw	**SPE_DIR bit changed on behalf of new input pattern** Coding see bit 0.
SPE_PERR	2	rw	**Wrong or invalid pattern detected at input** Coding see bit 0.
SPE_BIS	3	rw	**Bouncing input signal detected** Coding see bit 0.
SPE_RCMP	4	rw	**SPE revolution counter match event** Coding see bit 0.
0	31:5	r	**Reserved** Read as zero, shall be written as zero.

28.21.5.8 Register SPE[i]_IRQ_EN

SPEi Interrupt Enable Register

SPEi_IRQ_EN (i=0-5)

SPEi Interrupt Enable Register **(000830_H+i*80_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											SPE_RCMP_IRQ_EN	SPE_BIS_IRQ_EN	SPE_PERR_IRQ_EN	SPE_DCHG_IRQ_EN	SPE_NIPD_IRQ_EN
r											rw	rw	rw	rw	rw

Field	Bits	Type	Description
SPE_NIPD_IRQ_EN	0	rw	**SPE_NIPD_IRQ interrupt enable** 0_B Disable interrupt, interrupt is not visible outside GTM 1_B Enable interrupt, interrupt is visible outside GTM
SPE_DCHG_IRQ_EN	1	rw	**SPE_DCHG_IRQ interrupt enable** Coding see bit 0.
SPE_PERR_IRQ_EN	2	rw	**SPE_PERR_IRQ interrupt enable** Coding see bit 0.
SPE_BIS_IRQ_EN	3	rw	**SPE_BIS_IRQ interrupt enable** Coding see bit 0.
SPE_RCMP_IRQ_EN	4	rw	**SPE_RCMP_IRQ interrupt enable** Coding see bit 0.
0	31:5	r	**Reserved** Read as zero, shall be written as zero.

28.21.5.9 Register SPE[i]_IRQ_FORCINT

SPEi Interrupt Generation by Software

SPEi_IRQ_FORCINT (i=0-5)

SPEi Interrupt Generation by Software **(000834_H+i*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											TRG_SPE_RCMP	TRG_SPE_BIS	TRG_SPE_PERR	TRG_SPE_DCHG	TRG_SPE_NIPD
r											rw	rw	rw	rw	rw

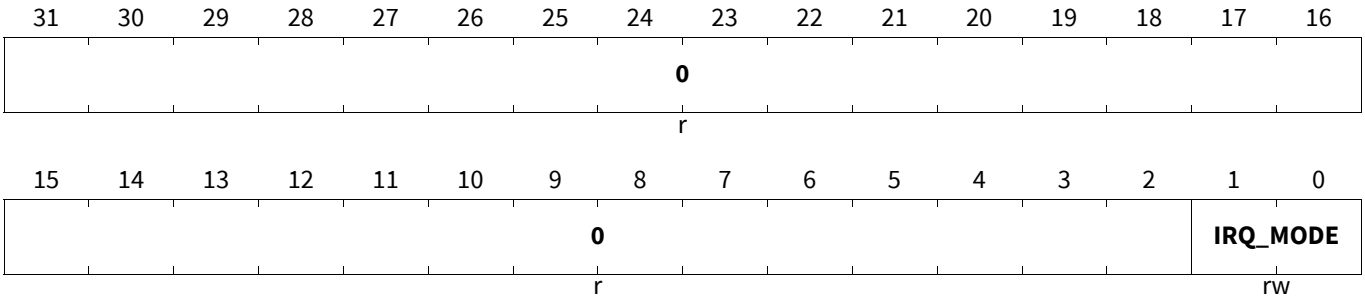
Field	Bits	Type	Description
TRG_SPE_NIPD	0	rw	**Force interrupt of SPE_NIPD** This bit is cleared automatically after interrupt is released. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B Corresponding bit in status register will not be forced 1_B Assert corresponding field in SPE_IRQ_NOTIFY register
TRG_SPE_DCHG	1	rw	**Force interrupt of SPE_DCHG** Coding see bit 0.
TRG_SPE_PERR	2	rw	**Force interrupt of SPE_PERR** Coding see bit 0.
TRG_SPE_BIS	3	rw	**Force interrupt of SPE_BIS** Coding see bit 0.
TRG_SPE_RCMP	4	rw	**Force interrupt of SPE_RCMP** Coding see bit 0.
0	31:5	r	**Reserved** Read as zero, shall be written as zero.

28.21.5.10Register SPE[i]_IRQ_MODE

SPEi Interrupt Mode Configuration Register

SPEi_IRQ_MODE (i=0-5)

SPEi Interrupt Mode Configuration Register(000838_H+i*80_H) **Application Reset Value: 0000 0000_H**



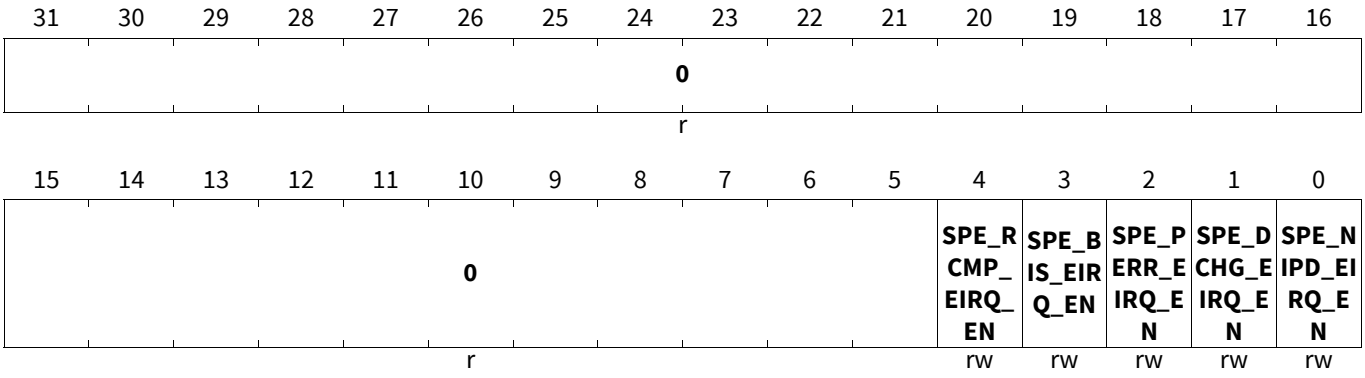
Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** The interrupt modes are described in **Section 28.4.5**. 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero

28.21.5.11Register SPE[i]_EIRQ_EN

SPEi Error Interrupt Enable Register

SPEi_EIRQ_EN (i=0-5)

SPEi Error Interrupt Enable Register **($00083C_H$+i*80_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SPE_NIPD_EIRQ_EN	0	rw	**SPE_NIPD_EIRQ interrupt enable** 0_B Disable error interrupt, error interrupt is not visible outside GTM 1_B Enable error interrupt, error interrupt is visible outside GTM

Field	Bits	Type	Description
SPE_DCHG_EIRQ_EN	1	rw	**SPE_DCHG_EIRQ error interrupt enable** Coding see bit 0.
SPE_PERR_EIRQ_EN	2	rw	**SPE_PERR_EIRQ error interrupt enable** Coding see bit 0.
SPE_BIS_EIRQ_EN	3	rw	**SPE_BIS_EIRQ error interrupt enable** Coding see bit 0.
SPE_RCMP_EIRQ_EN	4	rw	**SPE_RCMP_EIRQ error interrupt enable** Coding see bit 0.
0	31:5	r	**Reserved** Read as zero, shall be written as zero.

28.21.5.12Register SPE[i]_CTRL_STAT2

SPEi Control Status Register 2

SPEi_CTRL_STAT2 (i=0-5)
SPEi Control Status Register 2 **(000848_H+i*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

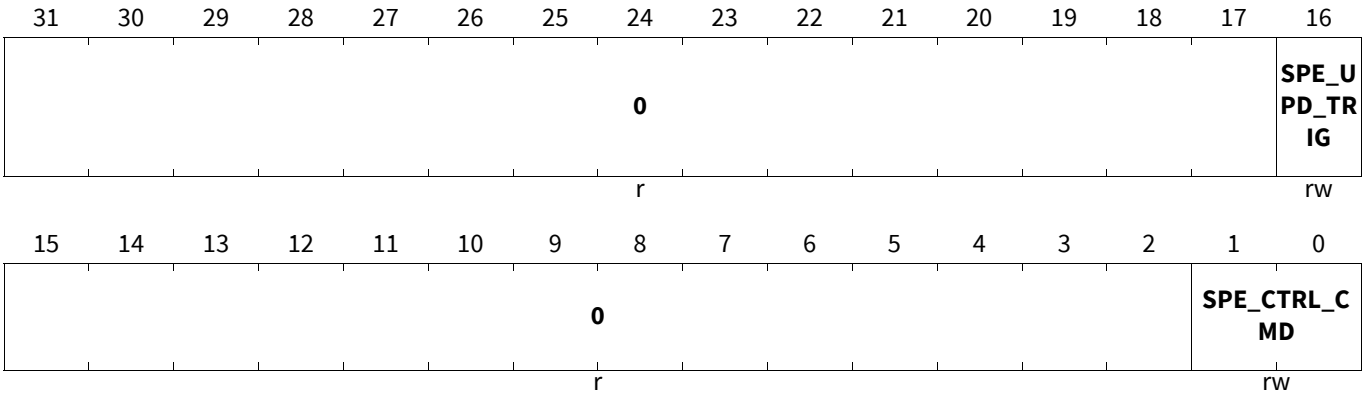
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					SPE_PAT_PTR_BWD			0							
r					rw			r							

Field	Bits	Type	Description
SPE_PAT_PTR_BWD	10:8	rw	**Pattern selector for TOM output signals in case of SPE_CTRL_CMD = 0b01 (e.g. backward direction)** Index into the SPE[i]_OUT_PAT[z] register table in case of SPE_CTRL_CMD = 0b01 which may be used for backward direction. Each register SPE[i]_OUT_PAT[x] is fixed assigned to one bit field IPx_PAT of register SPE[i]_PAT. Thus, the pointer SPE[i]_PAT_PTR_BWD represents an index to the selected SPE[i]_OUT_PAT[x] register as well as the actual detected input pattern IPx_PAT. The index pointer SPE_PAT_PTR_BWD is used if SPE_CTRL_CMD = 0b01. The index pointer SPE_PAT_PTR is used if SPE_CTRL_CMD = 0b00 (by default).
0	7:0, 31:11	r	**Reserved** Read as zero, shall be written as zero.

28.21.5.13Register SPE[i]_CMD

SPEi Command register

SPEi_CMD (i=0-5)
SPEi Command register **($00084C_H$+i*80_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
SPE_CTRL_CMD	1:0	rw	**SPE control command** 00_B Use SPE_PAT_PTR as an index pointer to select SPE[i]_OUT_PAT[z] 01_B Use SPE_PAT_PTR_BWD as an index pointer to select SPE[i]_OUT_PAT[z] 10_B Select SPE[i]_OUT_PAT6 11_B Select SPE[i]_OUT_PAT7
SPE_UPD_TRIG	16	rw	**SPE updater trigger** 1 = trigger update of SPE_OUT_CTRL with register selected by CTR_CMD multiplexer. This bit is automatically reset to 0.
0	15:2, 31:17	r	**Reserved** Read as zero, shall be written as zero.

28.22 Interrupt Concentrator Module (ICM)

28.22.1 Overview

The Interrupt Concentrator Module (ICM) is used to bundle the GTM interrupt lines of the individual sub-modules in a reasonable manner into interrupt groups. By this bundling a smaller amount of interrupt lines is visible at the outside of the GTM.

The individual interrupts of the GTM sub-modules and channels have to be enabled or disabled inside the sub-modules and channels.

The feed through architecture of bundled interrupt lines is used for the sub-modules AEI, ARU, BRC, CMP, SPE, PSM, TIM, DPLL, TOM, ATOM and MCS.

To determine the detailed interrupt source the microcontroller has to read the sub-module/channel interrupt notification register **NOTIFY** and serve the channel individual interrupt.

Please note, that the interrupts are only visible inside the ICM and in consequence outside of the GTM, when the interrupt is enabled inside the sub-modules themselves.

28.22.2 Bundling

The GTM sub-module individual interrupt sources are connected to the ICM. There, the individual interrupt lines are either feed through and signaled to the outside world or bundled a second time into groups and are then signaled to the outside world.The ICM interrupt bundling is described in the following sections.

28.22.2.1 GTM Infrastructure Interrupt Bundling

The first interrupt group contains interrupts of the infrastructure and safety components of the GTM. This interrupt group includes therefore interrupt lines coming from the AEI, ARU, BRC, PSM, SPE and CMP sub-modules. In this interrupt group each individual channel of the sub-modules has its own interrupt line to the outside world.

Thus, the active interrupt line can be used by the CPU to determine the GTM sub-module channel that raised the interrupt. The interrupts are also represented in the **ICM_IRQG_0** register. This register is typically not read by the CPU, but it is readable.

In addition the interrupt line status for 8 channels of each FIFO are shown in the **ICM_IRQG_PSM_0_CI** register. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_PSM_0_CI** register are typically not read out by the CPU, but they are readable.

In addition the interrupt line status for each SPE are shown in the **ICM_IRQG_SPE_CI** register. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_SPE_CI** register are typically not read out by the CPU, but they are readable.

28.22.2.2 DPLL Interrupt Bundling

The DPLL Interrupt group handles the interrupts coming from the DPLL sub-module of the GTM. Each of the individual DPLL interrupt lines has its own dedicated interrupt line to the outside world. The interrupts are additionally identified in the **ICM_IRQG_1** interrupt group register. This register is typically not read out by the CPU, but it is readable.

28.22.2.3 TIM Interrupt Bundling

Inside this group sub-modules which handle GTM input signals are treated. This is the case for the TIM[i] sub-modules. Each TIM sub-module channel is able to generate six (6) individual interrupts if enabled inside the TIM channel. This six interrupts are bundled into one interrupt per TIM channel connected to the ICM.

The ICM does no further bundling. Thus, for the GTM 32 interrupt lines *TIM[i]_IRQ*[y] are provided for the external microcontroller. The channel responsible for the interrupt can be determined by the raised interrupt line.

In addition, the **ICM_IRQG_2** and **ICM_IRQG_3** registers are mirrors for the TIM sub-module channel interrupts and typically not read out by the CPU, but it is readable.

Note: The above text is identical to the Bosch text published inside the GTM IP 3.1.5.1 specification. According to the migration guide provided by Bosch AE, the interrupt system scales in the amount with the number TIM modules. Each TIM provides 8 interrupt outputs according to the IRQG_2 and _3 registers. Therefore for adevice with 8 TIM modules 64 interrupts are provided to the outside of the GTM IP module. If Bosch AE publishes a new text this text will be provided by Infineon.

28.22.2.4 MCS Interrupt Bundling

For complex signal output generation, the MCS sub-modules are used inside the GTM. Each of these MCS sub-modules could have 8 channels with one interrupt line. This interrupt line is connected to the ICM sub-module and is feed through directly to the outside world.

In addition the interrupt line status for the first 8 channels of each MCS is shown in the **ICM_IRQG_4** and **ICM_IRQG_5** register. The interrupt line status for all used channels of each MCS are shown in the **ICM_IRQG_MCS[i]_CI** register. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQ4(/_5)** and **ICM_IRQG_MCS[i]_CI** register are typically not read out by the CPU, but they are readable.

28.22.2.5 TOM and ATOM Interrupt Bundling

For the TOM and ATOM sub-modules, the interrupts are bundled within the ICM sub-module a second time to reduce external interrupt lines. The interrupts are ORed in a manner that one GTM external interrupt line represents two adjacent TOM or ATOM channel interrupts. For TOM[i] and ATOM[i] the bundling is shown in **Figure 138**.

TOM[i]-input IRQs [i]=0..number of TOM's-1	TOM-output IRQs (OR-ed)		ATOM[i]-input IRQs [i]=0..number of ATOM's-1	ATOM-output IRQs (OR-ed)
TOM[i]_CH0_IRQ	GTM_TOM[i]_IRQ[0]		ATOM[i]_CH0_IRQ	GTM_ATOM[i]_IRQ[0]
TOM[i]_CH1_IRQ			ATOM[i]_CH1_IRQ	
TOM[i]_CH2_IRQ	GTM_TOM[i]_IRQ[1]		ATOM[i]_CH2_IRQ	GTM_ATOM[i]_IRQ[1]
TOM[i]_CH3_IRQ			ATOM[i]_CH3_IRQ	
TOM[i]_CH4_IRQ	GTM_TOM[i]_IRQ[2]		ATOM[i]_CH4_IRQ	GTM_ATOM[i]_IRQ[2]
TOM[i]_CH5_IRQ			ATOM[i]_CH5_IRQ	
TOM[i]_CH6_IRQ	GTM_TOM[i]_IRQ[3]		ATOM[i]_CH6_IRQ	GTM_ATOM[i]_IRQ[3]
TOM[i]_CH7_IRQ			ATOM[i]_CH7_IRQ	
TOM[i]_CH8_IRQ	GTM_TOM[i]_IRQ[4]			
TOM[i]_CH9_IRQ				
TOM[i]_CH10_IRQ	GTM_TOM[i]_IRQ[5]			
TOM[i]_CH11_IRQ				
TOM[i]_CH12_IRQ	GTM_TOM[i]_IRQ[6]			
TOM[i]_CH13_IRQ				
TOM[i]_CH14_IRQ	GTM_TOM[i]_IRQ[7]			
TOM[i]_CH15_IRQ				

Figure 138 TOM and ATOM interrupt bundling within ICM

The interrupts coming from the TOM[i] sub-modules are registered in the **ICM_IRQG_6 / ICM_IRQG_7 / ICM_IRQG_8** register. Always two TOM's are bundled in one ICM register, TOM0 and TOM1 are bundled in **ICM_IRQG_6**. To identify the TOM sub-module channel where the interrupt occurred, the CPU has to read out the **ICM_IRQG_6(/_7/_8)** register first before it goes to the TOM sub-module channel itself.

The **ICM_IRQG_6(/_7/_8)** register bits are cleared automatically, when their corresponding interrupt in the sub-module channels is cleared.

The interrupts coming from the ATOM[i] sub-modules are registered in the **ICM_IRQG_9** / **ICM_IRQG_10 /ICM_IRQG_11** register. Always four ATOM's are bundled in one ICM register. ATOM0, ATOM1, ATOM2 and ATOM3 are bundled in **ICM_IRQG_9.** To identify the ATOM sub-module channel where the interrupt occurred, the CPU has to read out the **ICM_IRQG_9(/_10/_11)** register first before it goes to the ATOM sub-module channel itself.

The **ICM_IRQG_9(/_10/_11)** register bits are cleared automatically, when their corresponding interrupt in the sub-module channels is cleared.

In addition the interrupt line status of two 16 channels TOM are shown in each **ICM_IRQG_TOM_[k]_CI (k:0...2)** register, TOM0 and TOM1 are bundled in **ICM_IRQG_TOM_0_CI**. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_TOM_[k]_CI** register are typically not read out by the CPU, but they are readable.

In addition the interrupt line status of four 8 channels ATOM are shown in each **ICM_IRQG_ATOM_[k]_CI (k:0...2)** register, ATOM0, ATOM1, ATOM2 and ATOM3 are bundled in **ICM_IRQG_ATOM_0_CI**. Typically, the interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_ATOM_[k]_CI** register are typically not read out by the CPU, but they are readable.

28.22.2.6 Module Error Interrupt Bundling

The Module Error Interrupt group handles the error interrupts coming from the BRC, FIFO, TIM, MCS, SPE, CMP, DPLL sub-module of the GTM. The Module Error interrupts are additionally identified in the **ICM_IRQG_MEI** error interrupt group register. This register is typically not read out by the CPU, but it is readable.

In addition the error interrupt line status for each SPE are shown in the **ICM_IRQG_SPE_CEI** register. Typically, the error interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_SPE_CEI** register are typically not read out by the CPU, but they are readable.

28.22.2.7 FIFO Channel Error Interrupt Bundling

The FIFO Channel Error Interrupt group handles the error interrupts coming from the FIFO channel of the GTM. The FIFO Channel Error interrupts are additionally identified in the **ICM_IRQG_CEI0** error interrupt group register. This register is typically not read out by the CPU, but it is readable.

The **ICM_IRQG_CEI0** register bits are cleared automatically, when their corresponding error interrupt in the sub-module channel is cleared.

In addition the error interrupt line status for 8 channels of each FIFO are shown in the **ICM_IRQG_PSM_0_CEI** register. Typically, the error interrupt source is determined by the corresponding interrupt line and the **ICM_IRQG_PSM_0_CEI** register are typically not read out by the CPU, but they are readable.

28.22.2.8 TIM Channel Error Interrupt Bundling

The TIM Channel Error Interrupt group handles the error interrupts coming from the TIM channel of the GTM. The TIM Channel Error interrupts are additionally identified for the sub-modules TIM0, TIM1, TIM2 and TIM3 in the **ICM_IRQG_CEI1** error interrupt group register and for the sub-modules TIM4, TIM5 and TIM6 in the **ICM_IRQG_CEI2** error interrupt group register. These register are typically not read out by the CPU, but they are readable.

The **ICM_IRQG_CEI1** and **ICM_IRQG_CEI2** register bits are cleared automatically, when their corresponding error interrupt in the sub-module channel is cleared.

28.22.2.9 MCS Channel Error Interrupt Bundling

The MCS Channel Error Interrupt group handles the error interrupts coming from the MCS channel of the GTM. All used 8 MCS Channel Error interrupts are additionally identified for each sub-modules MCS[i] in the **ICM_IRQG_MCS[i]_CEI** error interrupt group register. The first 8 MCS Channel Error interrupts are additionally identified for the sub-modules MCS0, MCS1, MCS2 and MCS3 in the **ICM_IRQG_CEI3** error interrupt group register and for the sub-modules MCS4, MCS5, MCS6 and MCS7 in the **ICM_IRQG_CEI4** error interrupt group register. These register are typically not read out by the CPU, but they are readable.

The **ICM_IRQG_MCS[i]_CEI**, **ICM_IRQG_CEI3** and **ICM_IRQG_CEI4** register bits are cleared automatically, when their corresponding error interrupt in the sub-module channel is cleared.

28.22.2.10 Error Interrupt Cluster Bundling

The Error Interrupt lines of up to 4 clusters are bundled in each **ICM_IRQG_CLS_[i]_MEI**.Actually each cluster collects one EIRQ of one TIM, MCS, SPE and FIFO.These register are typically not read out by the CPU, but they are readable.

28.22.3 ICM Interrupt Signals

Table 147 ICM Interrupt Signals

Signal	Description
GTM_AEI_IRQ	AEI Shared interrupt
GTM_ARU_IRQ[2:0]	[0]: ARU_NEW_DATA0 Interrupt [1]: ARU_NEW_DATA1 Interrupt [2]: ARU_ACC_ACK Interrupt
GTM_BRC_IRQ	BRC Shared interrupt
GTM_CMP_IRQ	CMP Shared interrupt
GTM_SPE[i]_IRQ	SPE Shared interrupt (i: 0…number of SPE's-1)
GTM_PSM[i]_IRQ[x]	PSM Shared interrupts (x: 0…7) (i: 0…number of PSM's-1)
GTM_DPLL_IRQ[0]	DPLL_DCGI: DPLL direction change interrupt
GTM_DPLL_IRQ[1]	DPLL_EDI; DPLL enable or disable interrupt
GTM_DPLL_IRQ[2]	DPLL_TINI: DPLL TRIG. min. hold time (THMI) viol. detected
GTM_DPLL_IRQ[3]	DPLL_TAXI: DPLL TRIG. max. hold time (THMA) viol. detected
GTM_DPLL_IRQ[4]	DPLL_SISI: DPLL STATE inactive slope detected
GTM_DPLL_IRQ[5]	DPLL_TISI: DPLL TRIGGER inactive slope detected
GTM_DPLL_IRQ[6]	DPLL_MSI: DPLL Missing STATE interrupt
GTM_DPLL_IRQ[7]	DPLL_MTI: DPLL Missing TRIGGER interrupt
GTM_DPLL_IRQ[8]	DPLL_SASI: DPLL STATE active slope detected
GTM_DPLL_IRQ[9]	DPLL_TASI: DPLL TRIG. active slope det. while NTI_CNT is 0
GTM_DPLL_IRQ[10]	DPLL_PWI: DPLL Plausibility window (PVT) viol. int. of TRIG.
GTM_DPLL_IRQ[11]	DPLL_W2I: DPLL Write access to RAM region 2 interrupt
GTM_DPLL_IRQ[12]	DPLL_W1I: DPLL Write access to RAM region 1b or 1c int.
GTM_DPLL_IRQ[13]	DPLL_GL1I: DPLL Get of lock interrupt for SUB_INC1
GTM_DPLL_IRQ[14]	DPLL_LL1I: DPLL Lost of lock interrupt for SUB_INC1
GTM_DPLL_IRQ[15]	DPLL_EI: DPLL Error interrupt
GTM_DPLL_IRQ[16]	DPLL_GL2I: DPLL Get of lock interrupt for SUB_INC2
GTM_DPLL_IRQ[17]	DPLL_LL2I: DPLL Lost of lock interrupt for SUB_INC2
GTM_DPLL_IRQ[18]	DPLL_TE0I: DPLL TRIGGER event interrupt 0
GTM_DPLL_IRQ[19]	DPLL_TE1I: DPLL TRIGGER event interrupt 1
GTM_DPLL_IRQ[20]	DPLL_TE2I: DPLL TRIGGER event interrupt 2
GTM_DPLL_IRQ[21]	DPLL_TE3I: DPLL TRIGGER event interrupt 3
GTM_DPLL_IRQ[22]	DPLL_TE4I; DPLL TRIGGER event interrupt 4
GTM_DPLL_IRQ[23]	DPLL_CDTI; DPLL calculated duration interrupt for TRIGGER
GTM_DPLL_IRQ[24]	DPLL_CDSI; DPLL calculated duration interrupt for STATE
GTM_DPLL_IRQ[25]	DPLL_TORI; TRIGGER out of range interrupt
GTM_DPLL_IRQ[26]	DPLL_SORI; STATE out of range interrupt
GTM_TIM[i]_IRQ[x]	TIM Shared interrupts (i: 0…number of TIM's-1) (x: 0…7)
GTM_MCS[i]_IRQ[x]	MCS Interrupt for channel x (x: 0…8) (i: 0…number of MCS's-1)
GTM_TOM[i]_IRQ[x]	TOM Shared interrupts for x:0…7 = {ch0||ch1…ch14||ch15} (i: 0…number of TOM's-1)

Table 147 ICM Interrupt Signals (cont'd)

Signal	Description
GTM_ATOM[i]_IRQ[x]	ATOM Shared interrupts for x:0…3 = {ch0||ch1…ch6||ch7} (i: 0…number of ATOM's-1)
GTM_ERR_IRQ	GTM Error Interrupt

Above table shows the GTM interrupt lines that are visible at the outside of the IP.

28.22.4 ICM Configuration Register Overview

Table 148 ICM Configuration Register Overview

Register Name	Description	see Page
ICM_IRQG_0	ICM Interrupt group register covering infrastructure and safety components (ARU, BRC, AEI, PSM0, PSM1, MAP, CMP,SPE)	689
ICM_IRQG_1	ICM Interrupt group register covering DPLL	690
ICM_IRQG_2	ICM Interrupt group register covering TIM0, TIM1, TIM2, TIM3	692
ICM_IRQG_3	ICM Interrupt group register covering TIM4, TIM5, TIM6, TIM7	694
ICM_IRQG_4	ICM Interrupt group register covering MCS0 to MCS3 sub-modules	695
ICM_IRQG_5	ICM Interrupt group register covering MCS4 to MCS6 sub-modules	696
ICM_IRQG_6	ICM Interrupt group register covering GTM output sub-modules TOM0 to TOM1	697
ICM_IRQG_7	ICM Interrupt group register covering GTM output sub-modules TOM2 to TOM3	698
ICM_IRQG_8	ICM Interrupt group register covering GTM output sub-modules TOM4 to TOM5	699
ICM_IRQG_9	ICM Interrupt group register covering GTM output sub-modules ATOM0, ATOM1, ATOM2 and ATOM3	700
ICM_IRQG_10	ICM Interrupt group register covering GTM output sub-modules ATOM4 to ATOM7	701
ICM_IRQG_11	ICM Interrupt group register covering GTM output sub-modules ATOM8 to ATOM11	702
ICM_IRQG_MEI	ICM Interrupt group register for module error interrupt information	703
ICM_IRQG_CEI0	ICM Interrupt group register 0 for channel error interrupt information	705
ICM_IRQG_CEI1	ICM Interrupt group register 1 for channel error interrupt information	706
ICM_IRQG_CEI2	ICM Interrupt group register 2 for channel error interrupt information	707
ICM_IRQG_CEI3	ICM Interrupt group register 3 for channel error interrupt information	708
ICM_IRQG_CEI4	ICM Interrupt group register 4 for channel error interrupt information	709
ICM_IRQG_MCS[i]_CI	ICM Interrupt group MCS i for Channel Interrupt information	710
ICM_IRQG_MCS[i]_CEI	ICM Interrupt group MCS i for Channel Error Interrupt information	710
ICM_IRQG_SPE_CI	ICM Interrupt group SPE for module Interrupt information	711
ICM_IRQG_SPE_CEI	ICM Interrupt group SPE for module Error Interrupt information	712
ICM_IRQG_PSM_0_CI	ICM Interrupt group PSM 0 for Channel Interrupt information of FIFO0, FIFO1, FIFO2	712
ICM_IRQG_PSM_0_CEI	ICM Interrupt group PSM 0 for Channel Error Interrupt information of FIFO0, FIFO1, FIFO2	713
ICM_IRQG_TOM_[k]_CI	ICM Interrupt group TOM k for Channel Interrupt information of TOMm	714

Table 148 ICM Configuration Register Overview (cont'd)

Register Name	Description	see Page
ICM_IRQG_ATOM_[k]_CI	ICM Interrupt group ATOM k for Channel Interrupt information of ATOMm	715
ICM_IRQG_CLS_[k]_MEI	ICM Interrupt group for module Error Interrupt information for each TIMm, MCSm, SPEm, FIFOm	716

28.22.5 ICM Configuration Register Description

28.22.5.1 Register ICM_IRQG_0

ICM Interrupt Group Register Covering Infrastructural and Safety Components ARU, BRC, AEI, PSM0, PSM1, MAP, CMP, SPE

ICM_IRQG_0
ICM Interrupt Group Register Covering Infrastructural and Safety Components ARU, BRC, AEI, PSM0, PSM1, MAP, CMP, SPE **(000600_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSM1_CH7_IRQ	PSM1_CH6_IRQ	PSM1_CH5_IRQ	PSM1_CH4_IRQ	PSM1_CH3_IRQ	PSM1_CH2_IRQ	PSM1_CH1_IRQ	PSM1_CH0_IRQ	PSM0_CH7_IRQ	PSM0_CH6_IRQ	PSM0_CH5_IRQ	PSM0_CH4_IRQ	PSM0_CH3_IRQ	PSM0_CH2_IRQ	PSM0_CH1_IRQ	PSM0_CH0_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				SPE5_IRQ	SPE4_IRQ	SPE3_IRQ	SPE2_IRQ	SPE1_IRQ	SPE0_IRQ	CMP_IRQ	AEI_IRQ	BRC_IRQ	ARU_ACC_ACK_IRQ	ARU_NEW_DATA1_IRQ	ARU_NEW_DATA0_IRQ
r				r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
ARU_NEW_DATA0_IRQ	0	r	**ARU_NEW_DATA0 interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
ARU_NEW_DATA1_IRQ	1	r	**ARU_NEW_DATA1 interrupt** Bit coding see bit 0.
ARU_ACC_ACK_IRQ	2	r	**ARU_ACC_ACK interrupt** Bit coding see bit 0.
BRC_IRQ	3	r	**BRC shared sub-module interrupt** Bit coding see bit 0.
AEI_IRQ	4	r	**AEI_IRQ interrupt** Bit coding see bit 0. Set this bit represents an OR function of the seven interrupt sources *AEI_TO_XPT*, *AEI_USP_ADDR*, *AEI_IM_ADDR*, *AEI_USP_BE*, *AEIM_USP_ADDR*, *AEIM_IM_ADDR* or *AEIM_USP_BE*.
CMP_IRQ	5	r	**CMP shared sub-module interrupt** Bit coding see bit 0.
SPE0_IRQ	6	r	**SPE0 shared sub-module interrupt** Bit coding see bit 0. Set this bit represents an OR function of the five interrupt sources *SPE_NIPD*, *SPE_DCHG*, *SPE_PERR*, *SPE_BIS* or *SPE_RCMP* of SPE instance 0.

Field	Bits	Type	Description
SPE1_IRQ	7	r	**SPE1 shared sub-module interrupt** See bit 0 and bit 6.
SPE2_IRQ	8	r	**SPE2 shared sub-module interrupt** See bit 0 and bit 6.
SPE3_IRQ	9	r	**SPE3 shared sub-module interrupt** See bit 0 and bit 6.
SPE4_IRQ	10	r	**SPE4 shared sub-module interrupt** See bit 0 and bit 6.
SPE5_IRQ	11	r	**SPE5 shared sub-module interrupt** See bit 0 and bit 6.
PSM0_CHx_IRQ (x=0-7)	x+16	r	**PSM0 shared sub-module channel x interrupt** See bit 0 and bit 6. Set this bit represents an OR function of the four interrupt sources *FIFO_EMPTY*, *FIFO_FULL*, *FIFO_LOWER_WM* or *FIFO_UPPER_WM* of FIFO instance 0 channel x.
PSM1_CHx_IRQ (x=0-7)	x+24	r	**PSM1 shared sub-module channel x interrupt** See bit 0 and bit 6. Set this bit represents an OR function of the four interrupt sources *FIFO_EMPTY*, *FIFO_FULL*, *FIFO_LOWER_WM* or *FIFO_UPPER_WM* of FIFO instance 1 channel x.
0	15:12	r	**Reserved** Read as zero, shall be written as zero.

28.22.5.2 Register ICM_IRQG_1

ICM Interrupt Group Register Covering DPLL

ICM_IRQG_1

ICM Interrupt Group Register Covering DPLL (000604$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					DPLL_SORI_IRQ	DPLL_TORI_IRQ	DPLL_CDSI_IRQ	DPLL_CDTI_IRQ	DPLL_TE4I_IRQ	DPLL_TE3I_IRQ	DPLL_TE2I_IRQ	DPLL_TE1I_IRQ	DPLL_TE0I_IRQ	DPLL_LL2I_IRQ	DPLL_GL2I_IRQ
r					r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPLL_EI_IRQ	DPLL_LL1I_IRQ	DPLL_GL1I_IRQ	DPLL_W1I_IRQ	DPLL_W2I_IRQ	DPLL_PWI_IRQ	DPLL_TASI_IRQ	DPLL_SASI_IRQ	DPLL_MTI_IRQ	DPLL_MSI_IRQ	DPLL_TISI_IRQ	DPLL_SISI_IRQ	DPLL_TAXI_IRQ	DPLL_TINI_IRQ	DPLL_EDI_IRQ	DPLL_DCGI_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
DPLL_DCGI_IRQ	0	r	**TRIGGER direction change detected** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
DPLL_EDI_IRQ	1	r	**DPLL enable/disable interrupt** Set this bit represents an OR function of the two interrupt sources *DPLL_PDI* or *DPLL_PEI.* Bit coding see bit 0.
DPLL_TINI_IRQ	2	r	**TRIGGER minimum hold time (THMI) violation detected interrupt** Bit coding see bit 0.
DPLL_TAXI_IRQ	3	r	**TRIGGER maximum hold time (THMA) violation detected interrupt** Bit coding see bit 0.
DPLL_SISI_IRQ	4	r	**STATE inactive slope detected interrupt** Bit coding see bit 0.
DPLL_TISI_IRQ	5	r	**TRIGGER inactive slope detected interrupt** Bit coding see bit 0.
DPLL_MSI_IRQ	6	r	**Missing STATE interrupt** Bit coding see bit 0.
DPLL_MTI_IRQ	7	r	**Missing TRIGGER interrupt** Bit coding see bit 0.
DPLL_SASI_IRQ	8	r	**STATE active slope detected** Bit coding see bit 0.
DPLL_TASI_IRQ	9	r	**TRIGGER active slope detected while NTI_CNT is zero** Bit coding see bit 0.
DPLL_PWI_IRQ	10	r	**Plausibility window (PVT) violation interrupt of TRIGGER** Bit coding see bit 0.
DPLL_W2I_IRQ	11	r	**Write access to RAM region 2 interrupt** Bit coding see bit 0.
DPLL_W1I_IRQ	12	r	**Write access to RAM region 1b or 1c interrupt** Bit coding see bit 0.
DPLL_GL1I_IRQ	13	r	**Get of lock interrupt for SUB_INC1** Bit coding see bit 0.
DPLL_LL1I_IRQ	14	r	**Loss of lock interrupt for SUB_INC1** Bit coding see bit 0.
DPLL_EI_IRQ	15	r	**Error interrupt** Bit coding see bit 0.
DPLL_GL2I_IRQ	16	r	**Get of lock interrupt for SUB_INC2** Bit coding see bit 0.
DPLL_LL2I_IRQ	17	r	**Loss of lock interrupt for SUB_INC2** Bit coding see bit 0.

Field	Bits	Type	Description
DPLL_TE0I_IRQ	18	r	**TRIGGER event interrupt 0** Bit coding see bit 0.
DPLL_TE1I_IRQ	19	r	**TRIGGER event interrupt 1** Bit coding see bit 0.
DPLL_TE2I_IRQ	20	r	**TRIGGER event interrupt 2** Bit coding see bit 0.
DPLL_TE3I_IRQ	21	r	**TRIGGER event interrupt 3** Bit coding see bit 0.
DPLL_TE4I_IRQ	22	r	**TRIGGER event interrupt 4** Bit coding see bit 0.
DPLL_CDTI_IRQ	23	r	**DPLL calculated duration interrupt for trigger** Bit coding see bit 0.
DPLL_CDSI_IRQ	24	r	**DPLL calculated duration interrupt for state** Bit coding see bit 0.
DPLL_TORI_IRQ	25	r	**DPLL calculated duration interrupt for state** Bit coding see bit 0.
DPLL_SORI_IRQ	26	r	**DPLL calculated duration interrupt for state** Bit coding see bit 0.
0	31:27	r	**Reserved** Read as zero, shall be written as zero.

28.22.5.3 Register ICM_IRQG_2

ICM Interrupt Group Register Covering TIM0, TIM1, TIM2, TIM3

ICM_IRQG_2

ICM Interrupt Group Register Covering TIM0, TIM1, TIM2, TIM3(000608_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIM3_CH7_IRQ	**TIM3_CH6_IRQ**	**TIM3_CH5_IRQ**	**TIM3_CH4_IRQ**	**TIM3_CH3_IRQ**	**TIM3_CH2_IRQ**	**TIM3_CH1_IRQ**	**TIM3_CH0_IRQ**	**TIM2_CH7_IRQ**	**TIM2_CH6_IRQ**	**TIM2_CH5_IRQ**	**TIM2_CH4_IRQ**	**TIM2_CH3_IRQ**	**TIM2_CH2_IRQ**	**TIM2_CH1_IRQ**	**TIM2_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM1_CH7_IRQ	**TIM1_CH6_IRQ**	**TIM1_CH5_IRQ**	**TIM1_CH4_IRQ**	**TIM1_CH3_IRQ**	**TIM1_CH2_IRQ**	**TIM1_CH1_IRQ**	**TIM1_CH0_IRQ**	**TIM0_CH7_IRQ**	**TIM0_CH6_IRQ**	**TIM0_CH5_IRQ**	**TIM0_CH4_IRQ**	**TIM0_CH3_IRQ**	**TIM0_CH2_IRQ**	**TIM0_CH1_IRQ**	**TIM0_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TIM0_CHx_IRQ (x=0-7)	x	r	**TIM0 shared interrupt channel x** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 0 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
TIM1_CHx_IRQ (x=0-7)	x+8	r	**TIM1 shared interrupt channel x** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 1 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
TIM2_CHx_IRQ (x=0-7)	x+16	r	**TIM2 shared interrupt channel x** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 2 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
TIM3_CHx_IRQ (x=0-7)	x+24	r	**TIM2 shared interrupt channel x** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 3 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.4 Register ICM_IRQG_3

ICM Interrupt Group Register Covering TIM4, TIM5, TIM6, TIM7

ICM_IRQG_3
ICM Interrupt Group Register Covering TIM4, TIM5, TIM6, TIM7(00060C$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIM7_CH7_IRQ	TIM7_CH6_IRQ	TIM7_CH5_IRQ	TIM7_CH4_IRQ	TIM7_CH3_IRQ	TIM7_CH2_IRQ	TIM7_CH1_IRQ	TIM7_CH0_IRQ	TIM6_CH7_IRQ	TIM6_CH6_IRQ	TIM6_CH5_IRQ	TIM6_CH4_IRQ	TIM6_CH3_IRQ	TIM6_CH2_IRQ	TIM6_CH1_IRQ	TIM6_CH0_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM5_CH7_IRQ	TIM5_CH6_IRQ	TIM5_CH5_IRQ	TIM5_CH4_IRQ	TIM5_CH3_IRQ	TIM5_CH2_IRQ	TIM5_CH1_IRQ	TIM5_CH0_IRQ	TIM4_CH7_IRQ	TIM4_CH6_IRQ	TIM4_CH5_IRQ	TIM4_CH4_IRQ	TIM4_CH3_IRQ	TIM4_CH2_IRQ	TIM4_CH1_IRQ	TIM4_CH0_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TIM4_CHx_IRQ (x=0-7)	x	r	**TIM4 shared interrupt channel x** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 4 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
TIM5_CHx_IRQ (x=0-7)	x+8	r	**TIM5 shared interrupt channel x** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 5 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
TIM6_CHx_IRQ (x=0-7)	x+16	r	**TIM6 shared interrupt channel x** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ*, *GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 6 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

Field	Bits	Type	Description
TIM7_CHx_IRQ (x=0-7)	x+24	r	**TIM7 shared interrupt channel x** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the six interrupt sources *NEWVALx_IRQ*, *ECNTOFLx_IRQ*, *CNTOFLx_IRQ, GPRXOFLx_IRQ*, GLITCHDETx_IRQ or *TODETx_IRQ* of TIM instance 7 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.5 Register ICM_IRQG_4

ICM Interrupt Group Register Covering MCS0 to MCS3 Sub-Modules

ICM_IRQG_4
ICM Interrupt Group Register Covering MCS0 to MCS3 Sub-Modules(000610_H)Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCS3_CH7_IRQ	**MCS3_CH6_IRQ**	**MCS3_CH5_IRQ**	**MCS3_CH4_IRQ**	**MCS3_CH3_IRQ**	**MCS3_CH2_IRQ**	**MCS3_CH1_IRQ**	**MCS3_CH0_IRQ**	**MCS2_CH7_IRQ**	**MCS2_CH6_IRQ**	**MCS2_CH5_IRQ**	**MCS2_CH4_IRQ**	**MCS2_CH3_IRQ**	**MCS2_CH2_IRQ**	**MCS2_CH1_IRQ**	**MCS2_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCS1_CH7_IRQ	**MCS1_CH6_IRQ**	**MCS1_CH5_IRQ**	**MCS1_CH4_IRQ**	**MCS1_CH3_IRQ**	**MCS1_CH2_IRQ**	**MCS1_CH1_IRQ**	**MCS1_CH0_IRQ**	**MCS0_CH7_IRQ**	**MCS0_CH6_IRQ**	**MCS0_CH5_IRQ**	**MCS0_CH4_IRQ**	**MCS0_CH3_IRQ**	**MCS0_CH2_IRQ**	**MCS0_CH1_IRQ**	**MCS0_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
MCS0_CHx_IRQ (x=0-7)	x	r	**MCS0 channel x interrupt** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 0 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
MCS1_CHx_IRQ (x=0-7)	x+8	r	**MCS1 channel x interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 1 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

Field	Bits	Type	Description
MCS2_CHx_IRQ (x=0-7)	x+16	r	**MCS2 channel x interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 2 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
MCS3_CHx_IRQ (x=0-7)	x+24	r	**MCS3 channel x interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 3 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.6 Register ICM_IRQG_5

ICM Interrupt Group Register Covering MCS4 to MCS6 Sub-Modules

ICM_IRQG_5
ICM Interrupt Group Register Covering MCS4 to MCS6 Sub-Modules(000614_H)Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCS7_CH7_IRQ	MCS7_CH6_IRQ	MCS7_CH5_IRQ	MCS7_CH4_IRQ	MCS7_CH3_IRQ	MCS7_CH2_IRQ	MCS7_CH1_IRQ	MCS7_CH0_IRQ	MCS6_CH7_IRQ	MCS6_CH6_IRQ	MCS6_CH5_IRQ	MCS6_CH4_IRQ	MCS6_CH3_IRQ	MCS6_CH2_IRQ	MCS6_CH1_IRQ	MCS6_CH0_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCS5_CH7_IRQ	MCS5_CH6_IRQ	MCS5_CH5_IRQ	MCS5_CH4_IRQ	MCS5_CH3_IRQ	MCS5_CH2_IRQ	MCS5_CH1_IRQ	MCS5_CH0_IRQ	MCS4_CH7_IRQ	MCS4_CH6_IRQ	MCS4_CH5_IRQ	MCS4_CH4_IRQ	MCS4_CH3_IRQ	MCS4_CH2_IRQ	MCS4_CH1_IRQ	MCS4_CH0_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
MCS4_CHx_IRQ (x=0-7)	x	r	**MCS4 channel x interrupt** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 4 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

Field	Bits	Type	Description
MCS5_CHx_IRQ (x=0-7)	x+8	r	**MCS5 channel x interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 5 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
MCS6_CHx_IRQ (x=0-7)	x+16	r	**MCS6 channel x interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 6 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
MCS7_CHx_IRQ (x=0-7)	x+24	r	**MCS7 channel x interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the three interrupt sources *MCS_IRQ*, *STK_ERR_IRQ* or *ERR_IRQ* of MCS instance 7 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.7 Register ICM_IRQG_6

ICM Interrupt Group Register Covering GTM Output Sub-Modules TOM0 to TOM1

ICM_IRQG_6
ICM Interrupt Group Register Covering GTM Output Sub-Modules TOM0 to TOM1(000618_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOM1_CH15_IRQ	**TOM1_CH14_IRQ**	**TOM1_CH13_IRQ**	**TOM1_CH12_IRQ**	**TOM1_CH11_IRQ**	**TOM1_CH10_IRQ**	**TOM1_CH9_IRQ**	**TOM1_CH8_IRQ**	**TOM1_CH7_IRQ**	**TOM1_CH6_IRQ**	**TOM1_CH5_IRQ**	**TOM1_CH4_IRQ**	**TOM1_CH3_IRQ**	**TOM1_CH2_IRQ**	**TOM1_CH1_IRQ**	**TOM1_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOM0_CH15_IRQ	**TOM0_CH14_IRQ**	**TOM0_CH13_IRQ**	**TOM0_CH12_IRQ**	**TOM0_CH11_IRQ**	**TOM0_CH10_IRQ**	**TOM0_CH9_IRQ**	**TOM0_CH8_IRQ**	**TOM0_CH7_IRQ**	**TOM0_CH6_IRQ**	**TOM0_CH5_IRQ**	**TOM0_CH4_IRQ**	**TOM0_CH3_IRQ**	**TOM0_CH2_IRQ**	**TOM0_CH1_IRQ**	**TOM0_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TOM0_CHx_IRQ (x=0-15)	x	r	**TOM0 channel x shared interrupt** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 0 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
TOM1_CHx_IRQ (x=0-15)	x+16	r	**TOM1 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 1 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.8 Register ICM_IRQG_7

ICM Interrupt Group Register Covering GTM Output Sub-Modules TOM2 to TOM3

ICM_IRQG_7
ICM Interrupt Group Register Covering GTM Output Sub-Modules TOM2 to TOM3($00061C_H$) **Application**
Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOM3_CH15_IRQ	**TOM3_CH14_IRQ**	**TOM3_CH13_IRQ**	**TOM3_CH12_IRQ**	**TOM3_CH11_IRQ**	**TOM3_CH10_IRQ**	**TOM3_CH9_IRQ**	**TOM3_CH8_IRQ**	**TOM3_CH7_IRQ**	**TOM3_CH6_IRQ**	**TOM3_CH5_IRQ**	**TOM3_CH4_IRQ**	**TOM3_CH3_IRQ**	**TOM3_CH2_IRQ**	**TOM3_CH1_IRQ**	**TOM3_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOM2_CH15_IRQ	**TOM2_CH14_IRQ**	**TOM2_CH13_IRQ**	**TOM2_CH12_IRQ**	**TOM2_CH11_IRQ**	**TOM2_CH10_IRQ**	**TOM2_CH9_IRQ**	**TOM2_CH8_IRQ**	**TOM2_CH7_IRQ**	**TOM2_CH6_IRQ**	**TOM2_CH5_IRQ**	**TOM2_CH4_IRQ**	**TOM2_CH3_IRQ**	**TOM2_CH2_IRQ**	**TOM2_CH1_IRQ**	**TOM2_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TOM2_CHx_IRQ (x=0-15)	x	r	**TOM2 channel x shared interrupt** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 2 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

Field	Bits	Type	Description
TOM3_CHx_IRQ (x=0-15)	x+16	r	**TOM3 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 3 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.9 Register ICM_IRQG_8

ICM Interrupt Group Register Covering GTM Output Sub-Modules TOM4 to TOM5

ICM_IRQG_8
ICM Interrupt Group Register Covering GTM Output Sub-Modules TOM4 to TOM5(000620_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOM5_CH15_IRQ	**TOM5_CH14_IRQ**	**TOM5_CH13_IRQ**	**TOM5_CH12_IRQ**	**TOM5_CH11_IRQ**	**TOM5_CH10_IRQ**	**TOM5_CH9_IRQ**	**TOM5_CH8_IRQ**	**TOM5_CH7_IRQ**	**TOM5_CH6_IRQ**	**TOM5_CH5_IRQ**	**TOM5_CH4_IRQ**	**TOM5_CH3_IRQ**	**TOM5_CH2_IRQ**	**TOM5_CH1_IRQ**	**TOM5_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOM4_CH15_IRQ	**TOM4_CH14_IRQ**	**TOM4_CH13_IRQ**	**TOM4_CH12_IRQ**	**TOM4_CH11_IRQ**	**TOM4_CH10_IRQ**	**TOM4_CH9_IRQ**	**TOM4_CH8_IRQ**	**TOM4_CH7_IRQ**	**TOM4_CH6_IRQ**	**TOM4_CH5_IRQ**	**TOM4_CH4_IRQ**	**TOM4_CH3_IRQ**	**TOM4_CH2_IRQ**	**TOM4_CH1_IRQ**	**TOM4_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TOM4_CHx_IRQ (x=0-15)	x	r	**TOM4 channel x shared interrupt** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 4 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
TOM5_CHx_IRQ (x=0-15)	x+16	r	**TOM5 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 5 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.10Register ICM_IRQG_9

ICM Interrupt Group Register Covering GTM Output Sub-Modules ATOM0, ATOM1, ATOM2 and ATOM3

ICM_IRQG_9
ICM Interrupt Group Register Covering GTM Output Sub-Modules ATOM0, ATOM1, ATOM2 and ATOM3
(000624_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATOM3_CH7_IRQ	ATOM3_CH6_IRQ	ATOM3_CH5_IRQ	ATOM3_CH4_IRQ	ATOM3_CH3_IRQ	ATOM3_CH2_IRQ	ATOM3_CH1_IRQ	ATOM3_CH0_IRQ	ATOM2_CH7_IRQ	ATOM2_CH6_IRQ	ATOM2_CH5_IRQ	ATOM2_CH4_IRQ	ATOM2_CH3_IRQ	ATOM2_CH2_IRQ	ATOM2_CH1_IRQ	ATOM2_CH0_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOM1_CH7_IRQ	ATOM1_CH6_IRQ	ATOM1_CH5_IRQ	ATOM1_CH4_IRQ	ATOM1_CH3_IRQ	ATOM1_CH2_IRQ	ATOM1_CH1_IRQ	ATOM1_CH0_IRQ	ATOM0_CH7_IRQ	ATOM0_CH6_IRQ	ATOM0_CH5_IRQ	ATOM0_CH4_IRQ	ATOM0_CH3_IRQ	ATOM0_CH2_IRQ	ATOM0_CH1_IRQ	ATOM0_CH0_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
ATOM0_CHx_IRQ (x=0-7)	x	r	**ATOM0 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 0 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
ATOM1_CHx_IRQ (x=0-7)	x+8	r	**ATOM1 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 1 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
ATOM2_CHx_IRQ (x=0-7)	x+16	r	**ATOM2 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 2 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

Field	Bits	Type	Description
ATOM3_CHx_IRQ (x=0-7)	x+24	r	**ATOM3 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 3 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.11Register ICM_IRQG_10

ICM Interrupt Group Register Covering GTM Output Sub-Modules ATOM4 to ATOM7

ICM_IRQG_10
ICM Interrupt Group Register Covering GTM Output Sub-Modules ATOM4 to ATOM7(000628_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATOM7_CH7_IRQ	**ATOM7_CH6_IRQ**	**ATOM7_CH5_IRQ**	**ATOM7_CH4_IRQ**	**ATOM7_CH3_IRQ**	**ATOM7_CH2_IRQ**	**ATOM7_CH1_IRQ**	**ATOM7_CH0_IRQ**	**ATOM6_CH7_IRQ**	**ATOM6_CH6_IRQ**	**ATOM6_CH5_IRQ**	**ATOM6_CH4_IRQ**	**ATOM6_CH3_IRQ**	**ATOM6_CH2_IRQ**	**ATOM6_CH1_IRQ**	**ATOM6_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOM5_CH7_IRQ	**ATOM5_CH6_IRQ**	**ATOM5_CH5_IRQ**	**ATOM5_CH4_IRQ**	**ATOM5_CH3_IRQ**	**ATOM5_CH2_IRQ**	**ATOM5_CH1_IRQ**	**ATOM5_CH0_IRQ**	**ATOM4_CH7_IRQ**	**ATOM4_CH6_IRQ**	**ATOM4_CH5_IRQ**	**ATOM4_CH4_IRQ**	**ATOM4_CH3_IRQ**	**ATOM4_CH2_IRQ**	**ATOM4_CH1_IRQ**	**ATOM4_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
ATOM4_CHx_IRQ (x=0-7)	x	r	**ATOM4 channel x shared interrupt** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 4 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
ATOM5_CHx_IRQ (x=0-7)	x+8	r	**ATOM5 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 5 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

Field	Bits	Type	Description
ATOM6_CHx_IRQ (x=0-7)	x+16	r	**ATOM6 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 6 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
ATOM7_CHx_IRQ (x=0-7)	x+24	r	**ATOM7 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 7 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.12Register ICM_IRQG_11

ICM Interrupt Group Register Covering GTM Output Sub-Modules ATOM8 to ATOM11

ICM_IRQG_11
ICM Interrupt Group Register Covering GTM Output Sub-Modules ATOM8 to ATOM11(00062C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATOM11_CH7_IRQ	ATOM11_CH6_IRQ	ATOM11_CH5_IRQ	ATOM11_CH4_IRQ	ATOM11_CH3_IRQ	ATOM11_CH2_IRQ	ATOM11_CH1_IRQ	ATOM11_CH0_IRQ	ATOM10_CH7_IRQ	ATOM10_CH6_IRQ	ATOM10_CH5_IRQ	ATOM10_CH4_IRQ	ATOM10_CH3_IRQ	ATOM10_CH2_IRQ	ATOM10_CH1_IRQ	ATOM10_CH0_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOM9_CH7_IRQ	ATOM9_CH6_IRQ	ATOM9_CH5_IRQ	ATOM9_CH4_IRQ	ATOM9_CH3_IRQ	ATOM9_CH2_IRQ	ATOM9_CH1_IRQ	ATOM9_CH0_IRQ	ATOM8_CH7_IRQ	ATOM8_CH6_IRQ	ATOM8_CH5_IRQ	ATOM8_CH4_IRQ	ATOM8_CH3_IRQ	ATOM8_CH2_IRQ	ATOM8_CH1_IRQ	ATOM8_CH0_IRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
ATOM8_CHx_IRQ (x=0-7)	x	r	**ATOM8 channel x shared interrupt** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 8 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

Field	Bits	Type	Description
ATOM9_CHx_IRQ (x=0-7)	x+8	r	**ATOM9 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 9 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
ATOM10_CHx_IRQ (x=0-7)	x+16	r	**ATOM10 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 10 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
ATOM11_CHx_IRQ (x=0-7)	x+24	r	**ATOM11 channel x shared interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 11 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.13Register ICM_IRQG_MEI

ICM Interrupt Group Register for Module Error Interrupt Information

ICM_IRQG_MEI

ICM Interrupt Group Register for Module Error Interrupt Information(000630_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						DPLL_EIRQ	CMP_EIRQ	SPE3_EIRQ	SPE2_EIRQ	SPE1_EIRQ	SPE0_EIRQ	MCS7_EIRQ	MCS6_EIRQ	MCS5_EIRQ	MCS4_EIRQ
r						r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCS3_EIRQ	MCS2_EIRQ	MCS1_EIRQ	MCS0_EIRQ	TIM7_EIRQ	TIM6_EIRQ	TIM5_EIRQ	TIM4_EIRQ	TIM3_EIRQ	TIM2_EIRQ	TIM1_EIRQ	TIM0_EIRQ	FIFO1_EIRQ	FIFO0_EIRQ	BRC_EIRQ	GTM_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
GTM_EIRQ	0	r	**AEI Error interrupt request** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the seven interrupt sources *AEI_TO_XPT_EIRQ*, *AEI_USP_ADDR_EIRQ*, *AEI_IM_ADDR_EIRQ*, *AEI_USP_BE_EIRQ*, *AEIM_USP_ADDR_EIRQ*, *AEIM_IM_ADDR_EIRQ* or *AEIM_USP_BE_EIRQ*. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
BRC_EIRQ	1	r	**BRC error interrupt** Bit coding see bit 0.
FIFO0_EIRQ	2	r	**FIFO0 error interrupt** Bit coding see bit 0.
FIFO1_EIRQ	3	r	**FIFO1 error interrupt** Bit coding see bit 0.
TIMx_EIRQ (x=0-7)	x+4	r	**TIMx error interrupt** Bit coding see bit 0.
MCSx_EIRQ (x=0-7)	x+12	r	**MCSx error interrupt** Bit coding see bit 0.
SPEx_EIRQ (x=0-3)	x+20	r	**SPEx error interrupt** Bit coding see bit 0.
CMP_EIRQ	24	r	**CMP error interrupt** Bit coding see bit 0.
DPLL_EIRQ	25	r	**DPLL error interrupt** Bit coding see bit 0.
0	31:26	r	**Reserved** Read as zero, shall be written as zero.

28.22.5.14Register ICM_IRQG_CEI0

ICM Interrupt Group Register 0 for Channel Error Interrupt Information

ICM_IRQG_CEI0
ICM Interrupt Group Register 0 for Channel Error Interrupt Information(000634_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								FIFO2_CH7_EIRQ	FIFO2_CH6_EIRQ	FIFO2_CH5_EIRQ	FIFO2_CH4_EIRQ	FIFO2_CH3_EIRQ	FIFO2_CH2_EIRQ	FIFO2_CH1_EIRQ	FIFO2_CH0_EIRQ
r								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFO1_CH7_EIRQ	FIFO1_CH6_EIRQ	FIFO1_CH5_EIRQ	FIFO1_CH4_EIRQ	FIFO1_CH3_EIRQ	FIFO1_CH2_EIRQ	FIFO1_CH1_EIRQ	FIFO1_CH0_EIRQ	FIFO0_CH7_EIRQ	FIFO0_CH6_EIRQ	FIFO0_CH5_EIRQ	FIFO0_CH4_EIRQ	FIFO0_CH3_EIRQ	FIFO0_CH2_EIRQ	FIFO0_CH1_EIRQ	FIFO0_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
FIFO0_CHx_EIRQ (x=0-7)	x	r	**FIFO0 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
FIFO1_CHx_EIRQ (x=0-7)	x+8	r	**FIFO1 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
FIFO2_CHx_EIRQ (x=0-7)	x+16	r	**FIFO2 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.22.5.15Register ICM_IRQG_CEI1

ICM Interrupt Group Register 1 for Channel Error Interrupt Information

ICM_IRQG_CEI1
ICM Interrupt Group Register 1 for Channel Error Interrupt Information(000638$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIM3_CH7_EIRQ	TIM3_CH6_EIRQ	TIM3_CH5_EIRQ	TIM3_CH4_EIRQ	TIM3_CH3_EIRQ	TIM3_CH2_EIRQ	TIM3_CH1_EIRQ	TIM3_CH0_EIRQ	TIM2_CH7_EIRQ	TIM2_CH6_EIRQ	TIM2_CH5_EIRQ	TIM2_CH4_EIRQ	TIM2_CH3_EIRQ	TIM2_CH2_EIRQ	TIM2_CH1_EIRQ	TIM2_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM1_CH7_EIRQ	TIM1_CH6_EIRQ	TIM1_CH5_EIRQ	TIM1_CH4_EIRQ	TIM1_CH3_EIRQ	TIM1_CH2_EIRQ	TIM1_CH1_EIRQ	TIM1_CH0_EIRQ	TIM0_CH7_EIRQ	TIM0_CH6_EIRQ	TIM0_CH5_EIRQ	TIM0_CH4_EIRQ	TIM0_CH3_EIRQ	TIM0_CH2_EIRQ	TIM0_CH1_EIRQ	TIM0_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TIM0_CHx_EIRQ (x=0-7)	x	r	**TIM0 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
TIM1_CHx_EIRQ (x=0-7)	x+8	r	**TIM1 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
TIM2_CHx_EIRQ (x=0-7)	x+16	r	**TIM2 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
TIM3_CHx_EIRQ (x=0-7)	x+24	r	**TIM3 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module

28.22.5.16Register ICM_IRQG_CEI2

ICM Interrupt Group Register 2 for Channel Error Interrupt Information

ICM_IRQG_CEI2
ICM Interrupt Group Register 2 for Channel Error Interrupt Information(00063C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIM7_CH7_EIRQ	TIM7_CH6_EIRQ	TIM7_CH5_EIRQ	TIM7_CH4_EIRQ	TIM7_CH3_EIRQ	TIM7_CH2_EIRQ	TIM7_CH1_EIRQ	TIM7_CH0_EIRQ	TIM6_CH7_EIRQ	TIM6_CH6_EIRQ	TIM6_CH5_EIRQ	TIM6_CH4_EIRQ	TIM6_CH3_EIRQ	TIM6_CH2_EIRQ	TIM6_CH1_EIRQ	TIM6_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM5_CH7_EIRQ	TIM5_CH6_EIRQ	TIM5_CH5_EIRQ	TIM5_CH4_EIRQ	TIM5_CH3_EIRQ	TIM5_CH2_EIRQ	TIM5_CH1_EIRQ	TIM5_CH0_EIRQ	TIM4_CH7_EIRQ	TIM4_CH6_EIRQ	TIM4_CH5_EIRQ	TIM4_CH4_EIRQ	TIM4_CH3_EIRQ	TIM4_CH2_EIRQ	TIM4_CH1_EIRQ	TIM4_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TIM4_CHx_EIRQ (x=0-7)	x	r	**TIM4 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
TIM5_CHx_EIRQ (x=0-7)	x+8	r	**TIM5 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
TIM6_CHx_EIRQ (x=0-7)	x+16	r	**TIM6 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
TIM7_CHx_EIRQ (x=0-7)	x+24	r	**TIM7 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module

28.22.5.17Register ICM_IRQG_CEI3

ICM Interrupt Group Register 3 for Channel Error Interrupt Information

ICM_IRQG_CEI3
ICM Interrupt Group Register 3 for Channel Error Interrupt Information(000640_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCS3_CH7_EIRQ	MCS3_CH6_EIRQ	MCS3_CH5_EIRQ	MCS3_CH4_EIRQ	MCS3_CH3_EIRQ	MCS3_CH2_EIRQ	MCS3_CH1_EIRQ	MCS3_CH0_EIRQ	MCS2_CH7_EIRQ	MCS2_CH6_EIRQ	MCS2_CH5_EIRQ	MCS2_CH4_EIRQ	MCS2_CH3_EIRQ	MCS2_CH2_EIRQ	MCS2_CH1_EIRQ	MCS2_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCS1_CH7_EIRQ	MCS1_CH6_EIRQ	MCS1_CH5_EIRQ	MCS1_CH4_EIRQ	MCS1_CH3_EIRQ	MCS1_CH2_EIRQ	MCS1_CH1_EIRQ	MCS1_CH0_EIRQ	MCS0_CH7_EIRQ	MCS0_CH6_EIRQ	MCS0_CH5_EIRQ	MCS0_CH4_EIRQ	MCS0_CH3_EIRQ	MCS0_CH2_EIRQ	MCS0_CH1_EIRQ	MCS0_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
MCS0_CHx_EIRQ (x=0-7)	x	r	**MCS0 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
MCS1_CHx_EIRQ (x=0-7)	x+8	r	**MCS1 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
MCS2_CHx_EIRQ (x=0-7)	x+16	r	**MCS2 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
MCS3_CHx_EIRQ (x=0-7)	x+24	r	**MCS3 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module

28.22.5.18Register ICM_IRQG_CEI4

ICM Interrupt Group Register 4 for Channel Error Interrupt Information

ICM_IRQG_CEI4
ICM Interrupt Group Register 4 for Channel Error Interrupt Information(000644_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCS7_CH7_EIRQ	MCS7_CH6_EIRQ	MCS7_CH5_EIRQ	MCS7_CH4_EIRQ	MCS7_CH3_EIRQ	MCS7_CH2_EIRQ	MCS7_CH1_EIRQ	MCS7_CH0_EIRQ	MCS6_CH7_EIRQ	MCS6_CH6_EIRQ	MCS6_CH5_EIRQ	MCS6_CH4_EIRQ	MCS6_CH3_EIRQ	MCS6_CH2_EIRQ	MCS6_CH1_EIRQ	MCS6_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCS5_CH7_EIRQ	MCS5_CH6_EIRQ	MCS5_CH5_EIRQ	MCS5_CH4_EIRQ	MCS5_CH3_EIRQ	MCS5_CH2_EIRQ	MCS5_CH1_EIRQ	MCS5_CH0_EIRQ	MCS4_CH7_EIRQ	MCS4_CH6_EIRQ	MCS4_CH5_EIRQ	MCS4_CH4_EIRQ	MCS4_CH3_EIRQ	MCS4_CH2_EIRQ	MCS4_CH1_EIRQ	MCS4_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
MCS4_CHx_EIRQ (x=0-7)	x	r	**MCS4 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
MCS5_CHx_EIRQ (x=0-7)	x+8	r	**MCS5 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
MCS6_CHx_EIRQ (x=0-7)	x+16	r	**MCS6 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
MCS7_CHx_EIRQ (x=0-7)	x+24	r	**MCS7 channel x error interrupt** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module

28.22.5.19Register ICM_IRQG_MCS[i]_CI

ICM Interrupt Group MCS i for Channel Interrupt Information

ICM_IRQG_MCSi_CI (i=0-9)
ICM Interrupt Group MCS i for Channel Interrupt Information(000720_H+i*4) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								MCS_CH7_IRQ	MCS_CH6_IRQ	MCS_CH5_IRQ	MCS_CH4_IRQ	MCS_CH3_IRQ	MCS_CH2_IRQ	MCS_CH1_IRQ	MCS_CH0_IRQ
r								r	r	r	r	r	r	r	r

Field	Bits	Type	Description
MCS_CHx_IRQ (x=0-7)	x	r	**MCS channel x interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.22.5.20Register ICM_IRQG_MCS[i]_CEI

ICM Interrupt Group MCS i for Channel Error Interrupt information

ICM_IRQG_MCSi_CEI (i=0-9)
ICM Interrupt Group MCS i for Channel Error Interrupt information(000664_H+i*4) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								MCS_CH7_EIRQ	MCS_CH6_EIRQ	MCS_CH5_EIRQ	MCS_CH4_EIRQ	MCS_CH3_EIRQ	MCS_CH2_EIRQ	MCS_CH1_EIRQ	MCS_CH0_EIRQ
r								r	r	r	r	r	r	r	r

Field	Bits	Type	Description
MCS_CHx_EIRQ (x=0-7)	x	r	**MCS channel x error interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
0	31:8	r	**Reserved** Read as zero, shall be written as zero.

28.22.5.21Register ICM_IRQG_SPE_CI

ICM Interrupt Group SPE for Module Interrupt Information

ICM_IRQG_SPE_CI

ICM Interrupt Group SPE for Module Interrupt Information(000770_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

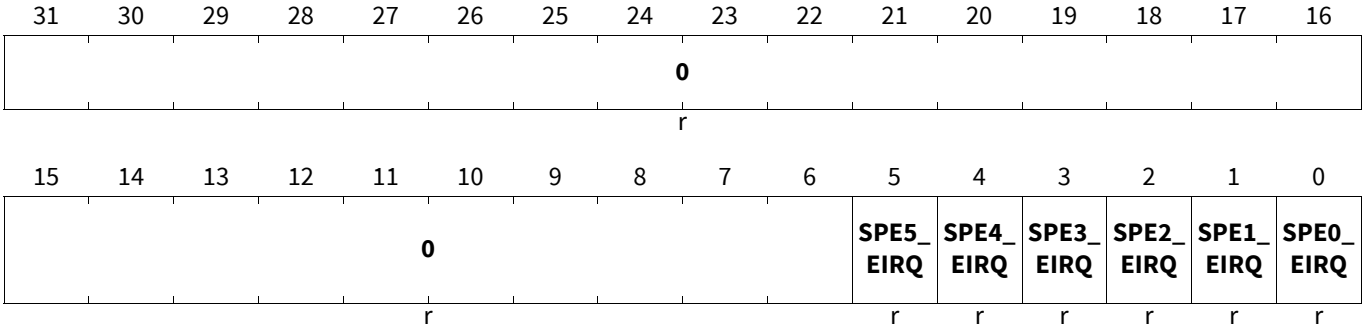
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										SPE5_IRQ	SPE4_IRQ	SPE3_IRQ	SPE2_IRQ	SPE1_IRQ	SPE0_IRQ
r										r	r	r	r	r	r

Field	Bits	Type	Description
SPEx_IRQ (x=0-5)	x	r	**SPE channel x interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
0	31:6	r	**Reserved** Read as zero, shall be written as zero

28.22.5.22Register ICM_IRQG_SPE_CEI

ICM Interrupt Group SPE for Module Error Interrupt Information

ICM_IRQG_SPE_CEI
ICM Interrupt Group SPE for Module Error Interrupt Information(0006B4$_H$) Application Reset Value: 0000 0000$_H$

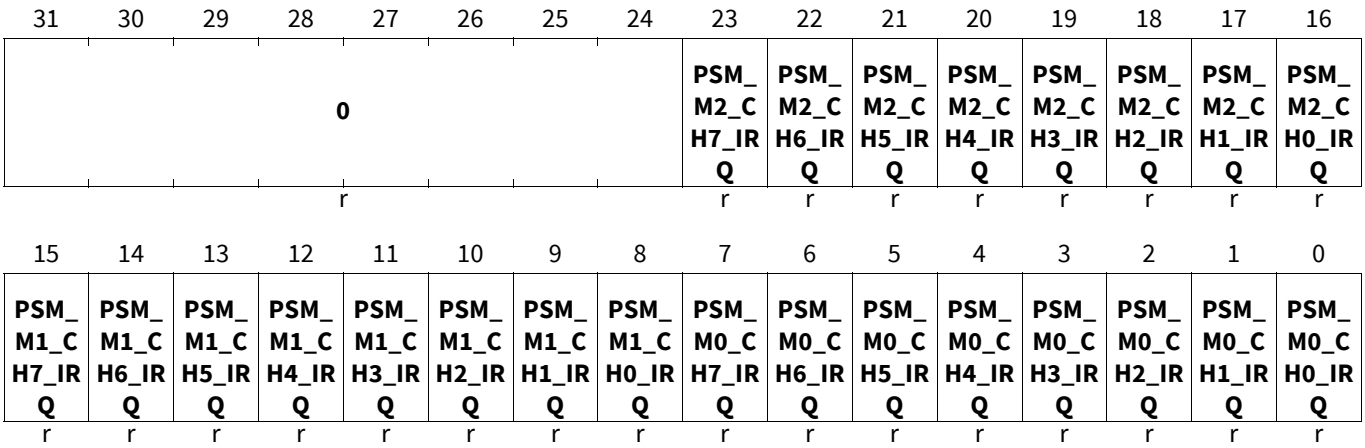


Field	Bits	Type	Description
SPEx_EIRQ (x=0-5)	x	r	**SPE channel x error interrupt** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
0	31:6	r	**Reserved** Read as zero, shall be written as zero.

28.22.5.23Register ICM_IRQG_PSM_0_CI

ICM Interrupt Group PSM 0 for Channel Interrupt Information of FIFO0, FIFO1, FIFO2

ICM_IRQG_PSM_0_CI
ICM Interrupt Group PSM 0 for Channel Interrupt Information of FIFO0, FIFO1, FIFO2(000760$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
PSM_M0_CHx_IRQ (x=0-7)	x	r	**PSM0 channel x shared interrupt (m=4*0+0)** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the four interrupt sources *FIFO_EMPTY*, *FIFO_FULL*, *FIFO_LOWER_WM* or *FIFO_UPPER_WM* of FIFO instance 0 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
PSM_M1_CHx_IRQ (x=0-7)	x+8	r	**PSM1 channel x shared interrupt (m=4*0+0)** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the four interrupt sources *FIFO_EMPTY*, *FIFO_FULL*, *FIFO_LOWER_WM* or *FIFO_UPPER_WM* of FIFO instance 1 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
PSM_M2_CHx_IRQ (x=0-7)	x+16	r	**PSM2 channel x shared interrupt (m=4*0+0)** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the four interrupt sources *FIFO_EMPTY*, *FIFO_FULL*, *FIFO_LOWER_WM* or *FIFO_UPPER_WM* of FIFO instance 2 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.22.5.24Register ICM_IRQG_PSM_0_CEI

ICM Interrupt Group PSM 0 for Channel Error Interrupt information of FIFO0, FIFO1, FIFO2

ICM_IRQG_PSM_0_CEI
ICM Interrupt Group PSM 0 for Channel Error Interrupt information of FIFO0, FIFO1, FIFO2($0006A4_H$)
Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								PSM_M2_CH7_EIRQ	PSM_M2_CH6_EIRQ	PSM_M2_CH5_EIRQ	PSM_M2_CH4_EIRQ	PSM_M2_CH3_EIRQ	PSM_M2_CH2_EIRQ	PSM_M2_CH1_EIRQ	PSM_M2_CH0_EIRQ
r								r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSM_M1_CH7_EIRQ	PSM_M1_CH6_EIRQ	PSM_M1_CH5_EIRQ	PSM_M1_CH4_EIRQ	PSM_M1_CH3_EIRQ	PSM_M1_CH2_EIRQ	PSM_M1_CH1_EIRQ	PSM_M1_CH0_EIRQ	PSM_M0_CH7_EIRQ	PSM_M0_CH6_EIRQ	PSM_M0_CH5_EIRQ	PSM_M0_CH4_EIRQ	PSM_M0_CH3_EIRQ	PSM_M0_CH2_EIRQ	PSM_M0_CH1_EIRQ	PSM_M0_CH0_EIRQ
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
PSM_M0_CHx_EIRQ (x=0-7)	x	r	**PSM0 channel x error interrupt (m=4*0+0)** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
PSM_M1_CHx_EIRQ (x=0-7)	x+8	r	**PSM1 channel x error interrupt (m=4*0+0)** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
PSM_M2_CHx_EIRQ (x=0-7)	x+16	r	**PSM2 channel x error interrupt (m=4*0+0)** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.22.5.25Register ICM_IRQG_TOM_[k]_CI

ICM Interrupt Group TOM k for Channel Interrupt Information of TOMm

ICM_IRQG_TOM_k_CI (k=0-2)
ICM Interrupt Group TOM k for Channel Interrupt Information of TOMm(0007A0$_H$+k*4) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOM_M1_CH15_IRQ	**TOM_M1_CH14_IRQ**	**TOM_M1_CH13_IRQ**	**TOM_M1_CH12_IRQ**	**TOM_M1_CH11_IRQ**	**TOM_M1_CH10_IRQ**	**TOM_M1_CH9_IRQ**	**TOM_M1_CH8_IRQ**	**TOM_M1_CH7_IRQ**	**TOM_M1_CH6_IRQ**	**TOM_M1_CH5_IRQ**	**TOM_M1_CH4_IRQ**	**TOM_M1_CH3_IRQ**	**TOM_M1_CH2_IRQ**	**TOM_M1_CH1_IRQ**	**TOM_M1_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOM_M0_CH15_IRQ	**TOM_M0_CH14_IRQ**	**TOM_M0_CH13_IRQ**	**TOM_M0_CH12_IRQ**	**TOM_M0_CH11_IRQ**	**TOM_M0_CH10_IRQ**	**TOM_M0_CH9_IRQ**	**TOM_M0_CH8_IRQ**	**TOM_M0_CH7_IRQ**	**TOM_M0_CH6_IRQ**	**TOM_M0_CH5_IRQ**	**TOM_M0_CH4_IRQ**	**TOM_M0_CH3_IRQ**	**TOM_M0_CH2_IRQ**	**TOM_M0_CH1_IRQ**	**TOM_M0_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TOM_M0_CHx_IRQ (x=0-15)	x	r	**TOMm channel x interrupt (m=2*k+0)** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 0 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
TOM_M1_CHx_IRQ (x=0-15)	x+16	r	**TOMm channel x interrupt (m=2*k+1)** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *TOM_CCU0TCx_IRQ* or *TOM_CCU1TCx_IRQ* of TOM instance 1 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.26Register ICM_IRQG_ATOM_[k]_CI

ICM Interrupt Group ATOM k for Channel Interrupt Information of ATOMm

ICM_IRQG_ATOM_k_CI (k=0-2)
ICM Interrupt Group ATOM k for Channel Interrupt Information of ATOMm(000790_H+k*4)Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATOM_M3_CH7_IRQ	**ATOM_M3_CH6_IRQ**	**ATOM_M3_CH5_IRQ**	**ATOM_M3_CH4_IRQ**	**ATOM_M3_CH3_IRQ**	**ATOM_M3_CH2_IRQ**	**ATOM_M3_CH1_IRQ**	**ATOM_M3_CH0_IRQ**	**ATOM_M2_CH7_IRQ**	**ATOM_M2_CH6_IRQ**	**ATOM_M2_CH5_IRQ**	**ATOM_M2_CH4_IRQ**	**ATOM_M2_CH3_IRQ**	**ATOM_M2_CH2_IRQ**	**ATOM_M2_CH1_IRQ**	**ATOM_M2_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOM_M1_CH7_IRQ	**ATOM_M1_CH6_IRQ**	**ATOM_M1_CH5_IRQ**	**ATOM_M1_CH4_IRQ**	**ATOM_M1_CH3_IRQ**	**ATOM_M1_CH2_IRQ**	**ATOM_M1_CH1_IRQ**	**ATOM_M1_CH0_IRQ**	**ATOM_M0_CH7_IRQ**	**ATOM_M0_CH6_IRQ**	**ATOM_M0_CH5_IRQ**	**ATOM_M0_CH4_IRQ**	**ATOM_M0_CH3_IRQ**	**ATOM_M0_CH2_IRQ**	**ATOM_M0_CH1_IRQ**	**ATOM_M0_CH0_IRQ**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
ATOM_M0_CHx_IRQ (x=0-7)	x	r	**ATOMm channel x interrupt (m=4*k+0)** This bit is only set, when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 0 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

Field	Bits	Type	Description
ATOM_M1_CHx_IRQ (x=0-7)	x+8	r	**ATOMm channel x interrupt (m=4*k+1)** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 1 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
ATOM_M2_CHx_IRQ (x=0-7)	x+16	r	**ATOMm channel x interrupt (m=4*k+2)** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 2 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module
ATOM_M3_CHx_IRQ (x=0-7)	x+24	r	**ATOMm channel x interrupt (m={4*k+3)** This bit is only set when the interrupt is enabled in the interrupt enable register of the corresponding sub-module. Set this bit represents an OR function of the two interrupt sources *CCU0TCx_IRQ* or *CCU1TCx_IRQ* of ATOM instance 2 channel x. 0_B No interrupt occurred 1_B Interrupt was raised by the corresponding sub-module

28.22.5.27Register ICM_IRQG_CLS_[k]_MEI

ICM Interrupt Group k for Module Error Interrupt Information for each TIMm, MCSm, SPEm, FIFOm

ICM_IRQG_CLS_k_MEI (k=0-2)
ICM Interrupt Group k for Module Error Interrupt Information for each TIMm, MCSm, SPEm, FIFOm
(000710_H+k*4) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				FIFO_M3_EIRQ	SPE_M3_EIRQ	MCS_M3_EIRQ	TIM_M3_EIRQ	0				FIFO_M2_EIRQ	SPE_M2_EIRQ	MCS_M2_EIRQ	TIM_M2_EIRQ
r				r	r	r	r	r				r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				FIFO_M1_EIRQ	SPE_M1_EIRQ	MCS_M1_EIRQ	TIM_M1_EIRQ	0				FIFO_M0_EIRQ	SPE_M0_EIRQ	MCS_M0_EIRQ	TIM_M0_EIRQ
r				r	r	r	r	r				r	r	r	r

Field	Bits	Type	Description
TIM_Mj_EIRQ (j=0-3)	8*j	r	**Error interrupt TIMm_EIRQ (m=4*k+j)** This bit is only set when the error interrupt is enabled in the error interrupt enable register of the corresponding sub-module. 0_B No error interrupt occurred 1_B Error interrupt was raised by the corresponding sub-module
MCS_Mj_EIRQ (j=0-3)	8*j+1	r	**Error interrupt MCSm_EIRQ (m=4*k+j)** Coding see bit 0.
SPE_Mj_EIRQ (j=0-3)	8*j+2	r	**Error interrupt SPEm_EIRQ (m=4*k+j)** Coding see bit 0.
FIFO_Mj_EIRQ (j=0-3)	8*j+3	r	**Error interrupt FIFOm_EIRQ (m=4*k+j)** Coding see bit 0.
0	31:28, 23:20, 15:12, 7:4	r	**Reserved** Read as zero, shall be written as zero.

28.23 Output Compare Unit (CMP)

28.23.1 Overview

The Output Compare Unit (CMP) is designed for the use in safety relevant applications (This module is not part of the Infineon safety manual). The main idea is to have the possibility to duplicate outputs in order to be compared in this unit. Because of the simple EXOR function used it is necessary to ensure the total cycle accurate output behavior of the output modules to be compared. This is given when two neighbored DTM channel (CDTM[n]_DTM[2*i] and CDTM[n]_DTM[2*i+1]) generate identical signals with phase shift zero at their outputs. This can be reached if they start their output generation at the same time. This start of synchronization is possible by means of the trigger mechanisms *TRIG_x* provided by the TOM or ATOM as shown in the TOM (chapter “Timer Output Module”) or ATOM (chapter “ARU-connected Timer Output Module”). It is not necessary to compare each output channel with each other.

The CMP enables the comparison of 2x24 channels of the CDTM0, CDTM1 and CDTM2 and is restricted to neighbored channels. The first 24 CMP channels are the first 24 DTM channels placed behind TOM0 and TOM1 and the second 24 CMP channels are the first 24 DTM channels placed behind the ATOM0, ATOM1 and ATOM2.

Note: When the channels were generated with a higher frequency than the frequency of cluster 1 it is not certain to catch the interrupt in the notify register. Avoid a comparison if freqency is unequal cluster 1 AND (cluster 0 OR cluster 2)

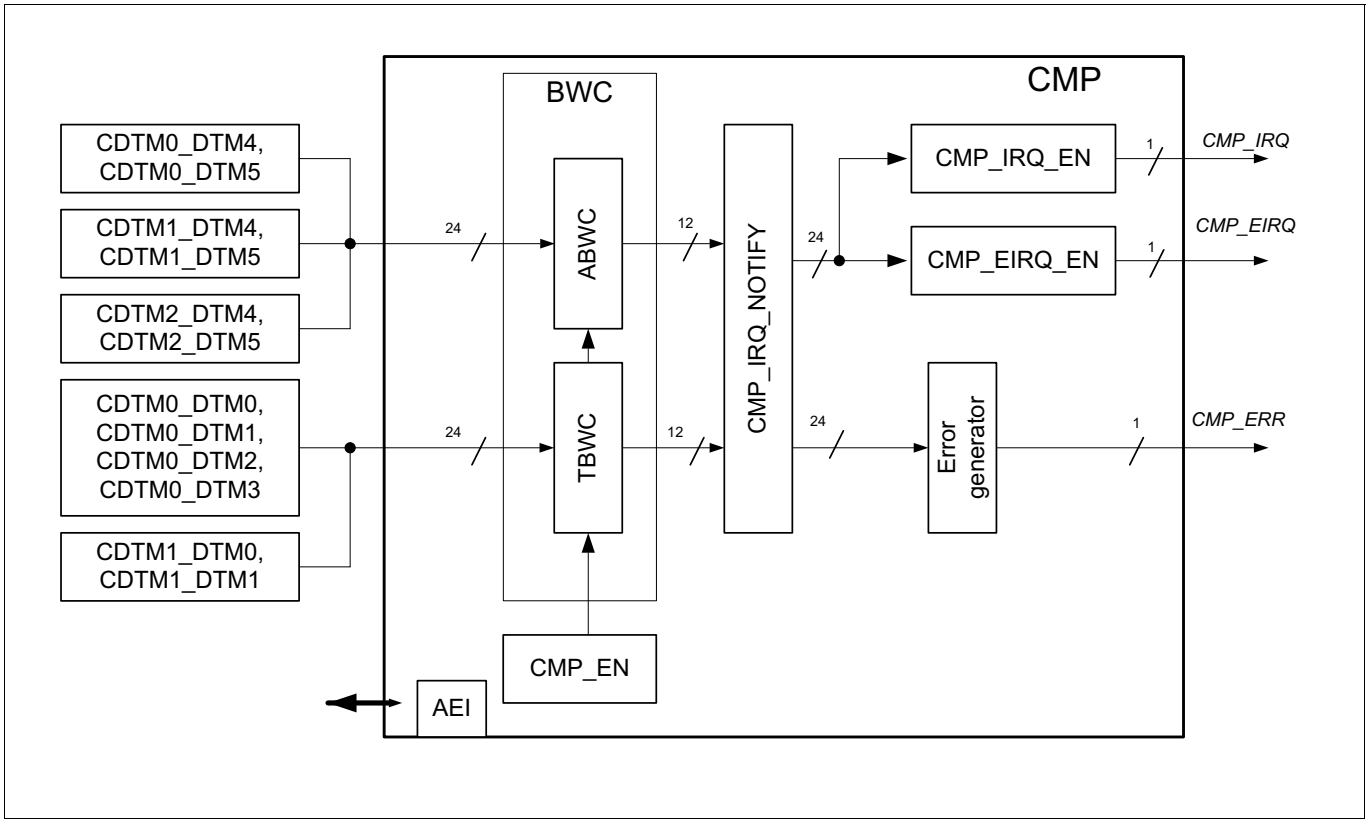


Figure 139 Architecture of the Compare Unit

28.23.2 Bitwise Compare Unit (BWC)

28.23.2.1 ABWC compare unit

Table 149 ABWC compare unit

ABWC Comparator	Comparator Input 1	Comparator Input 2
ABWC0	CDTM0_DTM4_CH0_OUT	CDTM0_DTM4_CH1_OUT
ABWC1	CDTM0_DTM4_CH2_OUT	CDTM0_DTM4_CH3_OUT
ABWC2	CDTM0_DTM5_CH0_OUT	CDTM0_DTM5_CH1_OUT
ABWC3	CDTM0_DTM5_CH2_OUT	CDTM0_DTM5_CH3_OUT
ABWC4	CDTM1_DTM4_CH0_OUT	CDTM1_DTM4_CH1_OUT
ABWC5	CDTM1_DTM4_CH2_OUT	CDTM1_DTM4_CH3_OUT
ABWC6	CDTM1_DTM5_CH0_OUT	CDTM1_DTM5_CH1_OUT
ABWC7	CDTM1_DTM5_CH2_OUT	CDTM1_DTM5_CH3_OUT
ABWC8	CDTM2_DTM4_CH0_OUT	CDTM2_DTM4_CH1_OUT
ABWC9	CDTM2_DTM4_CH2_OUT	CDTM2_DTM4_CH3_OUT
ABWC10	CDTM2_DTM5_CH0_OUT	CDTM2_DTM5_CH1_OUT
ABWC11	CDTM2_DTM5_CH2_OUT	CDTM2_DTM5_CH3_OUT

The Bitwise Compare Unit TBWC compares in pairs the combinations shown in following table

28.23.2.2 TBWC compare unit

Table 150 TBWC compare unit

TBWC Comparator	Comparator Input 1	Comparator Input 2
TBWC0	CDTM0_DTM0_CH0_OUT	CDTM0_DTM0_CH1_OUT
TBWC1	CDTM0_DTM0_CH2_OUT	CDTM0_DTM0_CH3_OUT
TBWC2	CDTM0_DTM1_CH0_OUT	CDTM0_DTM1_CH1_OUT
TBWC3	CDTM0_DTM1_CH2_OUT	CDTM0_DTM1_CH3_OUT
TBWC4	CDTM0_DTM2_CH0_OUT	CDTM0_DTM2_CH1_OUT
TBWC5	CDTM0_DTM2_CH2_OUT	CDTM0_DTM2_CH3_OUT
TBWC6	CDTM0_DTM3_CH0_OUT	CDTM0_DTM3_CH1_OUT
TBWC7	CDTM0_DTM3_CH2_OUT	CDTM0_DTM3_CH3_OUT
TBWC8	CDTM1_DTM0_CH0_OUT	CDTM1_DTM0_CH1_OUT
TBWC9	CDTM1_DTM0_CH2_OUT	CDTM1_DTM0_CH3_OUT
TBWC10	CDTM1_DTM1_CH0_OUT	CDTM1_DTM1_CH1_OUT
TBWC11	CDTM1_DTM1_CH2_OUT	CDTM1_DTM1_CH3_OUT

28.23.3 Configuration of the Compare Unit

Because of the restrictions described in the section above the Compare Unit consists of 24 antivalence (EXOR) elements, a select register **CMP_EN** which selects the corresponding comparisons and a status register **CMP_IRQ_NOTIFY** which shows and stores each mismatching result, when selected.

For each with **CMP_IRQ_EN** enabled mismatching error an interrupt signal on *CMP_IRQ* is generated.

For each with **CMP_EIRQ_EN** enabled mismatching error an interrupt signal on *CMP_EIRQ* is generated.

28.23.4 Error Generator

The error generator generates an error signal to be transmitted directly to the MON unit and independently from the *CMP_IRQ* and *CMP_EIRQ*. The error is set when in the **CMP_IRQ_NOTIFY** register at least one bit is set. The CMP_IRQ_NOTIFY bits are not mask able for this purpose.

Additionally *CMP_ERR* is a primary output port for interrupt actions by CPU itself.

28.23.5 CMP Interrupt Signal

Table 151 CMP Interrupt Signal

Signal	Description
CMP_EIRQ	Mismatching interrupt of outputs to be compared, when enabled
CMP_IRQ	Mismatching interrupt of outputs to be compared, when enabled

The CMP sub-module has two interrupt signals, one normal interrupt and one error interrupt. The source of both interrupt can be determined by reading the **CMP_IRQ_NOTIFY** register under consideration of **CMP_IRQ_EN** register and **CMP_EIRQ_EN** register. Each source can be forced separately for debug purposes using the interrupt force **CMP_IRQ_FORCINT** register. **CMP_IRQ_MODE** configures interrupt output characteristic. All interrupt modes are described in detail in the subparagraph “GTM-IP Interrupt Concept”

28.23.6 CMP Configuration Register Overview

Table 152 CMP Configuration Register Overview

Register Name	Description	see Page
CMP_EN	CMP comparator enable register	721
CMP_IRQ_NOTIFY	CMP event notification register	721
CMP_IRQ_EN	CMP interrupt enable register	722
CMP_IRQ_FORCINT	CMP interrupt force register	723
CMP_IRQ_MODE	CMP interrupt mode configuration register	723
CMP_EIRQ_EN	CMP error interrupt enable register	724

28.23.7 CMP Configuration Register Description

28.23.7.1 Register CMP_EN

CMP Comparator Enable Register

CMP_EN
CMP Comparator Enable Register **(000200_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TBWC 11_EN	TBWC 10_EN	TBWC 9_EN	TBWC 8_EN	TBWC 7_EN	TBWC 6_EN	TBWC 5_EN	TBWC 4_EN
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBWC 3_EN	TBWC 2_EN	TBWC 1_EN	TBWC 0_EN	ABWC 11_EN	ABWC 10_EN	ABWC 9_EN	ABWC 8_EN	ABWC 7_EN	ABWC 6_EN	ABWC 5_EN	ABWC 4_EN	ABWC 3_EN	ABWC 2_EN	ABWC 1_EN	ABWC 0_EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ABWCx_EN (x=0-11)	x	rw	**Enable comparator x in ABWC** see **Section 28.23.2** 0_B ABWC Comparator x is disabled 1_B ABWC Comparator x is enabled
TBWCx_EN (x=0-11)	x+12	rw	**Enable comparator x in TBWC** see **Section 28.23.2** 0_B TBWC comparator x is disabled 1_B TBWC comparator x is enabled
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.23.7.2 Register CMP_IRQ_NOTIFY

CMP Event Notification Register

CMP_IRQ_NOTIFY
CMP Event Notification Register **(000204_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TBWC 11	TBWC 10	TBWC 9	TBWC 8	TBWC 7	TBWC 6	TBWC 5	TBWC 4
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBWC 3	TBWC 2	TBWC 1	TBWC 0	ABWC 11	ABWC 10	ABWC 9	ABWC 8	ABWC 7	ABWC 6	ABWC 5	ABWC 4	ABWC 3	ABWC 2	ABWC 1	ABWC 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ABWCx (x=0-11)	x	rw	**Error indication for ABWCx** This bit will be cleared on a CPU write access of value '1'. (As the bit is rw, otherwise no clear.) A read access leaves the bit unchanged. 0_B No error recognized on DTMA sub-modules bits 0 and 1 (see **Section 28.23.2**) 1_B An error was recognized on corresponding DTMA sub-modules bits
TBWCx (x=0-11)	x+12	rw	**TOM sub-modules outputs bitwise comparator x error indication** This bit will be cleared on a CPU write access of value '1'. A read access leaves the bit unchanged. 0_B No error recognized on TOM sub-modules bits 0 and 1 (see **Section 28.23.2**) 1_B An error was recognized on corresponding TOM sub-modules bits
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.23.7.3 Register CMP_IRQ_EN

CMP Interrupt Enable Register

CMP_IRQ_EN
CMP Interrupt Enable Register **(000208$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TBWC 11_EN _IRQ	TBWC 10_EN _IRQ	TBWC 9_EN_ IRQ	TBWC 8_EN_ IRQ	TBWC 7_EN_ IRQ	TBWC 6_EN_ IRQ	TBWC 5_EN_ IRQ	TBWC 4_EN_ IRQ
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBWC 3_EN_ IRQ	TBWC 2_EN_ IRQ	TBWC 1_EN_ IRQ	TBWC 0_EN_ IRQ	ABWC 11_EN _IRQ	ABWC 10_EN _IRQ	ABWC 9_EN_ IRQ	ABWC 8_EN_ IRQ	ABWC 7_EN_ IRQ	ABWC 6_EN_ IRQ	ABWC 5_EN_ IRQ	ABWC 4_EN_ IRQ	ABWC 3_EN_ IRQ	ABWC 2_EN_ IRQ	ABWC 1_EN_ IRQ	ABWC 0_EN_ IRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ABWCx_EN_IRQ (x=0-11)	x	rw	**Enable ABWCx interrupt source for CMP_IRQ line** 0_B Interrupt source ABWCx is disabled 1_B Interrupt source ABWCx is enabled
TBWCx_EN_IRQ (x=0-11)	x+12	rw	**Enable TBWCx interrupt source for CMP_IRQ line** 0_B Interrupt source TBWCx is disabled 1_B Interrupt source TBWCx is enabled
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.23.7.4 Register CMP_IRQ_FORCINT

CMP Interrupt Force Register

CMP_IRQ_FORCINT
CMP Interrupt Force Register **($00020C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TRG_TBWC11	TRG_TBWC10	TRG_TBWC9	TRG_TBWC8	TRG_TBWC7	TRG_TBWC6	TRG_TBWC5	TRG_TBWC4
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRG_TBWC3	TRG_TBWC2	TRG_TBWC1	TRG_TBWC0	TRG_ABWC11	TRG_ABWC10	TRG_ABWC9	TRG_ABWC8	TRG_ABWC7	TRG_ABWC6	TRG_ABWC5	TRG_ABWC4	TRG_ABWC3	TRG_ABWC2	TRG_ABWC1	TRG_ABWC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TRG_ABWCx (x=0-11)	x	rw	**Trigger ABWCx bit in CMP_IRQ_NOTIFY register by software** This bit is cleared automatically after write. This bit is write protected by bit RF_PROT of register GTM_CTRL. 0_B No event triggering 1_B Assert corresponding field in CMP_IRQ_NOTIFY register
TRG_TBWCx (x=0-11)	x+12	rw	**Trigger TBWCx bit in CMP_IRQ_NOTIFY register by software** This bit is cleared automatically after write. 0_B No event triggering 1_B Assert corresponding field in CMP_IRQ_NOTIFY register
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.23.7.5 Register CMP_IRQ_MODE

CMP Interrupt Mode Configuration Register

CMP_IRQ_MODE
CMP Interrupt Mode Configuration Register **(000210_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														IRQ_MODE	
r														rw	

Field	Bits	Type	Description
IRQ_MODE	1:0	rw	**IRQ mode selection** The interrupt modes are described in **Section 28.22** 00_B Level mode 01_B Pulse mode 10_B Pulse-Notify mode 11_B Single-Pulse mode
0	31:2	r	**Reserved** Read as zero, shall be written as zero

28.23.7.6 Register CMP_EIRQ_EN

CMP error interrupt enable register

CMP_EIRQ_EN
CMP error interrupt enable register **(000214_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TBWC 11_EN _EIRQ	TBWC 10_EN _EIRQ	TBWC 9_EN_ EIRQ	TBWC 8_EN_ EIRQ	TBWC 7_EN_ EIRQ	TBWC 6_EN_ EIRQ	TBWC 5_EN_ EIRQ	TBWC 4_EN_ EIRQ
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBWC 3_EN_ EIRQ	TBWC 2_EN_ EIRQ	TBWC 1_EN_ EIRQ	TBWC 0_EN_ EIRQ	ABWC 11_EN _EIRQ	ABWC 10_EN _EIRQ	ABWC 9_EN_ EIRQ	ABWC 8_EN_ EIRQ	ABWC 7_EN_ EIRQ	ABWC 6_EN_ EIRQ	ABWC 5_EN_ EIRQ	ABWC 4_EN_ EIRQ	ABWC 3_EN_ EIRQ	ABWC 2_EN_ EIRQ	ABWC 1_EN_ EIRQ	ABWC 0_EN_ EIRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ABWCx_EN_EIRQ (x=0-11)	x	rw	**Enable ABWCx interrupt source for CMP_EIRQ line** 0_B Interrupt source ABWCx is disabled 1_B Interrupt source ABWCx is enabled
TBWCx_EN_EIRQ (x=0-11)	x+12	rw	**Enable TBWCx interrupt source for CMP_EIRQ line** 0_B Interrupt source TBWCx is disabled 1_B Interrupt source TBWCx is enabled
0	31:24	r	**Reserved** Read as zero, shall be written as zero.

28.24 Monitor Unit (MON)

28.24.1 Overview

The Monitor Unit (MON) is designed for the use in safety relevant applications (The MON module is not part of the Infineon Safety Manual). The main idea is to have a possibility to supervise common used circuitry and resources. In this way the activity of the clocks is supervised. In addition the characteristics of output signals can be checked in a MCS channel by a re-read-in via TIM and routing to the MCS. When the comparison fails an error signal is generated in MCS and sent to the monitor unit. One error signal per MCS summarizes the errors of all channels. By generating of an activity signal per channel for each such performed comparison, the activity of TIM, ARU and the used clocks is checked implicitly. In addition the ARU cycle time could be also compared in a MCS channel to given values.

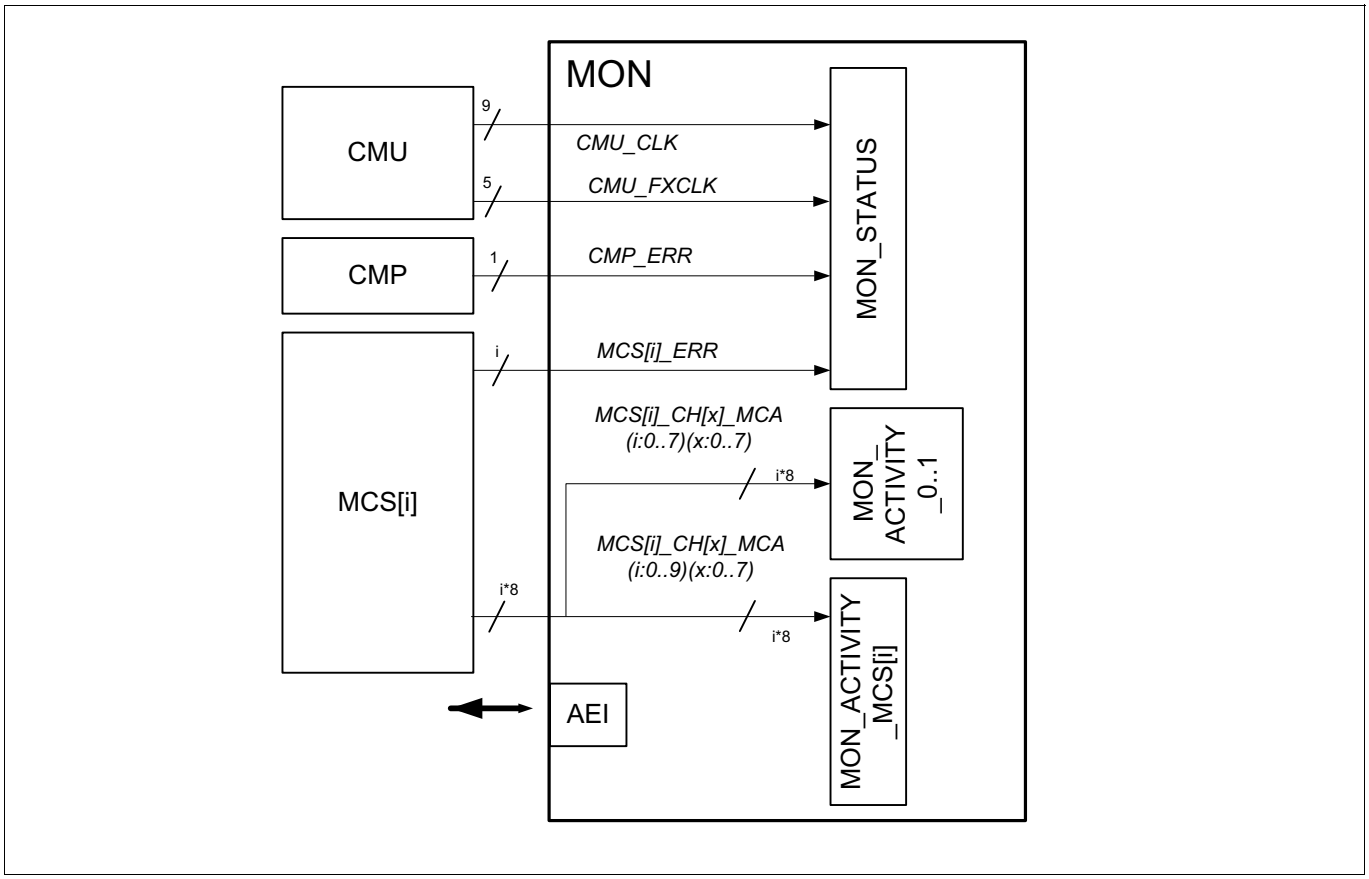


Figure 140 MON Block Diagram

28.24.1.1 Realization without Activity Checker of the clock signals

An activity checker of the clock signals used is not needed because these signals are only enables to be used in combination with the system clock. Therefore the clock enables are to be checked to have a high value.

28.24.2 Clock Monitoring

The monitor unit has a connection to each of the 9 clocks *CMU_CLK[x]* (x=0…8), provided by the CMU. Some of these clocks can be used for special tasks (see chapter “Clock Management Unit (CMU)”.

In addition the 5 clock inputs of the TOMs *CMU_FXCLK[y]* (y=0…4) are also connected to the MON unit.

The supervising of the clocks is done by scanning for activity of each clock.

A high value is defined as the state to be monitored.

When a high value of the clock enable is detected, the corresponding bit in the status register **MON_STATUS** is set.

The status register bits are reset by writing a one.

When the register is polled by the CPU and the time between two read accesses is higher than the period of the slowest clock, all bits of the corresponding clocks must have been set.

When polling in shorter time distances, not for all clocks an activity can be shown, although they are still working.

Because of the realization without a select register for the clock signals only the bits of the status register are to be considered for which the clock signal is enabled in the CMU.

28.24.3 CMP error Monitoring

The signal CMP_ERR is to be received directly from module CMP and is set if an error occurred.

28.24.4 Checking the Characteristics of Signals by MCS

By use of the MCS some given properties of signals can be checked. Such signals can be generated output signals of TOM or ATOM channels including DTM function, which are reread in into a TIM and the time stamp information is routed via ARU to the MCS module.

The corresponding MCS signal performs the check according to given properties. In this way signal high or low time as well as signal periods can be checked, also taking into account tolerances. When the check fails a MCS internal error signal is generated and ORed with the error signals of the other channels of the MCS module to a summarized error signal *MCS[i]_ERR*.

For each MCS a summarized error signal is transmitted to MON and monitored in the MON_STATUS register.

In order to check the execution of the comparison for each MCS channel an activity signal is generated. In the MCS[i]_CH[x]_MCA (i=0…9) (x=0…7) vector 8 bits for each MCS[i](i=0…9) instance are combined. The activity signals are stored in the **MON_ACTIVITY_MCS[i]** register. In addition the first 8 bits of MCS0…3 are stored in **MON_ACTIVITY_0** and the first 8 bits of MCS4…7 are stored in **MON_ACTIVITY_1**. The bits are set by a one signal and reset by writing a one to it (preferably after polling the status of the register).

Because the activity signal shows the execution of a comparison, the involved units for providing the signals and execution of comparison (like TIM, ARU and MCS itself) are checked implicitly to work accordingly. Also the involved clocks and time bases are checked in this way.

28.24.5 Checking ARU Cycle Time

The cycle time of the ARU can be checked, when this is essential for safety purposes. This check can be performed by an MCS channel. It should be noted that the MCS program for measuring the ARU round trip time must add a tolerance value.

The resulting error is reported to the MON unit using the summarized error signal *MCS[i]_ERR* for each MCS module in addition to an interrupt, generated in MCS. The same signals and status bits are used as in the case of checking the signal characteristics.

The corresponding MCS is programmed to get a fixed data value at address 0x1FF. The data value is always zero and is not blocked. When getting the access the time stamp value TBU_TS0 is stored in a register. The next time getting the access the new TBU_TS0 value is stored and the difference between both values is compared with a given value. When the comparison fails, an error flag is set in the MCS internal status register, an interrupt is generated and the error signal *MCS[i]_ERR* is provided.

When the check is performed, an activity signal MCS[i]_CH[x]_MCA (i=0…9)(x=0…7) is provided for each channel x for each MCS[i](i=0…9) instance together with a summarized interrupt MCS[i]_ERR for each MCS.

The activity signal sets a bit in the MON_ACTIVITY register.
The bits in the MON_ACTIVITY registers are reset by writing a one.
When the check fails, an interrupt is generated and the error signal MCS[i]_ERR is provided for the MON unit.
Figure 140 shows the block diagram of the Monitor Unit.

28.24.6 MON Interrupt Signals

The MON submodule has no interrupt signals.

28.24.7 MON Register Overview

Table 153 MON Register Overview

Register Name	Description	see Page
MON_STATUS	Monitor status register	728
MON_ACTIVITY_0	Monitor activity register 0	729
MON_ACTIVITY_1	Monitor activity register 1	729
MON_ACTIVITY_MCS[z]	Monitor activity register for MCS z	730

28.24.8 MON Configuration Register Description

28.24.8.1 Register MON_STATUS

Monitor Status Register

The MCS can be programmed to generate an error, when the comparison of signal values (duty time, cycle time) fails or also when the cycle time of the ARU (checking of the TBU_TS0 between two periodic accesses) is out of the expected range.

MON_STATUS
Monitor Status Register **(000180$_H$)** **Application Reset Value: 0000 4000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		MCS9_ERR	MCS8_ERR	MCS7_ERR	MCS6_ERR	MCS5_ERR	MCS4_ERR	MCS3_ERR	MCS2_ERR	MCS1_ERR	MCS0_ERR	0			CMP_ERR
r		r	r	r	r	r	r	r	r	r	r	r			r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ACT_CMU8	0	ACT_CMUFX4	ACT_CMUFX3	ACT_CMUFX2	ACT_CMUFX1	ACT_CMUFX0	ACT_CMU7	ACT_CMU6	ACT_CMU5	ACT_CMU4	ACT_CMU3	ACT_CMU2	ACT_CMU1	ACT_CMU0
r	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ACT_CMUx (x=0-7)	x	rw	**CMU_CLKx activity** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged. Bits is set, when a rising edge is detected at the considered clock.
ACT_CMUFXx (x=0-4)	x+8	rw	**CMU_CLKFXx activity** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged. Bits is set, when a rising edge is detected at the considered clock.
ACT_CMU8	14	rw	**CMU_CLK8 activity** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged. Bit is set, when a rising edge is detected at the considered clock.
CMP_ERR	16	r	**Error detected at CMP** This bit will be readable only. Bits is set, when the corresponding unit reports an error.
MCSx_ERR (x=0-9)	x+20	r	**Error detected at MCSx** This bit will be readable only. Bits is set, when the corresponding unit reports an error.
0	13, 15, 19:17, 31:30	r	**Reserved** Read as zero, shall be written as zero.

28.24.8.2 Register MON_ACTIVITY_0

Monitor Activity Register 0

When not all MCS modules are implemented or the channels are not used for check purposes with supervising, the corresponding activity bits remain zero.

MON_ACTIVITY_0
Monitor Activity Register 0 **(000184_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCA_3_7	MCA_3_6	MCA_3_5	MCA_3_4	MCA_3_3	MCA_3_2	MCA_3_1	MCA_3_0	MCA_2_7	MCA_2_6	MCA_2_5	MCA_2_4	MCA_2_3	MCA_2_2	MCA_2_1	MCA_2_0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCA_1_7	MCA_1_6	MCA_1_5	MCA_1_4	MCA_1_3	MCA_1_2	MCA_1_1	MCA_1_0	MCA_0_7	MCA_0_6	MCA_0_5	MCA_0_4	MCA_0_3	MCA_0_2	MCA_0_1	MCA_0_0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MCA_0_x (x=0-7)	x	rw	**Activity of check performed in module MCS 0 at channel x** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.
MCA_1_x (x=0-7)	x+8	rw	**Activity of check performed in module MCS 1 at channel x** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.
MCA_2_x (x=0-7)	x+16	rw	**Activity of check performed in module MCS 2 at channel x** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.
MCA_3_x (x=0-7)	x+24	rw	**Activity of check performed in module MCS 3 at channel x** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

28.24.8.3 Register MON_ACTIVITY_1

Monitor Activity Register 1

Note: When not all MCS modules are implemented or the channels are not used for check purposes with supervising, the corresponding activity bits remain zero.

MON_ACTIVITY_1
Monitor Activity Register 1 **(000188$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCA_7_7	MCA_7_6	MCA_7_5	MCA_7_4	MCA_7_3	MCA_7_2	MCA_7_1	MCA_7_0	MCA_6_7	MCA_6_6	MCA_6_5	MCA_6_4	MCA_6_3	MCA_6_2	MCA_6_1	MCA_6_0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCA_5_7	MCA_5_6	MCA_5_5	MCA_5_4	MCA_5_3	MCA_5_2	MCA_5_1	MCA_5_0	MCA_4_7	MCA_4_6	MCA_4_5	MCA_4_4	MCA_4_3	MCA_4_2	MCA_4_1	MCA_4_0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MCA_4_x (x=0-7)	x	rw	**Activity of check performed in module MCS 4 at channel x** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.
MCA_5_x (x=0-7)	x+8	rw	**Activity of check performed in module MCS 5 at channel x** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.
MCA_6_x (x=0-7)	x+16	rw	**Activity of check performed in module MCS 6 at channel x** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.
MCA_7_x (x=0-7)	x+24	rw	**Activity of check performed in module MCS 7 at channel x** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

28.24.8.4 Register MON_ACTIVITY_MCS[z]

Monitor Activity Register for MCS z

MON_ACTIVITY_MCSz (z=0-9)
Monitor Activity Register for MCS z **(00018C$_H$+z*4)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								MCA_7	MCA_6	MCA_5	MCA_4	MCA_3	MCA_2	MCA_1	MCA_0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MCA_x (x=0-7)	x	rw	**Activity of check performed in module MCS[z] at channel x** This bit will be cleared on a CPU write access of value 1. A read access leaves the bit unchanged.

Field	Bits	Type	Description
0	31:8	r	**Reserved** Read as zero, shall be written as zero

28.25 GTM Implementation

28.25.1 AURIX TC3xx Family GTM Configuration

Table 154 GTM Configuration by AURIX TC3xx Product

GTM Modules	TC39x	TC38x	TC37x	TC36x	TC35x (No GTM)	TC33x
TIM	8x8 ch. (TIM0-7)	7x8 ch. (TIM0-6)	6x8 ch. (TIM0-5)	3x8 ch. (TIM0-2)	0	2x8 ch. (TIM0-1)
TOM	6x16 ch. (TOM0-5)	5x16 ch. (TOM0-4)	3x16 ch. (TOM0-2)	2x16 ch. (TOM0-1)	0	2x16 ch. (TOM0-1)
ATOM	12x8 ch. (ATOM0-11)	9x8 ch. (ATOM0-8)	6x8 ch. (ATOM0-5)	4x8 ch. (ATOM0-3)	0	1x8 ch. (ATOM0)
DTM/CDTM	5x4 ch., 2x2 ch. 7x CDTM (**Table 156**)	4x4 ch., 2x2 ch. 6x CDTM (**Table 156**)	3x4 ch., 2x2 ch. 5x CDTM (**Table 156**)	2x4 ch., 2x2 ch. 4x CDTM (**Table 156**)	0	1x4 ch., 1x2 ch. 2x CDTM (**Table 156**)
MCS	10x8 ch. (MCS0-9)	7x8 ch. (MCS0-6)	5x8 ch. (MCS0-4)	3x8 ch. (MCS0-2)	0	0
SPE	6 (SPE0-5)	4 (SPE0-3)	2 (SPE0-1)	2 (SPE0-1)	0	2 (SPE0-1)
PSM	3 (PSM0-2)	2 (PSM0-1)	1 (PSM0)	1 (PSM0)	0	0
DPLL	1	1	1	1	0	0
TBU	4 (TBU0-3)	4 (TBU0-3)	4 (TBU0-3)	4 (TBU0-3)	0	3 (TBU0-2)
BRC	1	1	1	1	0	1
MON	1	1	1	1	0	1
CMP	1	1	1	1	0	1
GTM Clusters (max speed)	12 (CCM0-11) CCM0-4: 200 MHz max CCM5-11: 100 MHz max	9 (CCM0-8) CCM0-4: 200 MHz max CCM5-8: 100 MHz max	6 (CCM0-5) CCM0-4: 200 MHz max CCM5: 100 MHz max	4 (CCM0-3) CCM0-3: 200 MHz max	0	2 (CCM0-1) CCM0-1: 200 MHz max

Table 155 DTM/CDTM Labels Legacy

GTM v2.0 ... V3.0	GTM v3.1			
DTM[i]	CDTM[n]	CDTM[n]_DTM[j]	TOM/ATOM	Note
0	0	0	TOM0	
1	0	1	TOM0	
2	0	2	TOM0	Not used
3	0	3	TOM0	Not used
4	1	0	TOM1	
5	1	1	TOM1	
6	1	2	TOM1	Not used
7	1	3	TOM1	Not used
8	2	0	TOM2	
9	2	1	TOM2	
10	2	2	TOM2	Not used
11	2	3	TOM2	Not used
12	3	0	TOM3	
13	3	1	TOM3	
14	3	2	TOM3	Not used
15	3	3	TOM3	Not used
16	4	0	TOM4	
17	4	1	TOM4	
18	4	2	TOM4	Not used
19	4	3	TOM4	Not used
20	5	0	TOM5	Not used
21	5	1	TOM5	Not used
22	5	2	TOM5	Not used
23	5	3	TOM5	Not used
24	0	4	ATOM0	
25	0	5	ATOM0	
26	1	4	ATOM1	
27	1	5	ATOM1	
28	2	4	ATOM2	
29	2	5	ATOM2	
30	3	4	ATOM3	
31	3	5	ATOM3	
32	4	4	ATOM4	
33	4	5	ATOM4	
34	5	4	ATOM5	
35	5	5	ATOM5	

Table 155 DTM/CDTM Labels Legacy (cont'd)

GTM v2.0 ... V3.0	GTM v3.1			
36	6	4	ATOM6	
37	6	5	ATOM6	
38	7	4	ATOM7	Not used
39	7	5	ATOM7	Not used
40	8	4	ATOM8	Not used
41	8	5	ATOM8	Not used
42	9	4	ATOM9	Not used
43	9	5	ATOM9	Not used
44	10	4	ATOM10	Not used
45	10	5	ATOM10	Not used
46	11	4	ATOM11	Not used
47	11	5	ATOM11	Not used

Table 156 CDTM Connections by AURIX TC3xx Product

GTM Modules	TC39x	TC38x	TC37x	TC36x	TC35x	TC33x
TOM0_CH0..CH3	CDTM0_DTM0	CDTM0_DTM0	CDTM0_DTM0	CDTM0_DTM0	0	CDTM0_DTM0
TOM0_CH4..CH7	CDTM0_DTM1	CDTM0_DTM1	CDTM0_DTM1	CDTM0_DTM1	0	CDTM0_DTM1
TOM1_CH0..CH3	CDTM1_DTM0	CDTM1_DTM0	CDTM1_DTM0	CDTM1_DTM0	0	CDTM1_DTM0
TOM1_CH4..CH7	CDTM1_DTM1	CDTM1_DTM1	CDTM1_DTM1	CDTM1_DTM1	0	CDTM1_DTM1
TOM2_CH0..CH3	CDTM2_DTM0	CDTM2_DTM0	CDTM2_DTM0	-	0	-
TOM2_CH4..CH7	CDTM2_DTM1	CDTM2_DTM1	CDTM2_DTM1	-	0	-
TOM3_CH0..CH3	CDTM3_DTM0	CDTM3_DTM0	-	-	0	-
TOM3_CH4..CH7	CDTM3_DTM1	CDTM3_DTM1	-	-	0	-
TOM4_CH0..CH3	CDTM4_DTM0		-	-	0	-
TOM4_CH4..CH7	CDTM4_DTM1		-	-	0	-
ATOM0_CH0..CH3	CDTM0_DTM4	CDTM0_DTM4	CDTM0_DTM4	CDTM0_DTM4	0	CDTM0_DTM4
ATOM0_CH4..CH7	CDTM0_DTM5	CDTM0_DTM5	CDTM0_DTM5	CDTM0_DTM5	0	CDTM0_DTM5
ATOM1_CH0..CH3	CDTM1_DTM4	CDTM1_DTM4	CDTM1_DTM4	CDTM1_DTM4	0	
ATOM1_CH4..CH7	CDTM1_DTM5	CDTM1_DTM5	CDTM1_DTM5	CDTM1_DTM5	0	
ATOM2_CH0..CH3	CDTM2_DTM4	CDTM2_DTM4	CDTM2_DTM4	CDTM2_DTM4	0	
ATOM2_CH4..CH7	CDTM2_DTM5	CDTM2_DTM5	CDTM2_DTM5	CDTM2_DTM5	0	
ATOM3_CH0..CH3	CDTM3_DTM4	CDTM3_DTM4	CDTM3_DTM4	CDTM3_DTM4	0	-
ATOM3_CH4..CH7	CDTM3_DTM5	CDTM3_DTM5	CDTM3_DTM5	CDTM3_DTM5	0	-
ATOM4_CH0..CH3	CDTM4_DTM4	CDTM4_DTM4	CDTM4_DTM4	-	0	-
ATOM4_CH4..CH7	CDTM4_DTM5	CDTM4_DTM5	CDTM4_DTM5	-	0	-
ATOM5_CH0..CH3	CDTM5_DTM4	CDTM5_DTM4	-	-	0	-
ATOM5_CH4..CH7	CDTM5_DTM5	CDTM5_DTM5	-	-	0	-
ATOM6_CH0..CH3	CDTM6_DTM4	-	-	-	0	-

Table 156 CDTM Connections by AURIX TC3xx Product (cont'd)

GTM Modules	TC39x	TC38x	TC37x	TC36x	TC35x	TC33x
ATOM6_CH4..CH7	CDTM6_DTM5	-	-	-	0	-
Total DTM Instances	24	20	16	12	0	6

28.25.2 GTM Memories Address Map

GTM Memory Addresses

Table 157 GTM Memories

Memory	Words	Word Size	Start Address Offset	End Address Offset	Read Access Mode	Write Access Mode
PSM0 RAM	1024	29 Bit	0x00019000	0x00019FFC	U, SV	U, SV, P
PSM1 RAM	1024	29 Bit	0x0001D000	0x0001DFFC	U, SV	U, SV, P
PSM2 RAM	1024	29 Bit	0x00021000	0x00021FFC	U, SV	U, SV, P
DPLL RAM1A	128	24 Bit	0x00028200	0x000283FC	U, SV	U, SV, P
DPLL RAM1B	128	24 Bit	0x00028400	0x000285FC	U, SV	U, SV, P
DPLL RAM1C	256	24 Bit	0x00028600	0x000289FC	U, SV	U, SV, P
DPLL RAM2	4096	24 Bit	0x0002C000	0x0002FFFC	U, SV	U, SV, P
MCS0 RAM0/1	3072	32 Bit	0x00038000	0x0003AFFC	U, SV	U, SV, P
MCS1 RAM0/1	3072	32 Bit	0x00040000	0x00042FFC	U, SV	U, SV, P
MCS2 RAM0/1	3072	32 Bit	0x00048000	0x0004AFFC	U, SV	U, SV, P
MCS3 RAM0/1	3072	32 Bit	0x00050000	0x00052FFC	U, SV	U, SV, P
MCS4 RAM0/1	3072	32 Bit	0x00058000	0x0005AFFC	U, SV	U, SV, P
MCS5 RAM0/1	3072	32 Bit	0x00060000	0x00062FFC	U, SV	U, SV, P
MCS6 RAM0/1	3072	32 Bit	0x00068000	0x0006AFFC	U, SV	U, SV, P
MCS7 RAM0/1	3072	32 Bit	0x00070000	0x00072FFC	U, SV	U, SV, P
MCS8 RAM0/1	3072	32 Bit	0x00078000	0x0007AFFC	U, SV	U, SV, P
MCS9 RAM0/1	3072	32 Bit	0x00080000	0x00082FFC	U, SV	U, SV, P

Note: For the MCS, all the RAM0/1 memories are "FAST" (200 MHz). Internally, only four memories are instantiated and used by all the MCS.

ATOM Bit-Reversed Mode (PCM)

Bit-reversed mode (PCM) in ATOM SOMP is available in the following channels:

Table 158 Available PCM in ATOM SOMP mode

Module	CH0	CH1	CH2	CH3	CH4	CH5	CH6	CH7
ATOM0	no	yes	no	yes	no	yes	no	yes
ATOM1	no	yes	no	yes	no	yes	no	yes
ATOM2	no	yes	no	yes	no	yes	no	yes

Table 158 Available PCM in ATOM SOMP mode (cont'd)

Module	CH0	CH1	CH2	CH3	CH4	CH5	CH6	CH7
ATOM3	no	yes	no	yes	no	yes	no	yes
ATOM4	no	yes	no	yes	no	yes	no	yes
ATOM5	no	yes	no	yes	no	yes	no	yes
ATOM6	no	yes	no	yes	no	yes	no	yes
ATOM7	no	yes	no	yes	no	yes	no	yes
ATOM8	no	yes	no	yes	no	yes	no	yes
ATOM9	no	yes	no	yes	no	yes	no	yes
ATOM10	no	yes	no	yes	no	yes	no	yes
ATOM11	no	yes	no	yes	no	yes	no	yes

28.25.3 GTM Interrupts

In order to trigger an interrupt towards the Interrupt Router, the GTM has to generate a trigger signal with a minimum high length of one SPB clock cycle.

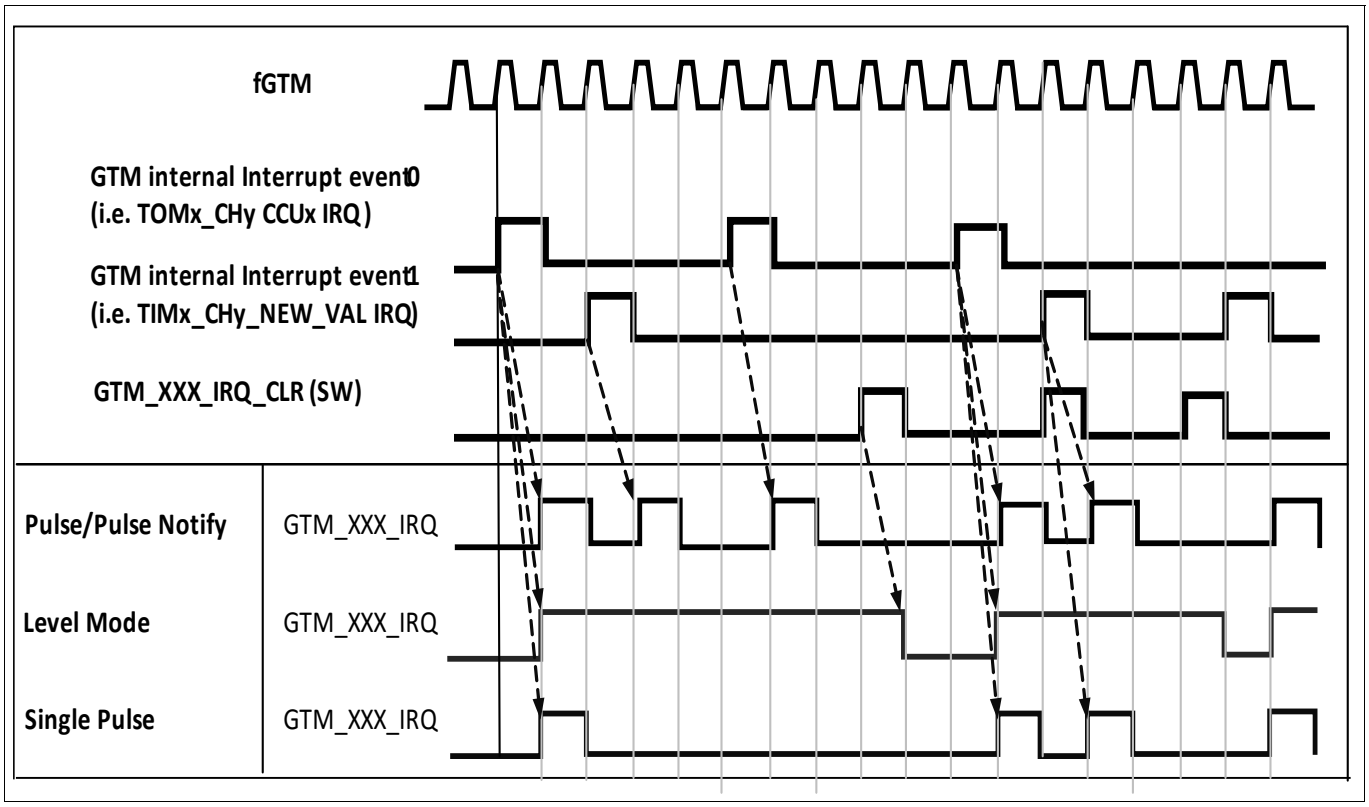


Figure 141 GTM Interrupt generation

After the interrupt ACK has been generated, the SRNs are cleared automatically on the Interrupt Router side, but this signal is not propagated to GTM, therefore to clear an interrupt on the GTM side, a SW approach is needed.

The GTM interrupts can be configured to work in four different modes: Level, Pulse,Pulse-Notify,Single-Pulse.

These modes change the way the GTM triggers are generated toward the Interrupt Router (IR).

Depending on the applications, one mode or the other could be used.

The **Figure 142** shows how the interrupt signal (GTM_IRQ_XXX) triggers an interrupt towards the IR, depending on IRQ_MODE.

From the **Figure 142** it is possible to understand that, using the Level Mode (Infineon default), in case that more internal “interrupt” events are generated (i.e. two TOM channels generating a CCU0 interrupt), just one interrupt signal is sent to the Interrupt Router, and no more interrupts are triggered until the SW clears all the “collected” interrupts.

In case that the Pulse-Notify mode is selected, every internal trigger, will be forwarded to the Interrupt Router.

For more details about the GTM interrupt modes, please refer to the GTM Interrupt concept chapter.

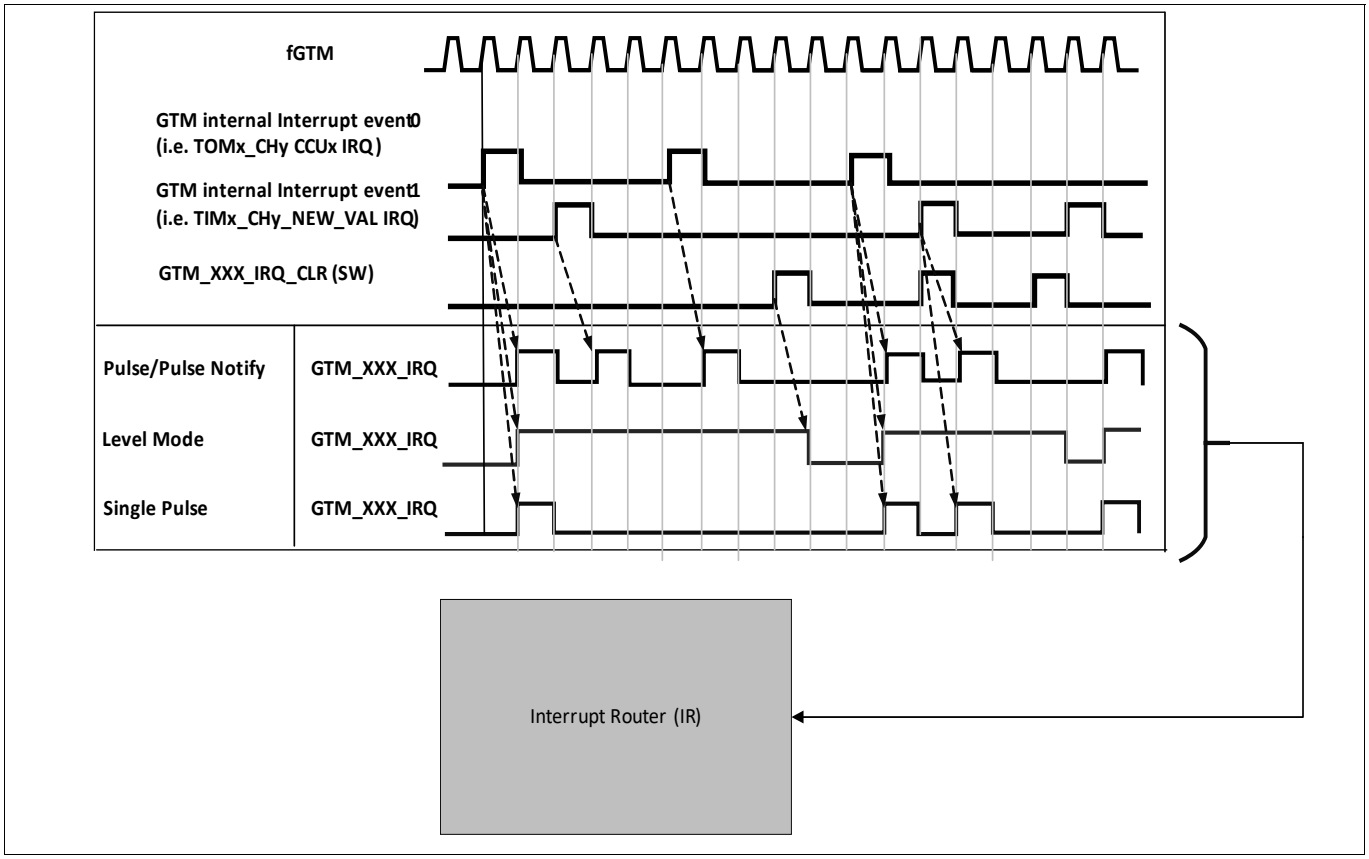


Figure 142 GTM Interrupts Modes

28.25.4 GTM Bridge

Within the infineon specific GTM wrapper, an interface (bridge) has been introduced to convert between the SPB bus protocol and the AEI protocol of the GTM IP in both directions. This introduces an additional delay of one SPB clock cycle per direction. In addition to that, additional pipeline registers have been implemented in the GTM IP to reach the required speed of the logic (200 MHz). Both contribute to an additional latency for read and write accesses towards the GTM IP registers and memories.

28.25.5 GTM Control Registers

This section lists all the Infineon specific registers implemented in the AURIX TC3x.

Table 159 Register Overview - IFX (sorted by Name)

Short Name	Long Name	Offset Address	Page Number
ACCEN0	Access Enable Register 0	$09FD10_H$	741
ACCEN1	Access Enable Register 1	$09FD14_H$	741
ADCTRIGiOUT0	ADC Trigger i Output Select 0 Register	$09FE40_H$+i*8	759
ADCTRIGiOUT1	ADC Trigger i Output Select 1 Register	$09FE44_H$+i*8	759
CANOUTSEL0	CAN0/CAN1 Output Select Register	$09FFDC_H$	764
CANOUTSEL1	CAN2 Output Select Register	$09FFE0_H$	764
CLC	Clock Control Register	$09FD00_H$	739
DATAINn	Data Input n Register	$09FED4_H$+n*4	776
DSADCINSELi	DSADC Input Select i Register	$09FE00_H$+i*4	757
DSADCOUTSELi0	DSADC Output Select i0 Register	$09FE20_H$+i*8	757
DSADCOUTSELi1	DSADC Output Select i1 Register	$09FE24_H$+i*8	758
DTMAUXINSEL	DTM_AUX Input Selection Register	$09FFD8_H$	749
DXINCON	Data Exchange Input Control Register	$09FED0_H$	775
DXOUTCON	Data Exchange Output Control Register	$09FE70_H$	771
INTOUTn	Interrupt Output Register n	$09FE9C_H$+n*4	772
LCDCDCOUTSEL	LCDCDC Output Select Register	$09FFD4_H$	766
MCSINTCLR	MCS Interrupt Clear Register	$09FECC_H$	774
MCSINTSTAT	MCS Interrupt Status Register	$09FEC8_H$	773
MCSTRIGOUTSEL	Trigger Output Select Register	$09FEC4_H$	772
MSCSETiCONj	MSC Set i Control j Register	$09FF00_H$+i*10_H +j*4	752
MSCiINHCON	MSCi Input High Control Register	$09FF94_H$+i*12	754
MSCiINLCON	MSCi Input Low Control Register	$09FF90_H$+i*12	754
MSCiINLEXTCON	MSCi Input Low Extended Control Register	$09FF98_H$+i*12	754
OCS	OCDS Control and Status	$09FD38_H$	739
ODA	OCDS Debug Access Register	$09FD34_H$	788
OTBU0T	OCDS TBU0 Trigger Register	$09FD18_H$	789
OTBU1T	OCDS TBU1 Trigger Register	$09FD1C_H$	790
OTBU2T	OCDS TBU2 Trigger Register	$09FD20_H$	790
OTBU3T	OCDS TBU3 Trigger Register	$09FD24_H$	791
OTSC0	OCDS Trigger Set Control 0 Register	$09FD2C_H$	785
OTSC1	OCDS Trigger Set Control 1 Register	$09FD30_H$	787
OTSS	OCDS Trigger Set Select Register	$09FD28_H$	784
PSI5OUTSEL	PSI5 Output Select Register	$09FFCC_H$	769

Table 159 Register Overview - IFX (sorted by Name) (cont'd)

Short Name	Long Name	Offset Address	Page Number
PSI5SOUTSEL	PSI5-S Output Select Register	$09FFD0_H$	769
RESET1	Kernel Reset Register 0	$09FD08_H$	742
RESET2	Kernel Reset Register 1	$09FD0C_H$	743
RESET_CLR	Kernel Reset Status Clear Register	$09FD04_H$	743
TIMnINSEL	TIMn Input Select Register	$09FD40_H$+n*4	747
TOUTSELn	Timer Output Select Register	$09FD60_H$+n*4	746
TRIGOUTn	Trigger Output Register n	$09FE74_H$+n*4	771

Clock Control Register

The clock control register is used to switch the GTM on or off, and to control its input clock rate. The GTM can be disabled by setting bit DISR to 1.

CLC
Clock Control Register **($09FD00_H$)** **Application Reset Value: $0000\ 0003_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	r	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the GTM module. 0_B No disable requested 1_B Disable requested
DISS	1	r	**Module Disable Status Bit** This bit indicates the current status of the GTM module. 0_B GTM module is enabled 1_B GTM module is disabled
EDIS	3	rw	**Sleep Mode Enable Control** Used for module's sleep mode control. 0_B Enabled 1_B Disabled
0	2, 31:4	r	**Reserved** Read as 0, shall be written with 0.

OCDS Control and Status

Note: OCS.SUSSTA is set to 1 (module is suspended) up to 2 gtm_clk cycles earlier before GTM changes into soft suspend mode.

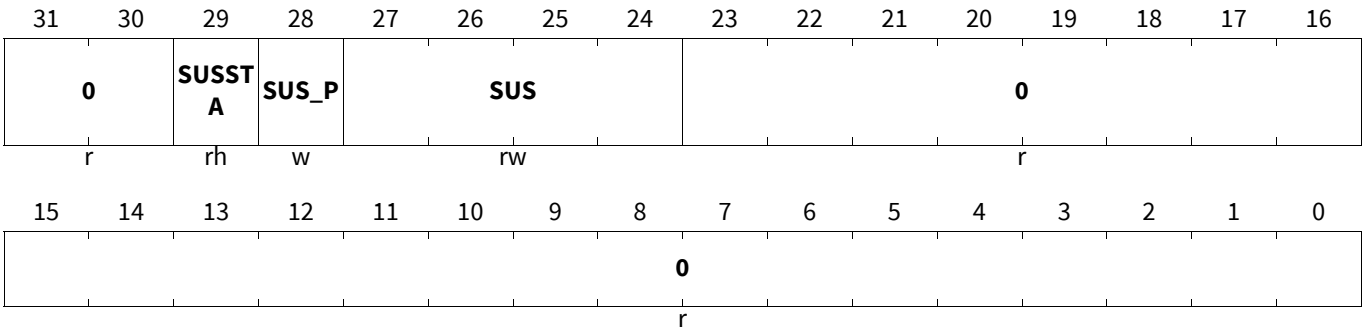
The OCDS control and status register OCS controls the module's behavior in suspend mode (used for debugging). The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The OCS register includes the module related control bits for the OCDS Trigger Bus (OTGB).

The register can only be written and the OCS control register bits are only effective while the OCDS is enabled (OCDS enable ='1'). While OCDS is not enabled, OCS reset values/modes are effective.

OCS

OCDS Control and Status (09FD38$_H$) Debug Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. In hard suspend mode, no registers could be read or written. The GTM control registers are not affected, except all the debug registers apart from OCS register, and the TRIGOUT* and INTOUT* registers, which cannot be written as well. 2_H Soft suspend (GTM Halt Mode). In soft suspend mode, registers could be read or written. **others**, Reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise it remains unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended Module is suspended
0	23:0, 31:30	r	**Reserved** Read as 0, shall be written with 0.

Table 160 Access Mode Restrictions of OCS sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **SUS_P**	w	SUS	set SUS_P during write access
(default)	rX	SUS	read only

Access Enable Register 0

The Access Enable Register 0 controls write access for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B,,EN31 -> TAG ID 011111B.

All registers and memories of the GTM are protected, except for the following registers:

ACCEN0 and ACCEN1.

ACCEN0

Access Enable Register 0 **($09FD10_H$)** **Application Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID master peripheral mapping).

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B , EN31 -> TAG ID 111111B.

All registers and memories of the GTM are protected, except for the following registers:

ACCEN0 and ACCEN1.

ACCEN1

Access Enable Register 1 **($09FD14_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0, shall be written with 0.

Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. To reset a module kernel, it is necessary to set the RST bits by writing a ´1´ in both Kernel Reset Registers related to the module kernel that should be reset (kernel 0 or kernel 1). In order to support modules with two kernels, the BPI_FPI provides two set of kernel reset registers. The RST bit will be re-set (cleared) by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing to it with ´1´.

Note: This reset function has the effect as bit RST.RST.

RESET1

Kernel Reset Register 0 (09FD08$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RSTSTAT	RST
r														rh	rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates whether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related RESET_CLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0, shall be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the GTM kernel. GTM kernel registers related to the Debug Reset (Class 1) are not influenced. To reset the GTM kernel, it is necessary to set the RST bits by writing a ´1´ in both Kernel Reset registers. The RST bit will be cleared with the end of the BPI kernel reset sequence.

RESET2
Kernel Reset Register 1 **(09FD0C$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															RST
r															rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers are set. The RST bit will be cleared (re-set to ´0´) after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0, shall be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Register Clear register is used to clear the Kernel Reset Status bit (RESET1.RSTSTAT).

RESET_CLR
Kernel Reset Status Clear Register **(09FD04$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															CLR
r															w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status RESET1.RSTSTAT

Field	Bits	Type	Description
0	31:1	r	**Reserved** Read as 0, shall be written with 0.

28.25.6 Port Connections

This sections summarizes the port connections which define the boundary of the GTM for the TC3x.

Table 161 GTM Boundary Connections Overview

Description	Table
GTM to GPIO Connections	see Appendix
GPIO to GTM Connections	**Table 28.25.6.2**
DTM_AUX Input connections	See Appendix
GTM EXT CLOCK to GPIO Connections	**Table 162**
GPIO Availability by Package	See Data Sheet

28.25.6.1 GTM Outputs to Port Connections

All pinning information has been moved to the appendix of the corresponding device.

The GTM outputs cmu_eclk[2:0] are connected directly to dedicated ports

Table 162 GTM External Clock to Port Mapping

GTM External Clock	Output to PIN
cmu_eclk0	P23.1
cmu_eclk1	P32.4
cmu_eclk2	P11.12

28.25.6.1.1 GTM Outputs to Port Control Registers (TOUTSEL)

The outputs of the GTM TOM, ATOM and DTM modules are not connected directly to the port output path. Each TOM, ATOM and DTM is connectable to several port alternate inputs via an output multiplexer.

The output signals to the MUX are called *gtm_tout_x*, where x is the running number.

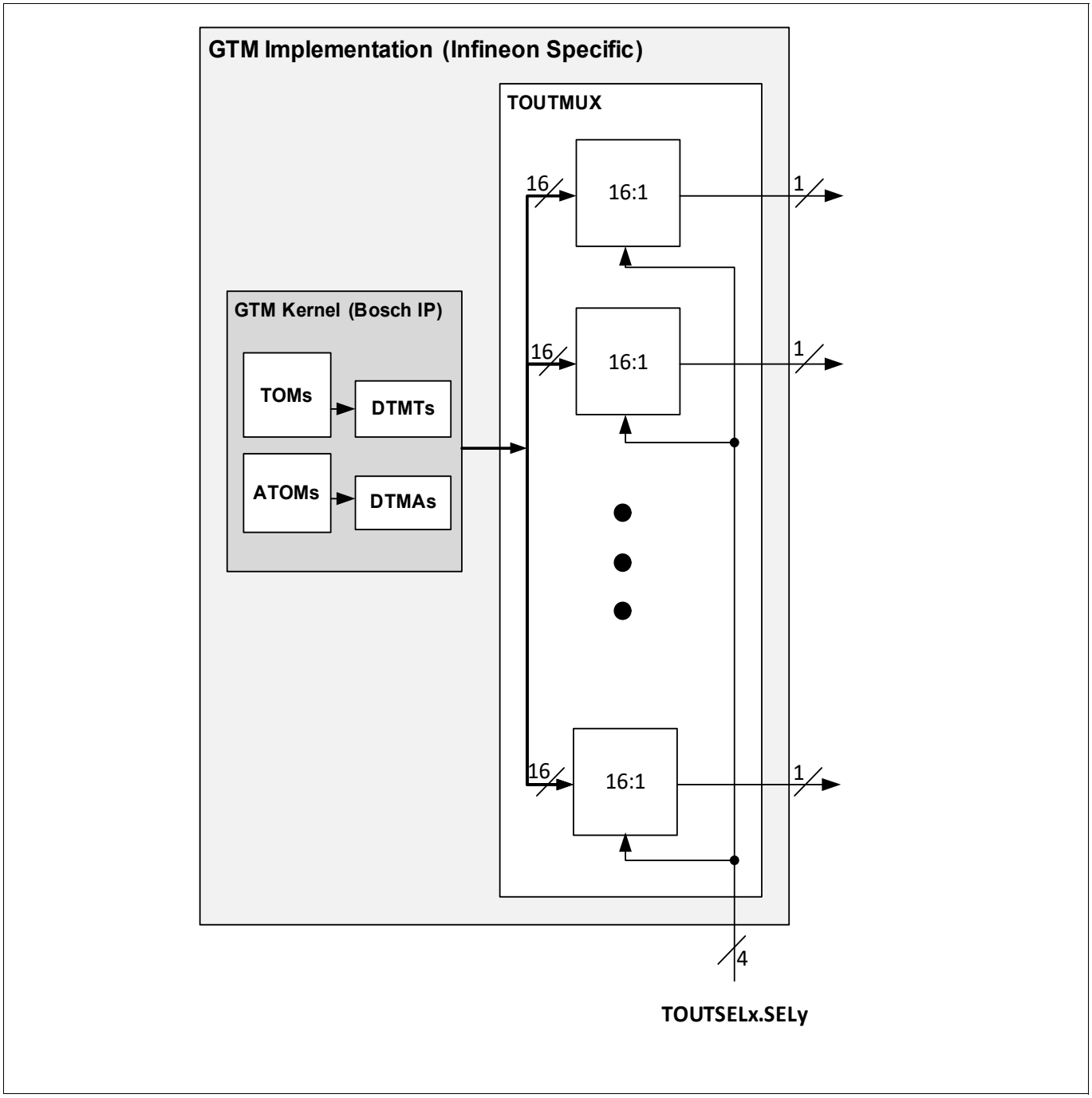
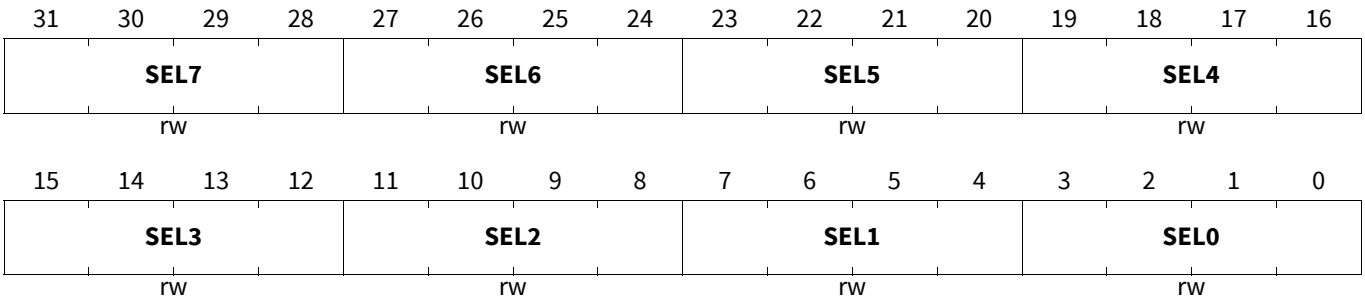


Figure 143 GTM Outputs to Port block diagram

Timer Output Select Register

TOUTSELn (n=0-33)
Timer Output Select Register **($09FD60_H$+n*4)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SELx (x=0-7)	4*x+3:4*x	rw	**TOUT(n*8 + x) Output Selection** This bit field defines which timer output is connected as TOUT(n*8+x). The mapping is defined in the specific appendix. *Note: If TOUT(n*8+x) is not defined in the specific appendix, this bit field has to be treated as reserved ('0', TOUT not routed to any ports).* *Note: SELx values not explicitly defined here are equivalent to the last defined SELx setting.*

28.25.6.2 GPIO to GTM Connections

The table below shows, for each available port, the GTM Input connections. The GTM Input modules (TIM) can be configured by setting the register.

All pinning information has been moved to the Appendix of the corresponding device.

28.25.6.2.1 Port to GTM Connections (TIMnINSEL)

The inputs to the GTM TIM modules (*gtm_timx_in[7:0]*) are not connected directly to the port input path. Each timed GPIO is connectable to two or more TIMs via an input multiplexer. In addition, the inputs from the on-chip peripherals are connected here, too.

TIMn Input Select Register

TIMnINSEL (n=0-7)
TIMn Input Select Register **($09FD40_H$+n*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CH7SEL				CH6SEL				CH5SEL				CH4SEL			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CH3SEL				CH2SEL				CH1SEL				CH0SEL			
rw				rw				rw				rw			

Field	Bits	Type	Description
CHxSEL (x=0-7)	4*x+3:4*x	rw	**TIM Channel x Input Selection** This bit defines which input is connected for TIMn channel x of the GTM. The input is either derived from a port pad or from an on-chip module.

Fast ADC Channel to GTM

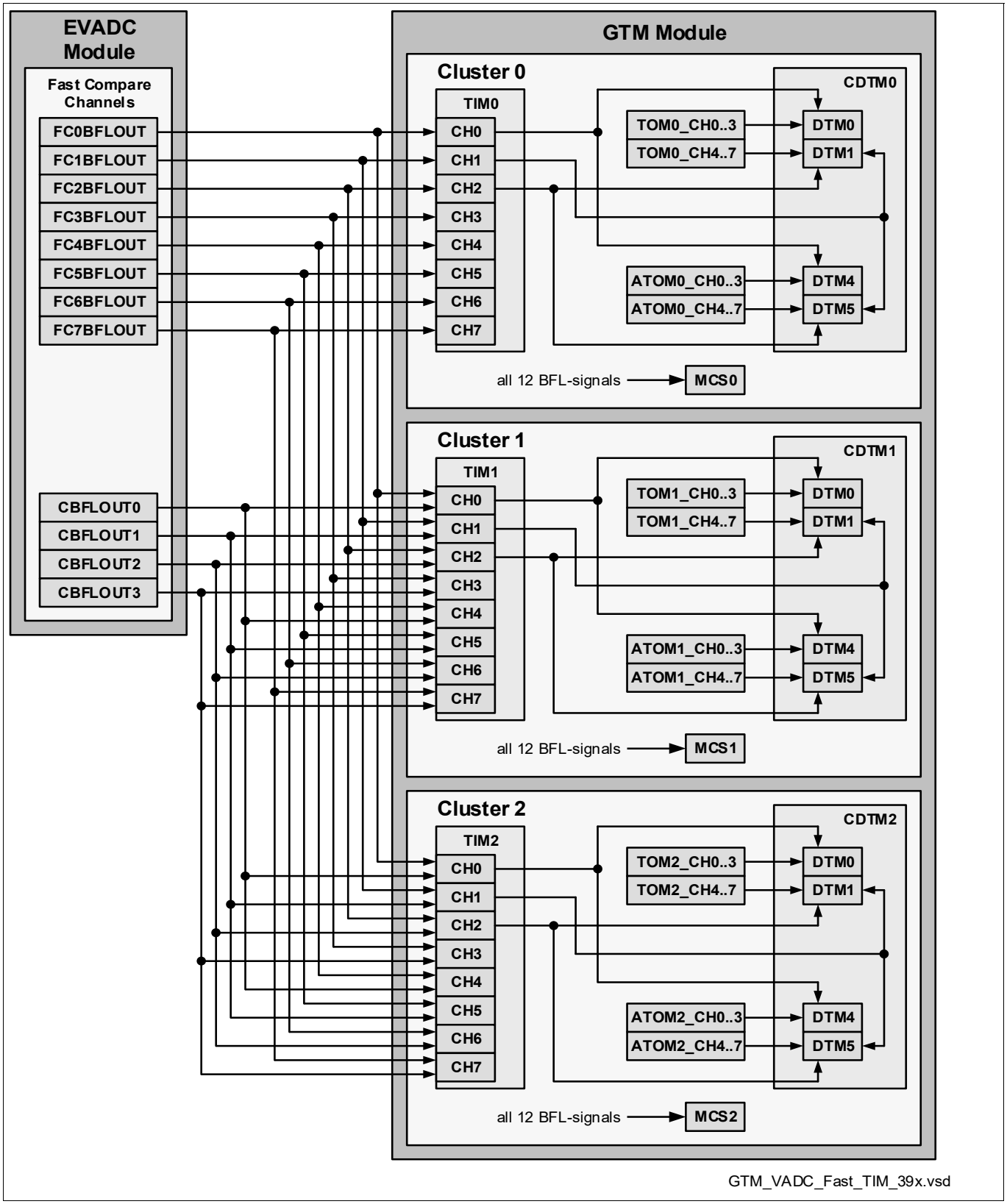


Figure 144 Fast ADC to GTM TIMs connections

28.25.6.2.2 GPIO/EDSADC to DTM_AUX Connections

DTM_AUX Input Selection Register

DTMAUXINSEL
DTM_AUX Input Selection Register (09FFD8$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						TSEL4		TSEL3		TSEL2		TSEL1		TSEL0	
r						rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		ASEL6		ASEL5		ASEL4		ASEL3		ASEL2		ASEL1		ASEL0	
r		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
ASELx (x=0-6)	2*x+1:2*x	rw	**CDTMx_DTM4_AUX Input Selection (ATOMx_CH0...3)** This bit field defines which GPIO/DSADC/EVADC signal is connected to the CDTMx_DTM4_AUX input.
TSELx (x=0-4)	2*x+17:2*x+16	rw	**CDTMx_DTM0_AUX Input Selection (TOMx_CH0...3)** This bit field defines which GPIO/DSADC/EVADC signal is connected to the CDTMx_DTM0_AUX input.
0	15:14, 31:26	r	**Reserved** Read as 0, shall be written with 0.

28.25.7 MSC Connections

This sections summarizes the connections to the MSC modules for the TC3x. The MSC interfaces (MSC0 and MSC1 and MSC2 and MSC3) provide a serial communication link typically used to connect power switches or other peripheral devices.

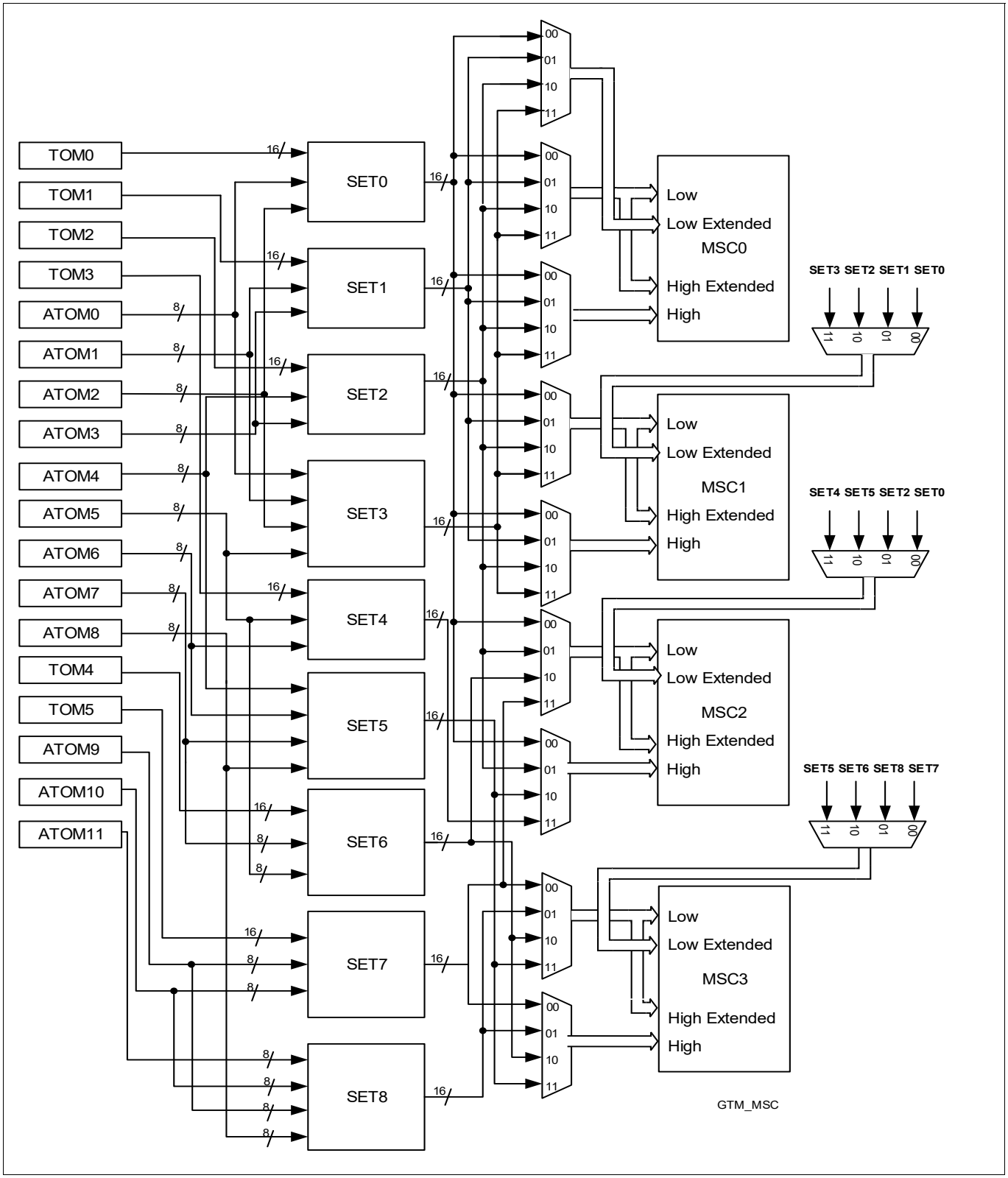


Figure 145 GTM to MSC connections

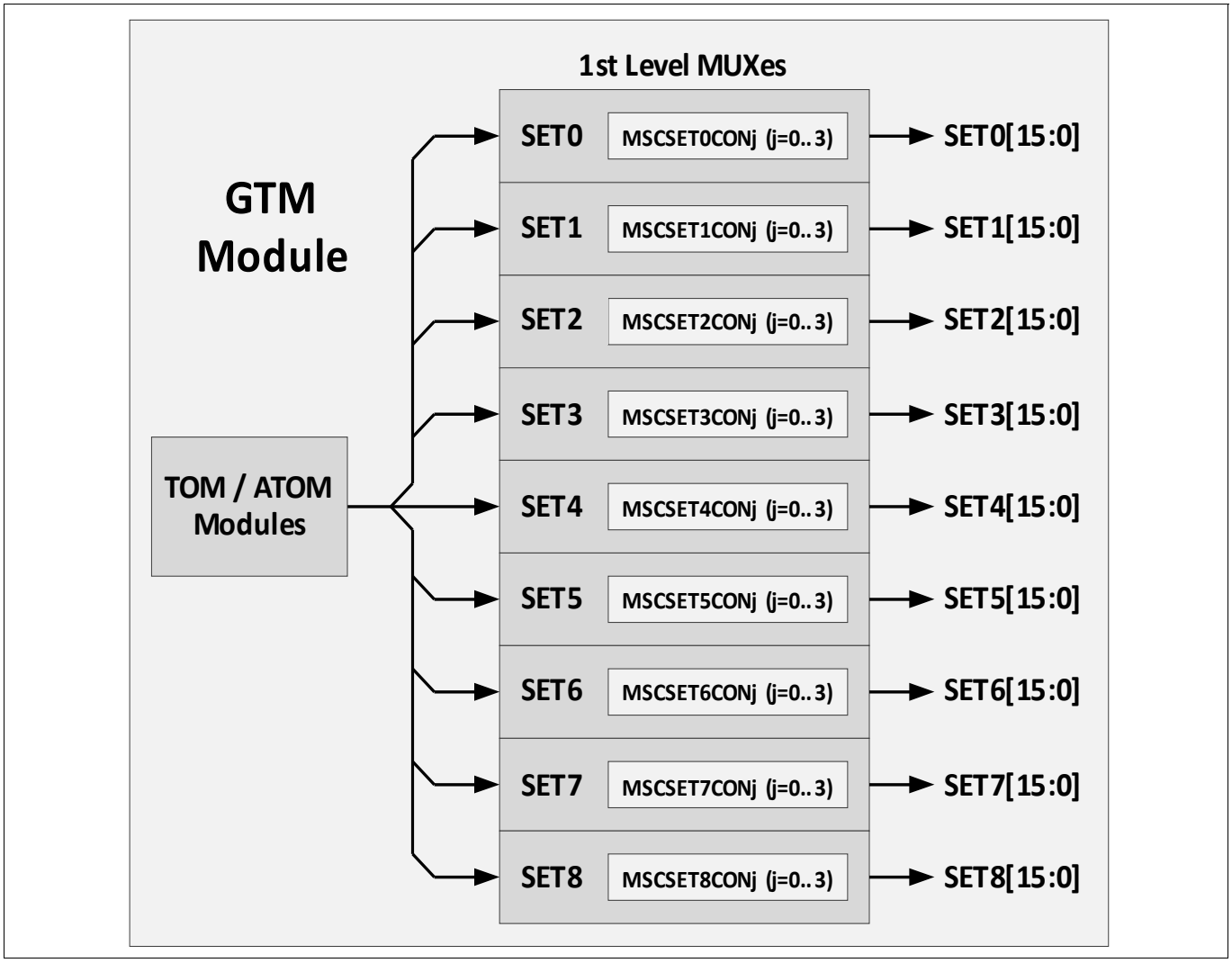


Figure 146 GTM to MSC, SETx Multiplexers

28.25.7.1 GTM to MSC Control Registers

MSC Set i Control j Register

MSCSETiCONj (i=0-8;j=0)
MSC Set i Control j Register **($09FF00_H$+i*10_H+j*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**SEL3**					**0**			**SEL2**				
r			rw					r			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**SEL1**					**0**			**SEL0**				
r			rw					r			rw				

Field	Bits	Type	Description
SELk (k=0-3)	8*k+4:8*k	rw	**Set i[k] Input Selection** This bit field defines the GTM timer source configured as Set i signal k out. Decoding is defined in the specific appendix.
0	31:29, 23:21, 15:13, 7:5	r	**Reserved** Read as 0, shall be written with 0.

MSCSETiCONj (i=0-8;j=1)
MSC Set i Control j Register **($09FF00_H$+i*10_H+j*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**SEL7**					**0**			**SEL6**				
r			rw					r			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**SEL5**					**0**			**SEL4**				
r			rw					r			rw				

Field	Bits	Type	Description
SELk (k=4-7)	8*k-28:8*k-32	rw	**Set i[k] Input Selection** This bit field defines the GTM timer source configured as Set i signal k out. Decoding is defined in the specific appendix.
0	31:29, 23:21, 15:13, 7:5	r	**Reserved** Read as 0, shall be written with 0.

MSCSETiCONj (i=0-8;j=2)
MSC Set i Control j Register **($09FF00_H$+i*10_H+j*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**SEL11**					**0**			**SEL10**				
r			rw					r			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**SEL9**					**0**			**SEL8**				
r			rw					r			rw				

Field	Bits	Type	Description
SELk (k=8-11)	8*k-60:8*k-64	rw	**Set i[k] Input Selection** This bit field defines the GTM timer source configured as Set i signal k out. Decoding is defined in the specific appendix.
0	31:29, 23:21, 15:13, 7:5	r	**Reserved** Read as 0, shall be written with 0.

MSCSETiCONj (i=0-8;j=3)
MSC Set i Control j Register **($09FF00_H$+i*10_H+j*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**SEL15**					**0**			**SEL14**				
r			rw					r			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**SEL13**					**0**			**SEL12**				
r			rw					r			rw				

Field	Bits	Type	Description
SELk (k=12-15)	8*k-92:8*k-96	rw	**Set i[k] Input Selection** This bit field defines the GTM timer source configured as Set i signal k out. Decoding is defined in the specific appendix.
0	31:29, 23:21, 15:13, 7:5	r	**Reserved** Read as 0, shall be written with 0.

MSCi Input Low Control Register

MSCiINLCON (i=0-3)
MSCi Input Low Control Register (09FF90$_H$+i*12) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEL15		SEL14		SEL13		SEL12		SEL11		SEL10		SEL9		SEL8	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL7		SEL6		SEL5		SEL4		SEL3		SEL2		SEL1		SEL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
SELx (x=0-15)	2*x+1:2*x	rw	**GTM MSCq Low x Output Selection** GTM output gtm_mscqltinl[x] is controlled by the timer output.

MSCi Input High Control Register

MSCiINHCON (i=0-3)
MSCi Input High Control Register (09FF94$_H$+i*12) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEL15		SEL14		SEL13		SEL12		SEL11		SEL10		SEL9		SEL8	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL7		SEL6		SEL5		SEL4		SEL3		SEL2		SEL1		SEL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
SELx (x=0-15)	2*x+1:2*x	rw	**GTM MSCq High x Output Selection** GTM output gtm_mscqaltinh[x] is controlled by the timer output.

MSCi Input Low Extended Control Register

MSCiINLEXTCON (i=0-3)
MSCi Input Low Extended Control Register (09FF98$_H$+i*12) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEL15		SEL14		SEL13		SEL12		SEL11		SEL10		SEL9		SEL8	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL7		SEL6		SEL5		SEL4		SEL3		SEL2		SEL1		SEL0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
SELx (x=0-15)	2*x+1:2*x	rw	**GTM MSCq LowExtended x Output Selection** GTM output gtm_mscqaltinext[x] is controlled by the timer output.

28.25.8 EDSADC Connections

28.25.8.1 EDSADC to GTM Connections

These registers define the EDSADC to GTM connections of TC3x.
The 16-to-1 multiplexer is implemented once per TIM0/1/2/3/4/5 module/channel. The input signal from the EDSADC is called *dsadcx_trig_in_y, where* x = number of EDSADC channel, y = SRM / SAUL / SBLL. The output of the multiplexer is called *timy_muxout_x*, (y = 0,1)(x = 0…7).

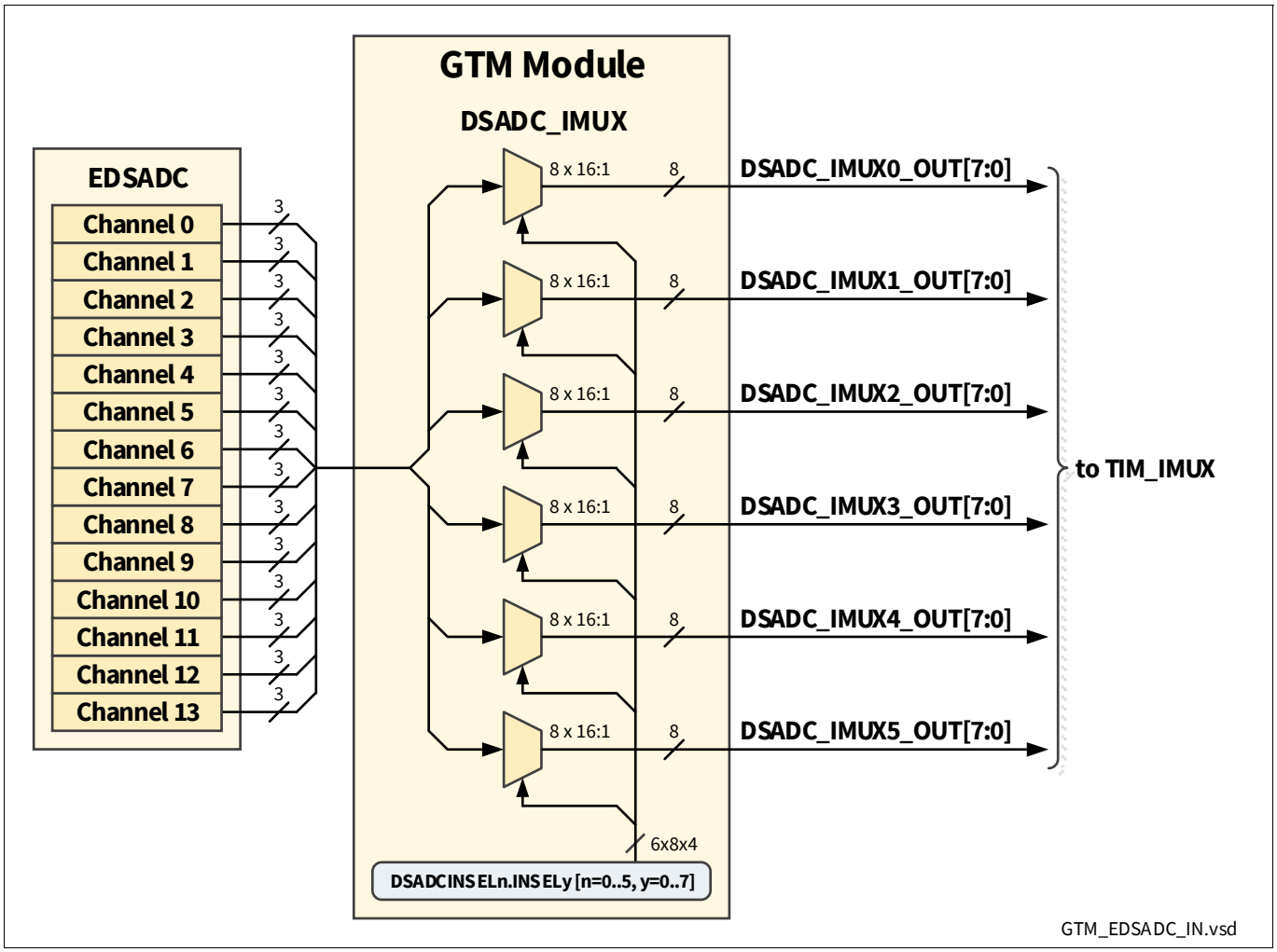


Figure 147 EDSADC to TIM connections

DSADC Input Select i Register

DSADCINSELi (i=0-5)

DSADC Input Select i Register **($09FE00_H$+i*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INSEL7				INSEL6				INSEL5				INSEL4			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INSEL3				INSEL2				INSEL1				INSEL0			
rw				rw				rw				rw			

Field	Bits	Type	Description
INSELj (j=0-7)	4*j+3:4*j	rw	**In Selection for DSADCn GTM connection** This bit field defines which DSADCn output is connected if the channel input mux is configured as DSADC input for TIMi channel j.

28.25.8.2 GTM to EDSADC Connections

Registers DSADCOUTSELxy define the GTM to EDSADC connections of TC3x.

The EDSADC trigger inputs (one per EDSADC) 16-to-1 multiplexer is implemented four times per EDSADC module/channel. The output signal to the EDSADC are called *dsadcx_trig0,dsadcx_trig1*, *dsadcx_trig2, and dsadcx_trig3*. The inputs of the multiplexer are called *dsadc_x_muxin[7:0]*, x = number of EDSADC channel.

Which of the possible trigger inputs is used by the EDSADC can be selected in the EDSADC related chapter.

DSADC Output Select i0 Register

DSADCOUTSELi0 (i=0-3)

DSADC Output Select i0 Register **($09FE20_H$+i*8)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEL7				SEL6				SEL5				SEL4			
rw				rw				rw				rw			

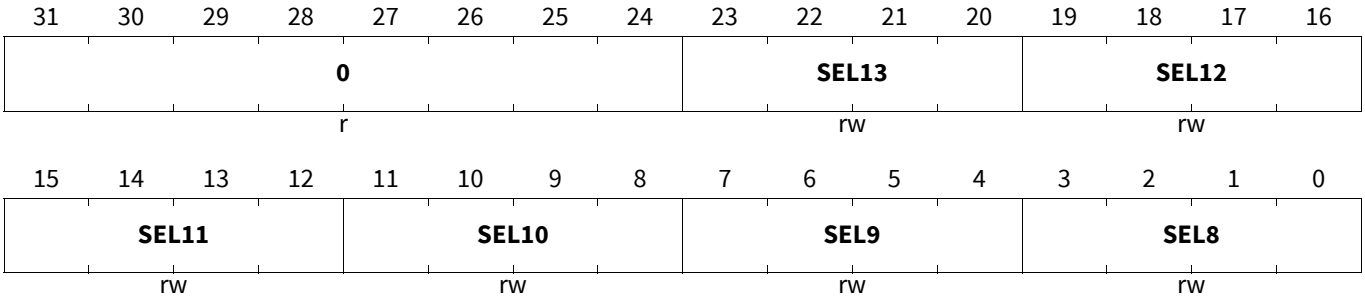
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL3				SEL2				SEL1				SEL0			
rw				rw				rw				rw			

Field	Bits	Type	Description
SELx (x=0-7)	4*x+3:4*x	rw	**Output Selection for DSADCx GTM connection** This bit field defines which TOM/ATOM channel output is used as DSADCx trigger i.

DSADC Output Select i1 Register

DSADCOUTSELi1 (i=0-3)

DSADC Output Select i1 Register **($09FE24_H$+i*8)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SELx (x=8-13)	4*x-29:4*x-32	rw	**Output Selection for GTM to DSADCx connection** This bit field defines which TOM/ATOM channel output is used as DSADCx trigger i.
0	31:24	r	**Reserved** Read as 0, shall be written with 0.

28.25.9 EVADC Connections

These registers define the GTM to EVADC connections of TC3x.
The ADC trigger inputs 16-to-1 multiplexer is implemented four times per ADC module / channel. The output signals to the ADC are called *adcx_trig0, adcx_trig1, adcx_trig2, adcx_trig3,* and *adcx_trig4*.

The inputs of the multiplexer are called *adc_x_muxin[7:0]*, x = number of ADCs.

ADC Trigger i Output Select 0 Register

ADCTRIGiOUT0 (i=0-4)
ADC Trigger i Output Select 0 Register **($09FE40_H$+i*8)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEL7				SEL6				SEL5				SEL4			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL3				SEL2				SEL1				SEL0			
rw				rw				rw				rw			

Field	Bits	Type	Description
SELx (x=0-7)	4*x+3:4*x	rw	**Output Selection for GTM to ADCx connection** This bit field defines which TOM/ATOM channel output is used as ADCx trigger i. The decoding is defined in the specific appendix depending on the ADC.

ADC Trigger i Output Select 1 Register

ADCTRIGiOUT1 (i=0-4)
ADC Trigger i Output Select 1 Register **($09FE44_H$+i*8)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL3				SEL2				SEL1				SEL0			
rw				rw				rw				rw			

Field	Bits	Type	Description
SELx (x=0-3)	4*x+3:4*x	rw	**Output Selection for GTM to ADCx connection** This bit field defines which TOM/ATOM channel output is used as ADCx+8 trigger i. Decoding is defined in the specific appendix.
0	31:16	r	**Reserved** Read as 0, shall be written with 0.

28.25.10 GTM ADC Interface

The GTM ADC Interface connects the MCSi (i=0..4) to all the available EVADC/EDSADC modules.

Each MCSi module (i=0..4) is able to read, in a flexible manner, the GxRES15 result of any EVADC channel, plus the result of any EDSADC channel.

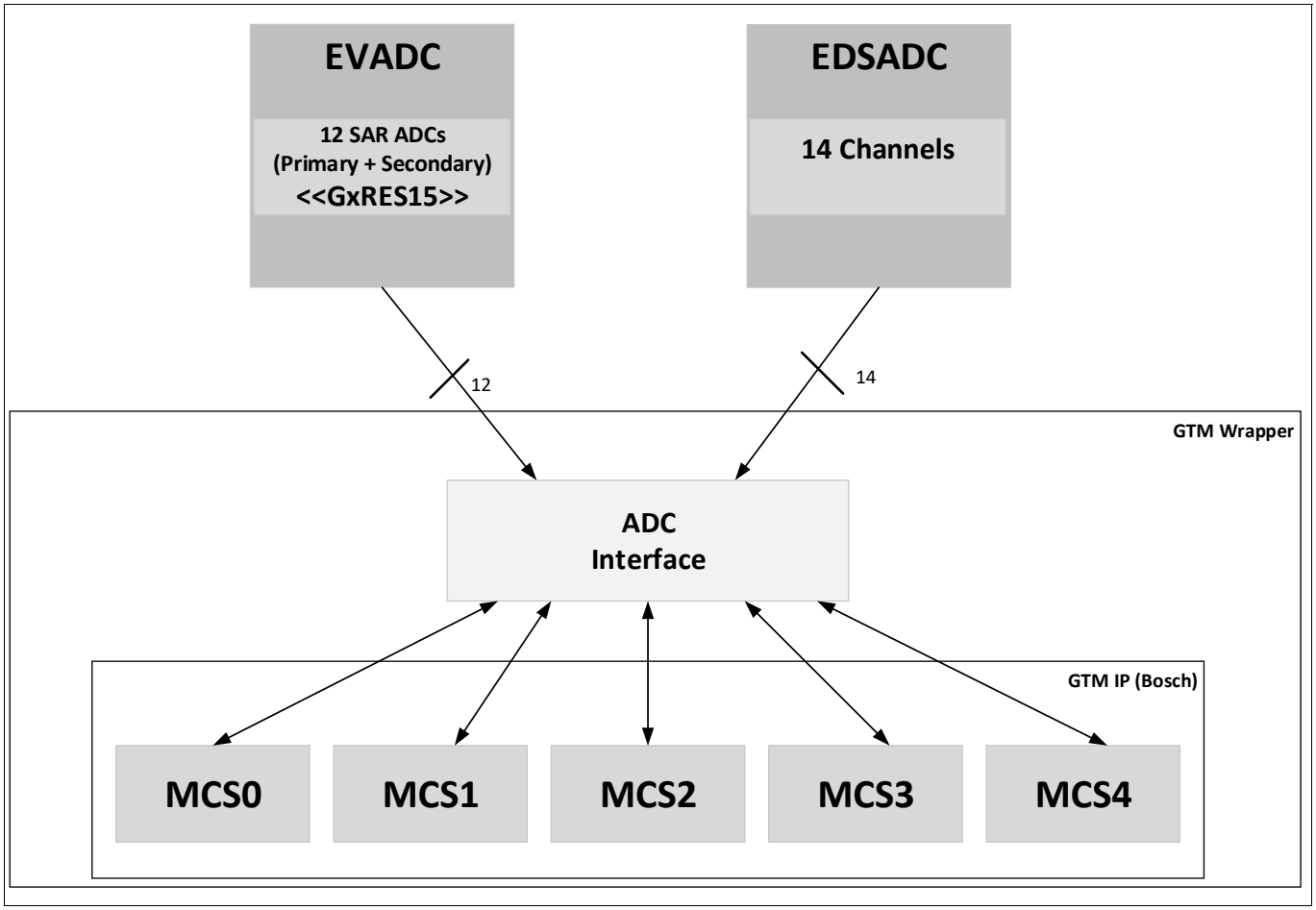


Figure 148 GTM ADC Interface

In order to address the desired EVADC/EDSADC channel, each MCS task should use the BRD (Bus Read) instructions defined in the MCS Chapter.

Inside the GTM Wrapper, the ADC interface is implemented in a way that each EVADC/EDSADC result can be provided to each MCSi (i=0..4) module, without any arbitration.

The ADC interface takes care locally of the valid bit handling.

Valid bit handling

As soon as a new ADC result is available, a local valid bit is set within the ADC interface logic. This information is provided towards the MCS via the ‚valid' signal, which is available along with the upper part of the actual data within the MHB registers (MHB = ack & valid & data[29:24]).

Each MCS read access, to an ADC result, resets the valid bit (i.e. BRD Rx, <ADC_ID>), No data (before the first ADC data becomes available) will be flagged by a „10" ‚req_status' and no ‚ack' pulse (causing an internal MCS error). In this way, the MCS can distinguish between non-available data and invalid data (data provided by the ADC but already read by another MCS or MCS channel).

MCS read access to a not supported address will cause an internal MCS error flagged by ‘req_status = 11’ and no “ack” pulse.

It is possible that two different MCS modules read the same ADC result at the same time, or within two consecutive clock cycles. In this corner case, for both MCS the valid bit will be set. At application level, countermeasures need to be taken in order to avoid such a behavior.

Table 163 ADC Interface Return Codes

CH_INFO	REQ	ACK	VALID	REQ_STATUS
0|1	1	0	1	11: unsupported address, <ADCi_CH=n> not available
0|1	1	1	1	00: new data available in <ADCi_DATA[23:0]> register. (Signed format for EDSADC)
0|1	1	1	0	00: no new data available
0|1	1	0	0	00: conversion in progress or ADC not enabled; No data available yet

MCS Master Interface offsets vs. EVADC/EDSADC

Inside the GTM IP, the connection between the ADI and the MCS is implemented with the MCS master interface (see MCS master interface address map in the product specific appendix), and the MCS should access the wanted ADC by using the defined ADC_CHx_DATA and/or ADC_CHx_STA (x=0..63) offsets.

Note: Currently only x=0..31 is available. The other addresses x=32..63 are reserved for future extensions of the GTM3.x IP. In GTM4 the address map has changed completely, therefore the addresses stay reserved.

Table 164 MCS Master Interface Offsets vs. EVADC/EDSADC

MCS Master Interface offset	EVADC, EDSADC Module
ADC_CH0_DATA	ADC channel 0
ADC_CH0_STA	ADC channel 0
ADC_CH1_DATA	ADC channel 1
ADC_CH1_STA	ADC channel 1
ADC_CH2_DATA	ADC channel 2
ADC_CH2_STA	ADC channel 2
ADC_CH3_DATA	ADC channel 3
ADC_CH3_STA	ADC channel 3
ADC_CH4_DATA	ADC channel 4
ADC_CH4_STA	ADC channel 4
ADC_CH5_DATA	ADC channel 5
ADC_CH5_STA	ADC channel 5
ADC_CH6_DATA	ADC channel 6
ADC_CH6_STA	ADC channel 6

Table 164 MCS Master Interface Offsets vs. EVADC/EDSADC (cont'd)

MCS Master Interface offset	EVADC, EDSADC Module
ADC_CH7_DATA	ADC channel 7
ADC_CH7_STA	ADC channel 7
ADC_CH8_DATA	ADC channel 8
ADC_CH8_STA	ADC channel 8
ADC_CH9_DATA	ADC channel 9
ADC_CH9_STA	ADC channel 9
ADC_CH10_DATA	ADC channel 10
ADC_CH10_STA	ADC channel 10
ADC_CH11_DATA	ADC channel 11
ADC_CH11_STA	ADC channel 11
ADC_CH12_DATA	Not Used
ADC_CH12_STA	Not Used
ADC_CH13_DATA	Not Used
ADC_CH13_STA	Not Used
ADC_CH14_DATA	Not Used
ADC_CH14_STA	Not Used
ADC_CH15_DATA	Not Used
ADC_CH15_STA	Not Used
ADC_CH16_DATA	EDSADC channel 0
ADC_CH16_STA	EDSADC channel 0
ADC_CH17_DATA	EDSADC channel 1
ADC_CH17_STA	EDSADC channel 1
ADC_CH18_DATA	EDSADC channel 2
ADC_CH18_STA	EDSADC channel 2
ADC_CH19_DATA	EDSADC channel 3
ADC_CH19_STA	EDSADC channel 3
ADC_CH20_DATA	EDSADC channel 4
ADC_CH20_STA	EDSADC channel 4
ADC_CH21_DATA	EDSADC channel 5
ADC_CH21_STA	EDSADC channel 5
ADC_CH22_DATA	EDSADC channel 6
ADC_CH22_STA	EDSADC channel 6
ADC_CH23_DATA	EDSADC channel 7
ADC_CH23_STA	EDSADC channel 7
ADC_CH24_DATA	EDSADC channel 8
ADC_CH24_STA	EDSADC channel 8
ADC_CH25_DATA	EDSADC channel 9

Table 164 MCS Master Interface Offsets vs. EVADC/EDSADC (cont'd)

MCS Master Interface offset	EVADC, EDSADC Module
ADC_CH25_STA	EDSADC channel 9
ADC_CH26_DATA	EDSADC channel 10
ADC_CH26_STA	EDSADC channel 10
ADC_CH27_DATA	EDSADC channel 11
ADC_CH27_STA	EDSADC channel 11
ADC_CH28_DATA	EDSADC channel 12
ADC_CH28_STA	EDSADC channel 12
ADC_CH29_DATA	EDSADC channel 13
ADC_CH29_STA	EDSADC channel 13

MCS Interface Usage

Each MCSi (i=0..4) task can read from an EVADC or EDSADC using the MCS bus read command BRD or BRDI:

Example:

- BRD Rx ADC_CHy_DATA or BRD Rx ADC_CHy_STA[1)]

At the end of the read command, the content of the registers will be the following:

- MHB = ack & valid & MHB[5:0]
- Rx[15:0] = ADC_CHy Result

Reading from EVADC

- MHB[5:0] = MHB[5:4]= 00_B, MHB[3:0]= EVADC channel number
- Rx[23:16] = 0x00000000

Reading from EDSADC

- MHB[5:0] = MHB[5:2]= 0000_B, MHB[1:0]= EDSADC input multiplexer
- Rx[23:16] = each bit is equal to EDSADC[15] (the sign is extended to all leading bits)

28.25.11 SENT Connections

This register defines the GTM to SENT connections of TC3x.
The trigger generation 16-to-1 multiplexer is implemented four times per EVADC module / channel. The output signal to the EVADC is called *adcx_trig0, adcx_trig1,* adcx_trig2, and adcx_trig3.

As the SENT channels overlay and replace ADC channels, the ADC triggers will be also reused for the SENT channels. Therefore, no additional outputs for separate operation are defined here. The *adcx_trig0[x=0-11] are* connected to sent_trig_i [i=0-11].

In addition, also the dsadc_x_trig0 [x=0…3] are connected to the sent_trig_i [i=12-15]

1) ADC_CHx_DATA and ADC_CHx_STA are equivalent, will return the same data

28.25.12 CAN / TTCAN Connection

CAN0/CAN1 Output Select Register

This register holds the selection for the trigger outputs to the CAN0/CAN1 modules. Bit fields SEL0..3 define the selection for triggers 0..3 for CAN0, while bit fields SEL4..7 define the selection for triggers 0..3 for CAN1.

CANOUTSEL0
CAN0/CAN1 Output Select Register **($09FFDC_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEL7				SEL6				SEL5				SEL4			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL3				SEL2				SEL1				SEL0			
rw				rw				rw				rw			

Field	Bits	Type	Description
SELx (x=0-7)	4*x+3:4*x	rw	**Output Selection for GTM to CAN connection x** This bit field defines which TOM/ATOM channel output is used as CAN0/CAN1 node trigger x. The decoding is defined in the specific appendix.

CAN2 Output Select Register

This register holds the selection for the trigger outputs 0..3 to the CAN2 module.

CANOUTSEL1
CAN2 Output Select Register **($09FFE0_H$)** **Application Reset Value: $XXXX\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL3				SEL2				SEL1				SEL0			
rw				rw				rw				rw			

Field	Bits	Type	Description
SELx (x=0-3)	4*x+3:4*x	rw	**Output Selection for GTM to CAN connection x** This bit field defines which TOM/ATOM channel output is used as CAN2 node trigger x. The decoding is defined in the specific appendix.
0	31:16	r	**Reserved** Read as 0, shall be written with 0.

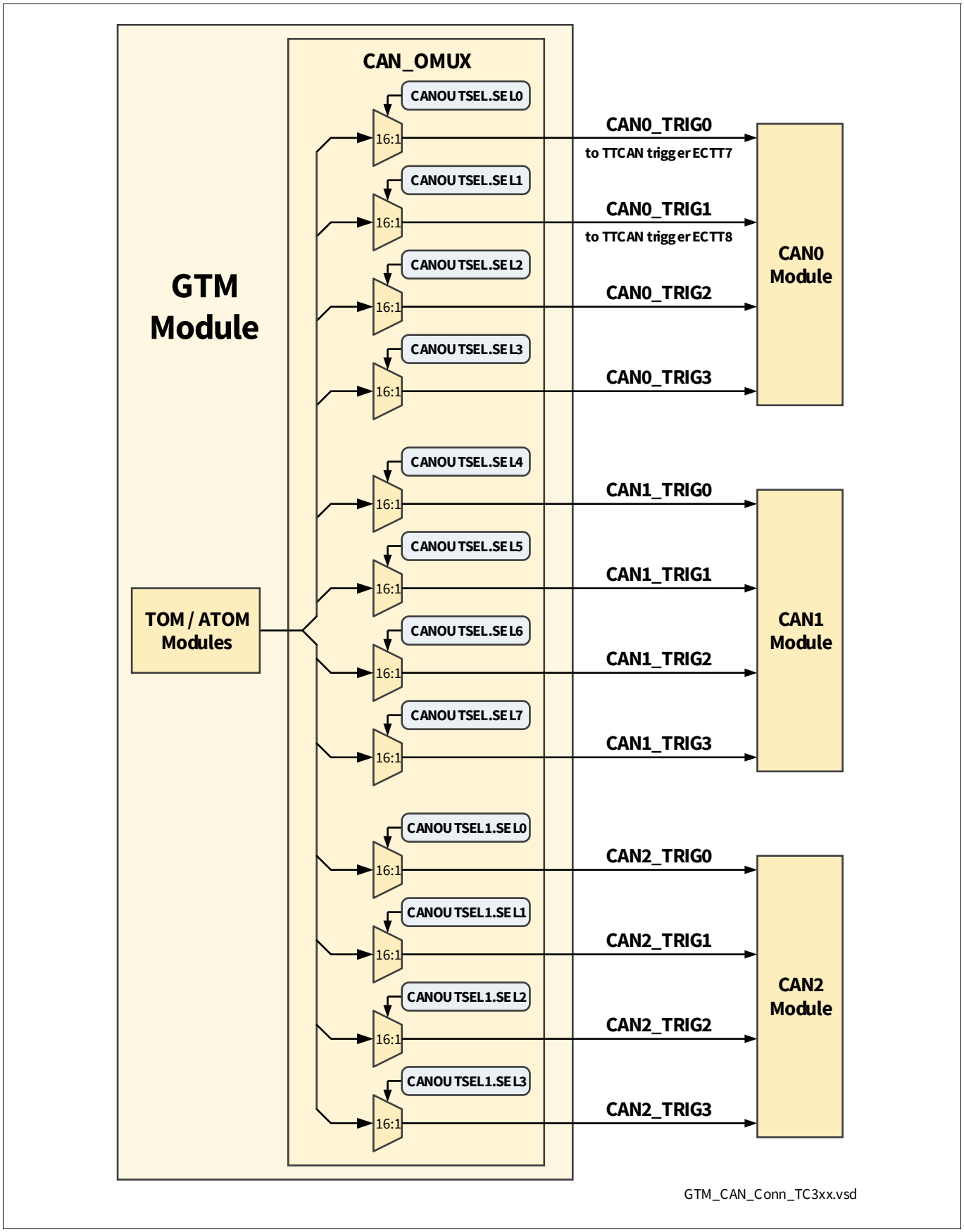


Figure 149 GTM to CAN/TTCAN Connections

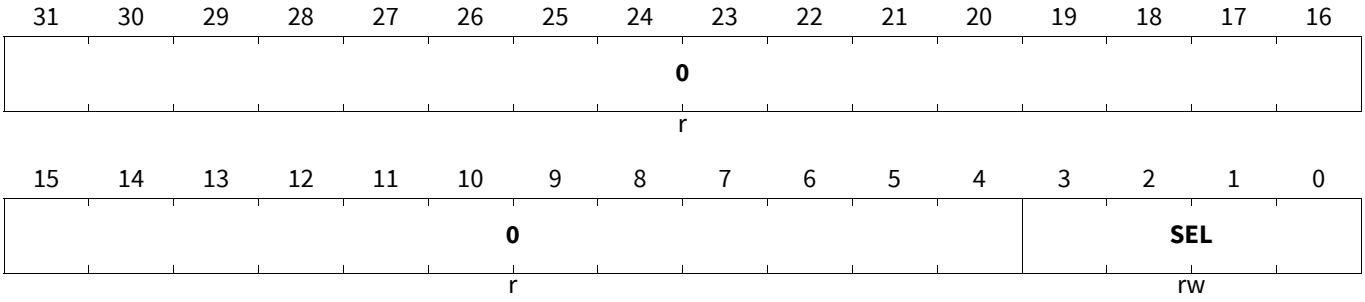
28.25.13 LC DC/DC Connection

This register defines the GTM to LC DC/DC connections of TC3x.

LCDCDC Output Select Register

LCDCDCOUTSEL

LCDCDC Output Select Register (09FFD4$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SEL	3:0	rw	**Output Selection for GTM to LCDCDC connection** This bit field defines which TOM/ATOM channel output is used as LCDCDC signal. Decoding is defined in the specific appendix.
0	31:4	r	**Reserved** Read as 0, shall be written with 0.

28.25.14 CCU6x Connections

This section summarizes the connections to the CCU6x.

Table 165 GTM to CCU60 Connections

GTM Output	CCU60 Input
TOM0_8	T12HRD
TOM1_8	T13HRD

Table 166 GTM to CCU61 Connections

GTM Output	CCU61 Input
ATOM0_1 (CDTM0_DTM4_1)	T12HRD
TOM0_9	T13HRD

28.25.15 PSI5 Connections

This register defines the GTM to PSI5 connections of TC3x.

The PSI5 trigger inputs are generated by 16-to-1 multiplexer that are implemented twice per PSI5 module / channel plus 3 combined for all PSI5 channels together. The output signal to the PSI5 is called *psi5x_trig0*. The inputs of the multiplexer are called *psi5_x_muxin[7:0]*, x = number of PSI5 triggers, TC3x has 6 triggers.

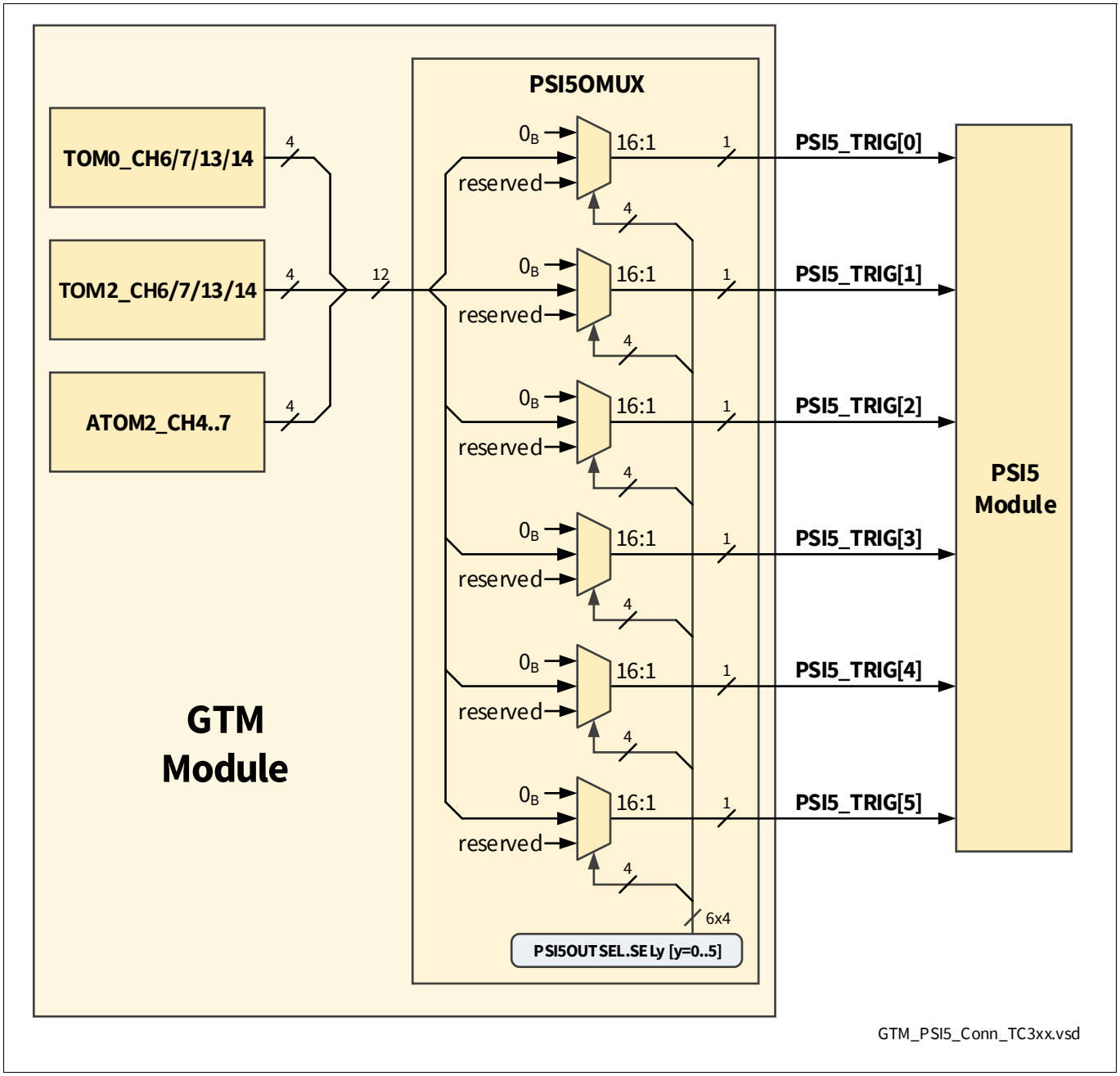


Figure 150 GTM to PSI5 Connections

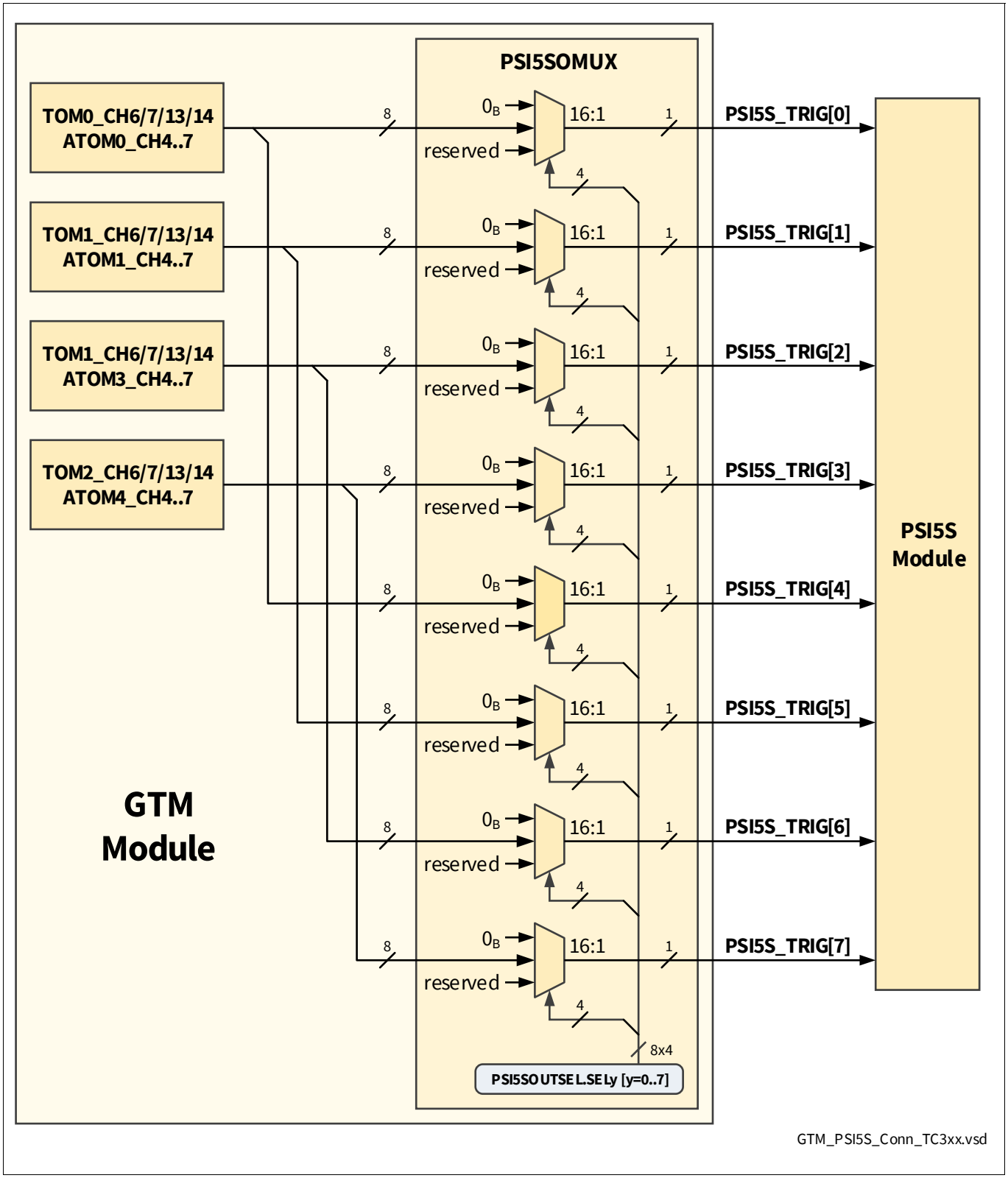


Figure 151 GTM to PSI5S Connections

PSI5 Output Select Register

PSI5OUTSEL

PSI5 Output Select Register **(09FFCC$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SEL5				SEL4			
r								rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL3				SEL2				SEL1				SEL0			
rw				rw				rw				rw			

Field	Bits	Type	Description
SELx (x=0-5)	4*x+3:4*x	rw	**Output Selection for GTM to PSI5x connection** This bit field defines which TOM/ATOM channel output is used as PSI5 trigger x. Decoding is defined in the specific appendix.
0	31:24	r	**Reserved** Read as 0, shall be written with 0.

PSI5-S Output Select Register

PSI5SOUTSEL

PSI5-S Output Select Register **(09FFD0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEL7				SEL6				SEL5				SEL4			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL3				SEL2				SEL1				SEL0			
rw				rw				rw				rw			

Field	Bits	Type	Description
SELx (x=0-7)	4*x+3:4*x	rw	**Output Selection for GTM to PSI5-S connection** This bit field defines which TOM/ATOM channel output is used as PSI5-S trigger x. Decoding is defined in the specific appendix.

28.25.16 GTM Data Exchange Registers

The registers define additional data exchange and control options for GTM towards the on-chip system of TC3x.

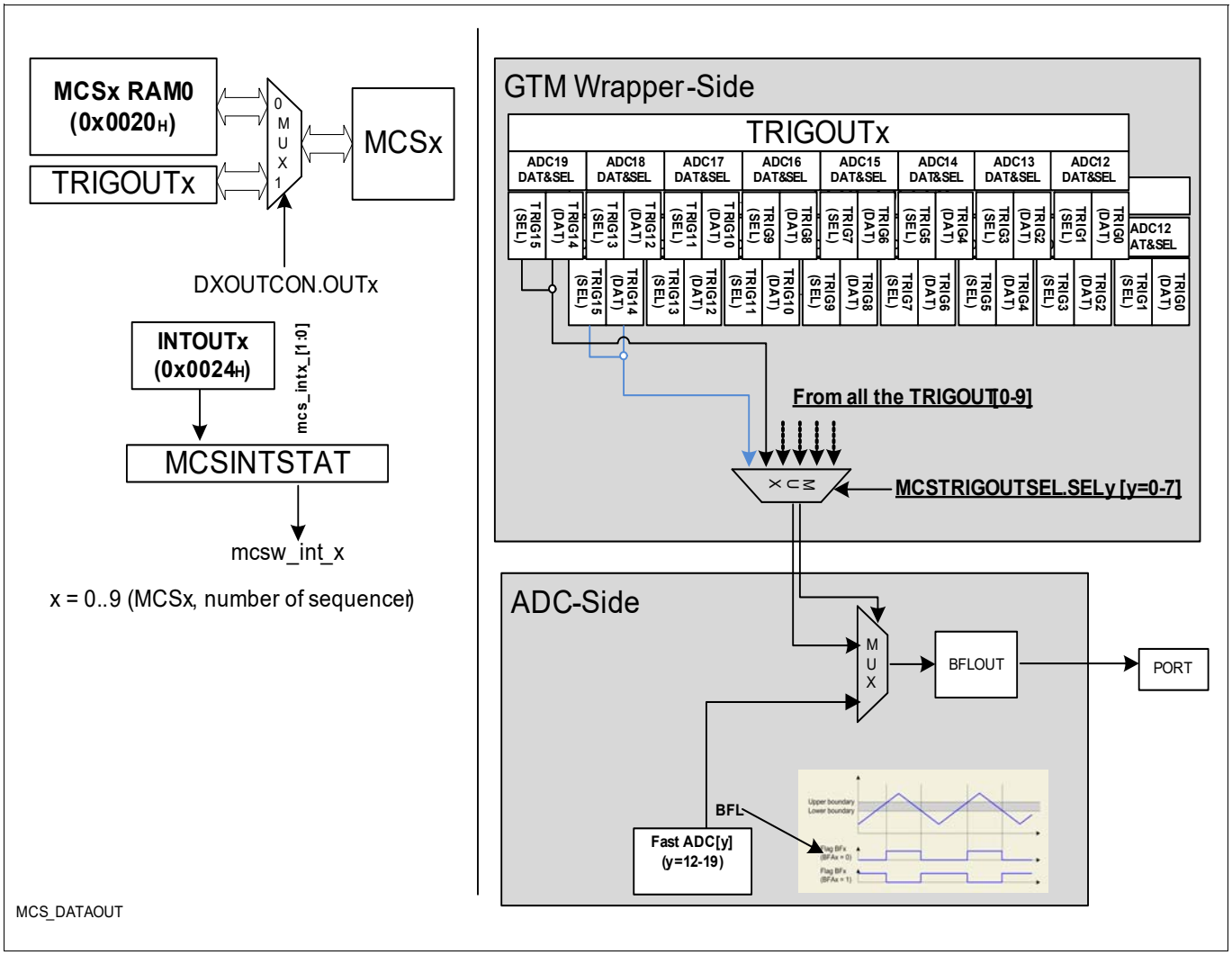


Figure 152 MCS Trigger Output Path to BFLOUT

The TRIGOUTx and INTOUTn registers enable the GTM multichannel sequencers to generate dedicated triggers and interrupts by writing specific values to those registers. The TRIGOUT registers are mirrored to address 0x0020 of the sequencer RAM0 and the INTOUTx to the address 0x0024.

If a write for address $0x0020_H$/$0x0024_H$ updates the RAM or the TRIGOUT/INTOUT registers, is controlled by the bus accessible register DXOUTCON.

This need to be configured only once when setting up the GTM with the reminder of the system.

Register MCSINTSTAT contains interrupt status flags and register MCSINTCLR provides the option to clear these status flags from the system bus. Both registers are optional register.

The data exchange register are also accessible during suspend mode.

Data Exchange Output Control Register

DXOUTCON
Data Exchange Output Control Register (09FE70$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						OUT9	OUT8	OUT7	OUT6	OUT5	OUT4	OUT3	OUT2	OUT1	OUT0
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
OUTx (x=0-9)	x	rw	**Output 0x Control** This bit defines whether register TRIGOUTx/INTOUTx is accessible from the MCS instead of RAM0 or not. 0_B RAM0 memory is accessed 1_B Register TRIGOUTx/INTOUTx is accessed
0	31:10	r	**Reserved** Read as 0, shall be written with 0.

Trigger Output Register n

TRIGOUTn (n=0-9)
Trigger Output Register n (09FE74$_H$+n*4) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRIG15		TRIG14		TRIG13		TRIG12		TRIG11		TRIG10		TRIG9		TRIG8	
w		w		w		w		w		w		w		w	

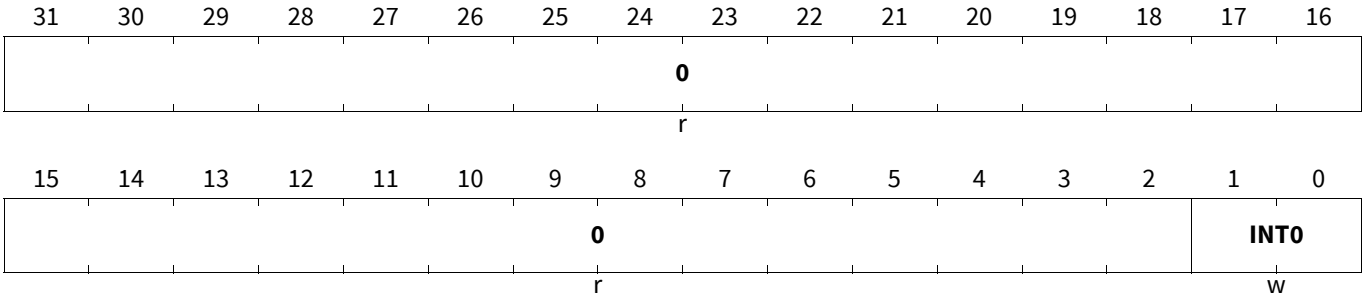
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIG7		TRIG6		TRIG5		TRIG4		TRIG3		TRIG2		TRIG1		TRIG0	
w		w		w		w		w		w		w		w	

Field	Bits	Type	Description
TRIGx (x=0-15)	2*x+1:2*x	w	**Trigger x** This bit field defines whether a trigger x request is generated or not. In case of read, the value 0x00000000 is returned. 00_B No modification of current state 01_B Pending trigger x is cleared 10_B Trigger x is set 11_B No modification of current state

Interrupt Output Register n

INTOUTn (n=0-9)

Interrupt Output Register n **(09FE9C$_H$+n*4)** **Application Reset Value: 0000 0000$_H$**

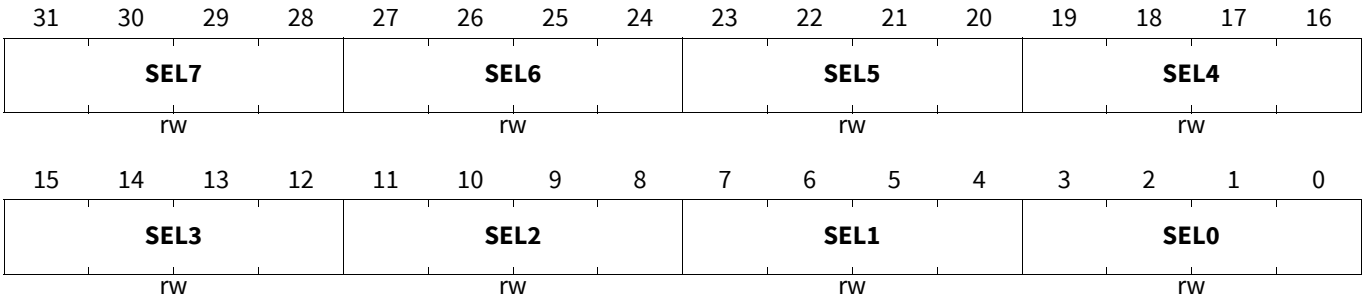


Field	Bits	Type	Description
INT0	1:0	w	**Interrupt Trigger Request 0** This bit field defines whether an interrupt request 0 is generated or not.In case of read, the value 0x00000000 is returned 00_B No modification of current state 01_B Pending interrupt request 0 is cleared 10_B Interrupt request 0 is set 11_B No modification of current state
0	31:2	r	**Reserved** Read as 0, shall be written with 0.

Trigger Output Select Register

MCSTRIGOUTSEL

Trigger Output Select Register **(09FEC4$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
SELk (k=0-7)	4*k+3:4*k	rw	**Selects which MCS triggers go to FCkBFDAT/SEL**

MCS to ADC Triggers

The triggers x0 can be used either as GPIO outputs or as gated trigger inputs for some ADC channels.

Table 167 MCSx (x=0..9) Trigger Assignments

MCS Trigger Output	Port	ADC Input	Control
MCS_TRIG0	n.A.	FC0BFDAT	TRIGOUTx.TRIG0
MCS_TRIG1	n.A.	FC0BFSEL	TRIGOUTx.TRIG1
MCS_TRIG2	n.A.	FC1BFDAT	TRIGOUTx.TRIG2
MCS_TRIG3	n.A.	FC1BFSEL	TRIGOUTx.TRIG3
MCS_TRIG4	n.A.	FC2BFDAT	TRIGOUTx.TRIG4
MCS_TRIG5	n.A.	FC2BFSEL	TRIGOUTx.TRIG5
MCS_TRIG6	n.A.	FC3BFDAT	TRIGOUTx.TRIG6
MCS_TRIG7	n.A.	FC3BFSEL	TRIGOUTx.TRIG7
MCS_TRIG8	n.A.	FC4BFDAT	TRIGOUTx.TRIG8
MCS_TRIG9	n.A.	FC4BFSEL	TRIGOUTx.TRIG9
MCS_TRIG10	n.A.	FC5BFDAT	TRIGOUTx.TRIG10
MCS_TRIG11	n.A.	FC5BFSEL	TRIGOUTx.TRIG11
MCS_TRIG12	n.A.	FC6BFDAT	TRIGOUTx.TRIG12
MCS_TRIG13	n.A.	FC6BFSEL	TRIGOUTx.TRIG13
MCS_TRIG14	n.A.	FC7BFDAT	TRIGOUTx.TRIG14
MCS_TRIG15	n.A.	FC7BFSEL	TRIGOUTx.TRIG15

Note:

1. *The MCSx number configuration is product specific, please check the **GTM Configuration by AURIX TC3xx Product** for the available MCS channels.*
2. *The number of Fast ADC channels (FCxBFDAT/SEL) is product specific (please refer to the EVADC chapter for more details), therefore, for smaller TC3x derivatives, not all the MCS_TRIGx are used/connected.*

Table 168 MCSx (x=0..9) Interrupt Trigger Assignments

MCS Trigger Output	Interrupt Router	Control
MCSINTSTAT.MCSn0	SRC_GTMMCSWx0	INTOUTx.INT0

MCS Interrupt Status Register

MCSINTSTAT

MCS Interrupt Status Register (09FEC8$_H$) Application Reset Value: 0000 0000$_H$

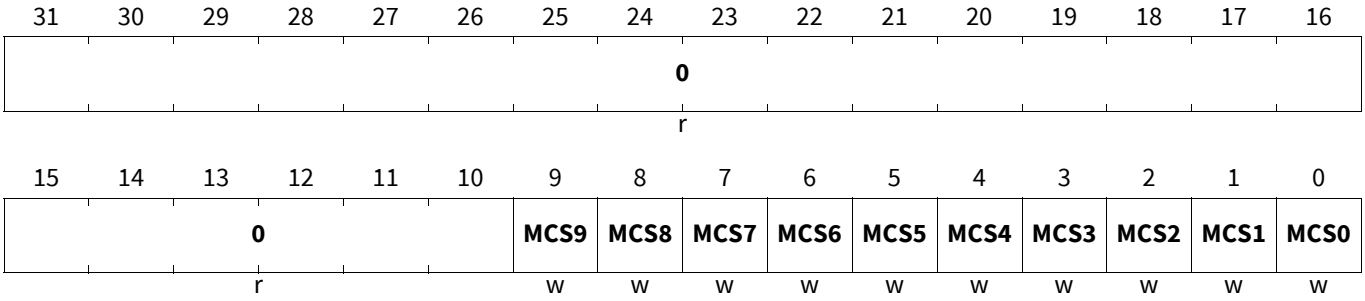
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						MCS90	MCS80	MCS70	MCS60	MCS50	MCS40	MCS30	MCS20	MCS10	MCS00
r						rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
MCSn0 (n=0-9)	n	rh	**MCSn RAM0 Interrupt 0 Status Flag** The requested interrupt is SRC_GTMMCSWn0. This bit is cleared whenbit MCSINTCLR.MCSn is set. 0_B No interrupt was requested 1_B An interrupt was requested
0	31:10	r	**Reserved** Read as 0, shall be written with 0.

MCS Interrupt Clear Register

MCSINTCLR
MCS Interrupt Clear Register **($09FECC_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
MCSx (x=0-9)	x	w	**MCSn RAM0 Interrupt 0 Status Clear Bit** This bit is always read as zero. 0_B No action 1_B Bit MCSINTSTAT.MCSn0 is cleared
0	31:10	r	**Reserved** Read as 0, shall be written with 0.

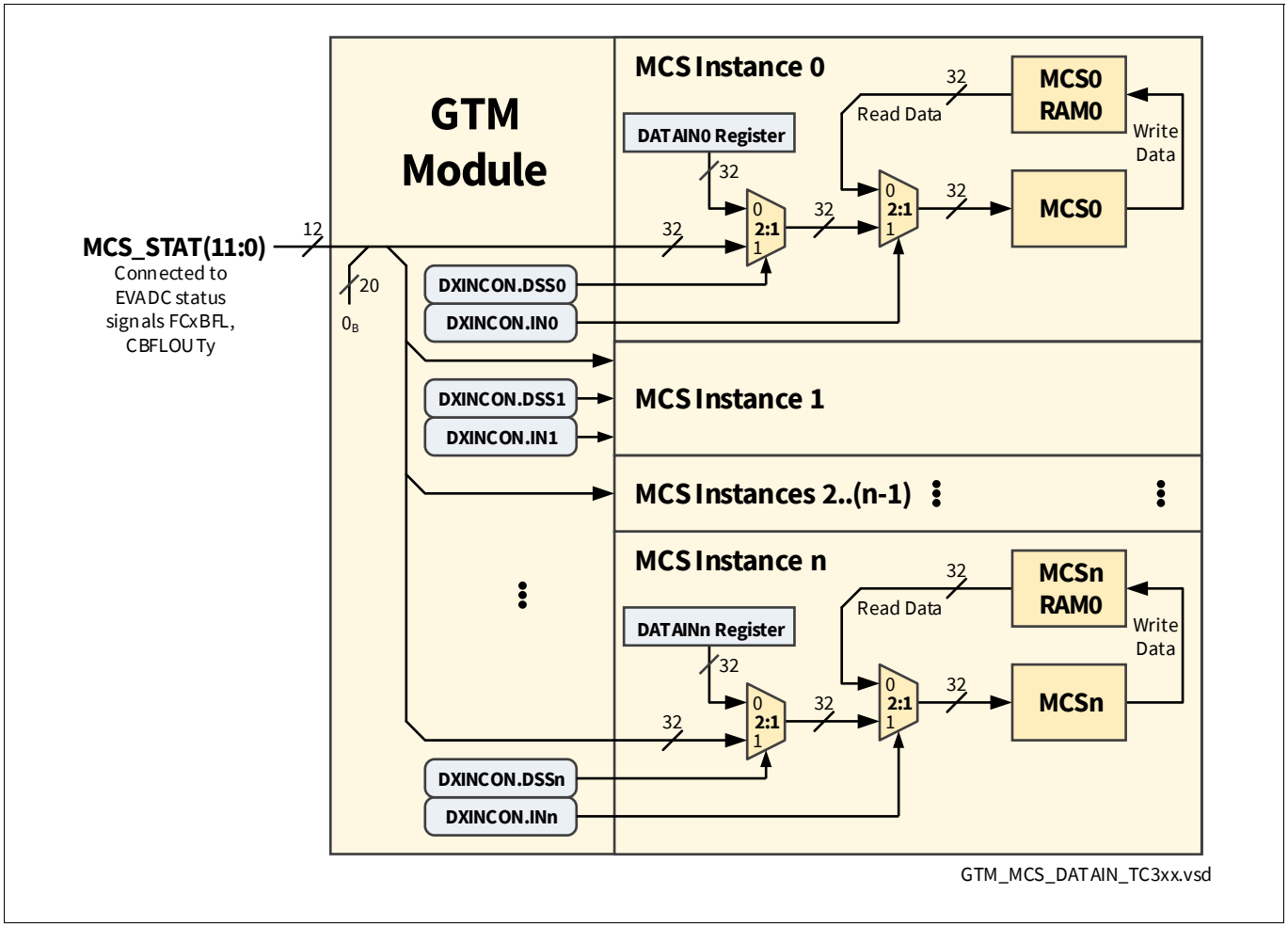


Figure 153 MCS Data Input Path

The DATAIN registers enable the GTM internal sequencers to access data provided by other masters in the system directly in the sequencer data flow without ARU interaction. In addition, the status of dedicated system triggers can also be observed here. The DATAIN registers are mirrored to address 0028_H for sequencer RAM0. Whether a read of address 0028_H fetches data from the RAM or the DATAIN register, is controlled by the bus-accessible register DXINCON in central for all sequencers and there RAMs. In addition, register DATAIN contains the control whether the 12 LSBs of the DATAIN register contain data or the status of the EVADC module (see **Table 169**). This need to be configured only once when setting up the GTM with the remainder of the system.

Data Exchange Input Control Register

DXINCON
Data Exchange Input Control Register (09FED0$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						DSS9	DSS8	DSS7	DSS6	DSS5	DSS4	DSS3	DSS2	DSS1	DSS0
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						IN9	IN8	IN7	IN6	IN5	IN4	IN3	IN2	IN1	IN0
r						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
INx (x=0-9)	x	rw	**Input 0x Control** This bit defines whether register DATAINx is read from the MCS instead of RAM0 or not. 0_B RAM0 memory is read 1_B Register DATAINx is read
DSSx (x=0-9)	x+16	rw	**Data Source Select 0x Control** This bit defines whether the 12 LSB of the read operation directed to register DATAIN0x deliver the register content or the state of 12 inputs. 0_B Register DATAINx[11:0] is read 1_B MCS_STATx[11:0] is read
0	15:10, 31:26	r	**Reserved** Read as 0, shall be written with 0.

Data Input n Register

DATAINn (n=0-9)
Data Input n Register **(09FED4$_H$+n*4)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															
rw															

Field	Bits	Type	Description
DATA	31:0	rw	**Data** This bit field holds the data for the RAM0 replacement.

Input connections

Table 169 MCS Data Input Signal Connections

MCS Status Input	Input
MCS_STAT0	FC0BFL
MCS_STAT1	FC1BFL
MCS_STAT2	FC2BFL
MCS_STAT3	FC3BFL
MCS_STAT4	FC4BFL
MCS_STAT5	FC5BFL
MCS_STAT6	FC6BFL
MCS_STAT7	FC7BFL
MCS_STAT8	CBFLOUT0

Table 169 MCS Data Input Signal Connections (cont'd)

MCS Status Input	Input
MCS_STAT9	CBFLOUT1
MCS_STAT10	CBFLOUT2
MCS_STAT11	CBFLOUT3
all other	Reserved

Note: The number of FCxBFL and CBFLOUTx is product specific. Please refer to the specific device appendix, section "EVADC to GTM Connections", for information on which MCS_STATx inputs are connected to FCxBFL/CBFLOUTy signals.

Note: The correct working behavior is guaranteed only if the GTMDIV is set to 0001_B (default value).

28.25.17 SCU Connections

This section summarizes the connections to the SCU (System Control Unit). The TOM3 connection does not exist on the TC37x device and below.

Table 170 GTM to SCU Connections

GTM Output	SCU Input
TOM0_12	Input41 (ERS4.REQ4B)
TOM1_12	Input51 (ERS5.REQ5B)
TOM2_12	Input65 (ERS6.REQ6F)
TOM3_12	Input75 (ERS7.REQ7F)

28.25.18 HSM Connections

This section summarizes the connections to the HSM.

Table 171 GTM to HSM Connections

GTM Output	HSM Input
GTM_TIM0_IRQ4 (TIM0_CH4)	EXT_INT0
GTM_TIM0_IRQ5 (TIM0_CH5)	EXT_INT1
GTM_TIM0_IRQ6 (TIM0_CH6)	EXT_INT2
GTM_TIM0_IRQ7 (TIM0_CH7)	EXT_INT3
GTM_ATOM0_IRQ2 (ATOM0_CH4, ATOM0_CH5)	EXT_INT4
GTM_ATOM0_IRQ3 (ATOM0_CH6, ATOM0_CH7)	EXT_INT5
GTM_MCS0_IRQ0 (MCS0_CH0)	EXT_INT6
GTM_MCS0_IRQ1 (MCS0_CH1)	EXT_INT7

Note: In case of 2 interrupt pulses with less than 3 SPB clock cycles in between, the second one will not be routed to the HSM

28.25.19 GTM Debug Interface

28.25.19.1GTM OCDS Interface

GTM supports software development and debugging with the following features:

- Read/write access to all GTM registers and memories while running or suspended
- Tool read access via DAP/Cerberus are per default non-destructive (controlled via ODA register)
- Selected GTM signals can be routed to device pins
- GTM can be suspended by OCDS (controlled via OCS register)
- Single stepping of MCS channels by using suspend and OTGS
- MCDS trace support for IO signals, MCS, ARU, DPLL and TBU

28.25.19.1.1 GTM Suspend and Single Stepping

When GTM is suspended, all the activities are stopped, but read accesses are possible for evaluating the internal state, whereas write accesses will have no effect on registers. Suspend is globally controlled in OTGS and GTM locally with the OCS register.

For specific and limited MCS software development cases, the suspend functionality can be used by a debugger for single stepping of MCS channels in the following way:

- GTM is suspended
- MCS clock frequency is configured to be the same as SPB clock frequency
- The OTGS TL timer is used by the debugger to generate clock pulses
- The tool checks after each clock pulse whether the next instruction has been reached already

This sequence can be executed by the tool transparent to the user. This single-stepping approach has limitations for specific instructions and ARU accesses, and it is not a replacement for trace based debugging methods.

28.25.19.1.2 OCDS Trigger Bus (OTGB) Interface

Figure 154 shows a block diagram of the GTM with the trigger and trace connections to pins and MCDS. The OTGB0/1 trigger busses are connected via the regular OTGB infrastructure to device pins and MCDS. The GTM specific OTGBM0/1 busses directly to MCDS. OTGB0/1 are 16-bit trigger busses, OTGBM0/1 are 32-bit wide. Further information about OTGB0/1 is available in the OTGM (OCDS Trigger MUX) section of the OCDS chapter.

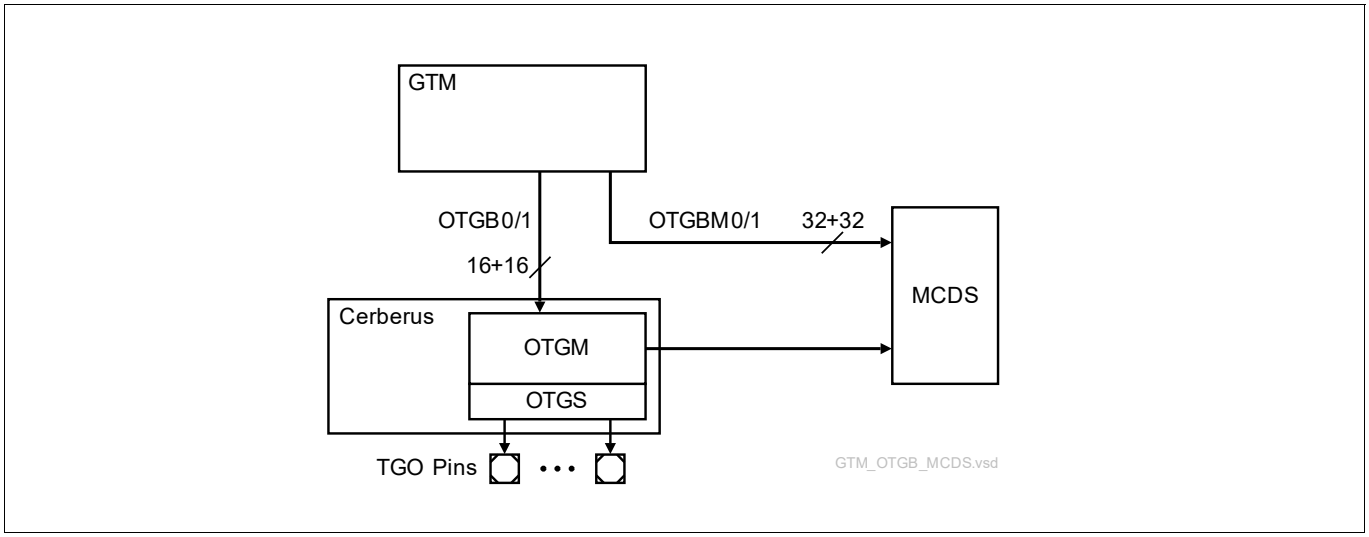


Figure 154 GTM Trigger and Trace Connections

GTM Trigger and Trace Features

- IO signals
  - TIM input signals (after filter)
  - TOM output signals
  - ATOM output signals
  - SPE NIPD and DIR signals
  - Groups of 8 signals (e.g. all inputs of a specific TIM, etc.)
  - Two or four arbitrary groups using OTGB0/1
  - Signals can be used as triggers or processed with OTGS
  - Signals can be routed to TGO device pins via OTGS
- MCS
  - Channel number, PC, and data access type (read or write)[1)]
  - Optional data value on OTGBM1
  - In parallel to IO signal tracing
- ARU
  - ARU_DBG_DATA0/1_H/L (time multiplexed)
  - In parallel to IO signal tracing
  - In parallel to MCS channel number, PC, and data access type tracing
- DPLL
  - TASI and SASI signals as IO signals
  - Accesses to RAM1A/1BC/2 (only one at a time) in parallel to IO signal tracing
- TBU
  - Triggers for individual TBU comparators as IO signals
  - Trace one selected TBU in parallel to IO signal tracing

1) The data captured by EMEM is word aligned (0,1,2..), while MCS PC increments are incremented by 4 (0,4,8,0xC...)

The GTM module has one 16-bit and twelve 32-bit Trigger Sets (**Table 172**), which are selected with the OTSS register.

Table 172 GTM Trigger Sets

Trigger Set	Details
TS16_IOS Trigger Set IO and Other Signals	**Table 174**
TS32_MCA MCS Channel, Address and PC	**Table 175**
TS32_MCD MCS Data	**Table 176**
TS32_ARU Trigger Set ARU	**Table 177**
TS32_DRA1A/1BC/2 DPLLRAM Access Address (3 Sets)	**Table 178**
TS32_DRD1A/1BC/2 DPLLRAM Access Data (3 Sets)	**Table 179**
TS32_TTB0/1/2/3 TBU Time Stamps (3 Sets)	**Table 180**

Table 173 shows all Trigger Set mapping options. OTGB0/1 can be set independently of OTGBM0/1.

Table 173 Trigger Set Mapping Options

OTGB0	OTGB1	OTGBM0	OTGBM1
TS16_IOS	TS16_IOS	TS32_MCA	TS32_MCD
		TS32_DRA1A/1BC/2	TS32_DRD1A/1BC/2
		TS32_ARU	TS32_ARU
		TS32_TTB0/1/2/3	

The OTGB0/1 signals relate to the GTM internal clock, which can be slower or equal to the OTGB clock.

Note: The above trigger sets only exist, if the corresponding resources are existing on the device.

Note: In case of GTM kernel reset (GTM module reset via BPI), the output signals towards the Cerberus (OTBG0/1) and towards the MCDS (OTGBM0/1) will be '1' for 3 SPB clock cycles due to the reset isolation functionality (usually, outputs are driven to '0' during this period). This will assure that a GTM kernel reset can be determined at the debug system.

IO and Other Signals Trigger Sets

The IO Trigger Sets consist of the most important TIM, TOM and ATOM signals in groups of 8. In addition, there are two groups of SPE signals and one group for miscellaneous signals. The multiplexer allows to map arbitrary and different signal groups to the high and the low byte of a 16-bit Trigger Set. In addition, it is possible to use one or two 16-bit Trigger Sets with this flexibility. All this is controlled with OTSC0 register.

Please note that the TS16_IOS signals are sampled with the OTGB clock, which can be slower than the GTM clock. For normal clock configurations, this difference will not be more than a factor of two, which means a max. delay of one GTM clock.

Table 174 TS16_IOS Trigger Set IO and Other Signals

Bits	Name	Description	
[7:0]	SG0	A group of 8 signals from a selected TIM, TOM, ATOM or SPE	
		TIM	All 8 input signals after filter F_OUT
		TOML	Lower 8 output signals TOM_OUT
		TOMH	Higher 8 output signals TOM_OUT
		ATOM	All 8 output signals ATOM_CHx_OUT
		SPE	For OTSC0.BxxMI Module Instance 0: Bits 0,1: SPE0_NIPD, SPE_DIR0 Bits 2,3: SPE1_NIPD, SPE_DIR1 Bits 4,5: SPE2_NIPD, SPE_DIR2 (00_B if GTM has no SPE2) Bits 6,7: SPE3_NIPD, SPE_DIR3 (00_B if GTM has no SPE3) For OTSC0.BxxMI Module Instance 1: Bits 0,1: SPE4_NIPD, SPE_DIR4 (00_B if GTM has no SPE4) Bits 2,3: SPE5_NIPD, SPE_DIR5 (00_B if GTM has no SPE5) **others**, reserved (value is 0)
		MISC	Bit 0: DPLL TASI (if DPLL exists) Bit 1: DPLL SASI (if DPLL exists) Bit 4: TBU0 trigger (OTBU0T) Bit 5: TBU1 trigger (OTBU1T) Bit 6: TBU2 trigger (OTBU2T) Bit 7: TBU3 trigger (OTBU3T) (if TBU channel 3 exists) **others**, reserved
[15:8]	SG1	Independent selection with same options as for bits [7:0]	

MCS Trigger Sets

There are two MCS Trigger Sets: TS32_MCA (**Table 175**) and TS32_MCD (**Table 176**). TS32_MCA allows to observe the PC of all or a selected MCS channel(s). If only one channel is selected, the TS32_MCA value will stay stable until the MCS executes this channel again. In case of data access by CPU or by a channel, it represents the data address. With TS32_MCD, the associated data value and even the fetched instruction opcode can be traced. Usually, the latter is avoided to reduce the trace bandwidth. All this is controlled with OTSC1.

Table 175 TS32_MCA MCS Channel, Address and PC

Bits	Name	Description
[13:0]	ADDR	PC of MCS channel or address of data access (32-bit word address)
[15:14]	AQ	ADDR Qualifier 0_H PC of MCS channel 1_H Address of data read 2_H Address of data write 3_H Reserved
[19:16]	CH	MCS channel number
[31:20]		Reserved

Table 176 TS32_MCD MCS Data

Bits	Description
[31:0]	Read or written data value of current channel instruction

ARU Trigger Sets

There are four 29-bit ARU data words with debug information (ARU_DBG_DATA0_L/H and ARU_DBG_DATA1_L/H). These words may get valid in the same GTM kernel clock cycle, but they will never change in consecutive cycles. This property is used to transmit them with time multiplexing over the OTGBM bus.

Table 177 TS32_ARU Trigger Set ARU

Bits	Name	Description
[28:0]	DATA	ARU_DBG_DATA
29		Reserved
[31:30]	DQ	DATA Qualifier 0_H Is ARU_DBG_DATA0_L 1_H Is ARU_DBG_DATA0_H 2_H Is ARU_DBG_DATA1_L 3_H Is ARU_DBG_DATA1_H

DPLL Trigger Sets

The TASI and SASI signals are part of the MISC signal group of TS16_IOS (**Table 174**).

The following tables **Table 178** and **Table 179** define six additional DPLL Trigger Sets for observing the DPLL RAM accesses. Only one of the three DPLL (1A, 1B+C, 2) RAMs can be observed at a time. The Trigger Set selection is done with OTSS.

Table 178 TS32_DRA1A/1BC/2 DPLLRAM Access Address

Bits	Name	Description
[13:0]	ADDR	Address of data access. Range starts with address 0 for the selected RAM (1A, 1B+C or 2). Effective width depends on selected RAM
[15:14]	AQ	ADDR Qualifier 0_H DPLL data read access 1_H DPLL data write access 2_H CPU read access 3_H CPU write access
[31:16]		Reserved

Table 179 TS32_DRD1A/1BC/2 DPLLRAM Access Data

Bits	Name	Description
[23:0]	DATA	Read or written data
[31:24]		Reserved

TBU Trigger Sets

Table 180 defines the Trigger Sets for the observation of the different time bases. The Trigger Set selection (one at a time) is done with OTSS.

Table 180 TS32_TTB0/1/2/3 TBU Time Stamps

Bits	Name	Description
[26:0]	TS	Current TBU_TS0/1/2/3 time stamp. Effective width depends on selected time base.
[31:27]		Reserved

28.25.19.1.3 GTM Debug Registers

All debug control registers are cleared by Debug Reset and by each System Reset when OCDS is disabled. It is not changed by System Reset when OCDS is enabled.

OCDS Trigger Bus (OTGB)

Accesses are only supported for byte, half word and word data, and require Supervisor Mode.

OCDS Trigger Set Select Register

Note: OCS.SUSSTA is set to 1 (module is suspended) up to 2 gtm_clk cycles earlier before GTM changes into soft suspend mode.

OTSS

OCDS Trigger Set Select Register **(09FD28$_H$)** **Debug Reset Value: 0000 0000$_H$**

Field	Bits	Type	Description
OTGB0	3:0	rw	**Trigger Set for OTGB0** 0_H No Trigger Set selected 1_H Trigger Set TS16_IOS (**Table 174**) **others**, reserved
OTGB1	11:8	rw	**Trigger Set for OTGB1** 0_H No Trigger Set selected 1_H Trigger Set TS16_IOS (**Table 174**) **others**, reserved

Field	Bits	Type	Description
OTGBM0	19:16	rw	**Trigger Set for OTGBM0** 0_H No Trigger Set selected 1_H Trigger Set TS32_MCA (**Table 175**)Exists only if MCS exists, otherwise 0 2_H Trigger Set TS32_DRA1A (**Table 178**)Exists only if DPLL exists, otherwise 0 3_H Trigger Set TS32_DRA1BC (**Table 178**)Exists only if DPLL exists, otherwise 0 4_H Trigger Set TS32_DRA2 (**Table 178**)Exists only if DPLL exists, otherwise 0 5_H Trigger Set TS32_ARU (**Table 177**) 6_H Trigger Set TS32_TTB0 (**Table 180**) 7_H Trigger Set TS32_TTB1 (**Table 180**) 8_H Trigger Set TS32_TTB2 (**Table 180**) 9_H Trigger Set TS32_TTB3 (**Table 180**)Exists only if TBU channel 3 exists, otherwise 0 **others**, Reserved
OTGBM1	27:24	rw	**Trigger Set for OTGBM1** 0_H No Trigger Set selected No Trigger Set selected 1_H Trigger Set TS32_MCD (**Table 176**)Exists only if MCS exists, otherwise 0 2_H Trigger Set TS32_DRD1A (**Table 179**)Exists only if DPLL exists, otherwise 0 3_H Trigger Set TS32_DRD1BC (**Table 179**)Exists only if DPLL exists, otherwise 0 4_H Trigger Set TS32_DRD2 (**Table 179**)Exists only if DPLL exists, otherwise 0 5_H Trigger Set TS32_ARU (**Table 177**) **others**, Reserved
0	7:4, 15:12, 23:20, 31:28	r	**Reserved** Read as 0, shall be written with 0.

OCDS Trigger Set Control 0 Register

Note: OCS.SUSSTA is set to 1 (module is suspended) up to 2 gtm_clk cycles earlier before GTM changes into soft suspend mode.

OTSC0

OCDS Trigger Set Control 0 Register **(09FD2C$_H$)** **Debug Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
B1HMI				**0**	**B1HMT**			**B1LMI**				**0**	**B1LMT**		
rw				r	rw			rw				r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B0HMI				**0**	**B0HMT**			**B0LMI**				**0**	**B0LMT**		
rw				r	rw			rw				r	rw		

Field	Bits	Type	Description
B0LMT	2:0	rw	**OTGB0 TS16_IOS Low Byte Module Type** 000_B No Module selected 001_B TIM (F_OUT[7:0]) 010_B TOML (TOM_OUT[7:0]) 011_B TOMH (TOM_OUT[15:8]) 100_B ATOM (ATOM_OUT[7:0]) 101_B SPE signals (**Table 174**) 110_B MISC signals (**Table 174**) **others**, Reserved
B0LMI	7:4	rw	**OTGB0 TS16_IOS Low Byte Module Instance** Index of the module instance. Index starts with 0, the max. value depends on GTM configuration and module type. For SPE signals, there is a specific mapping (**Table 174**). For MISC signals, this index is ignored.
B0HMT	10:8	rw	**OTGB0 TS16_IOS High Byte Module Type** 000_B No Module selected 001_B TIM (F_OUT[7:0]) 010_B TOML (TOM_OUT[7:0]) 011_B TOMH (TOM_OUT[15:8]) 100_B ATOM (ATOM_OUT[7:0]) 101_B SPE signals (**Table 174**) 110_B MISC signals (**Table 174**) **others**, Reserved
B0HMI	15:12	rw	**OTGB0 TS16_IOS High Byte Module Instance** Index of the module instance. Index starts with 0, the max. value depends on GTM configuration and module type. For SPE signals, there is a specific mapping (**Table 174**). For MISC signals, this index is ignored.
B1LMT	18:16	rw	**OTGB1 TS16_IOS Low Byte Module Type** 000_B No Module selected 001_B TIM (F_OUT[7:0]) 010_B TOML (TOM_OUT[7:0]) 011_B TOMH (TOM_OUT[15:8]) 100_B ATOM (ATOM_OUT[7:0]) 101_B SPE signals (**Table 174**) 110_B MISC signals (**Table 174**) **others**, Reserved

Field	Bits	Type	Description
B1LMI	23:20	rw	**OTGB1 TS16_IOS Low Byte Module Instance** Index of the module instance. Index starts with 0, the max. value depends on GTM configuration and module type. For SPE signals, there is a specific mapping (**Table 174**). For MISC signals, this index is ignored.
B1HMT	26:24	rw	**OTGB1 TS16_IOS High Byte Module Type** 000_B No Module selected 001_B TIM (F_OUT[7:0]) 010_B TOML (TOM_OUT[7:0]) 011_B TOMH (TOM_OUT[15:8]) 100_B ATOM (ATOM_OUT[7:0]) 101_B SPE signals (**Table 174**) 110_B MISC signals (**Table 174**) **others**, Reserved
B1HMI	31:28	rw	**OTGB1 TS16_IOS High Byte Module Instance** Index of the module instance. Index starts with 0, the max. value depends on GTM configuration and module type. For SPE signals, there is a specific mapping (**Table 174**). For MISC signals, this index is ignored.
0	3, 11, 19, 27	r	**Reserved** Read as 0, shall be written with 0.

OCDS Trigger Set Control 1 Register

Note: OCS.SUSSTA is set to 1 (module is suspended) up to 2 gtm_clk cycles earlier before GTM changes into soft suspend mode.

OTSC1

OCDS Trigger Set Control 1 Register **($09FD30_H$)** **Debug Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						MOE	0	MI				MCS			
r						rw	r	rw				rw			

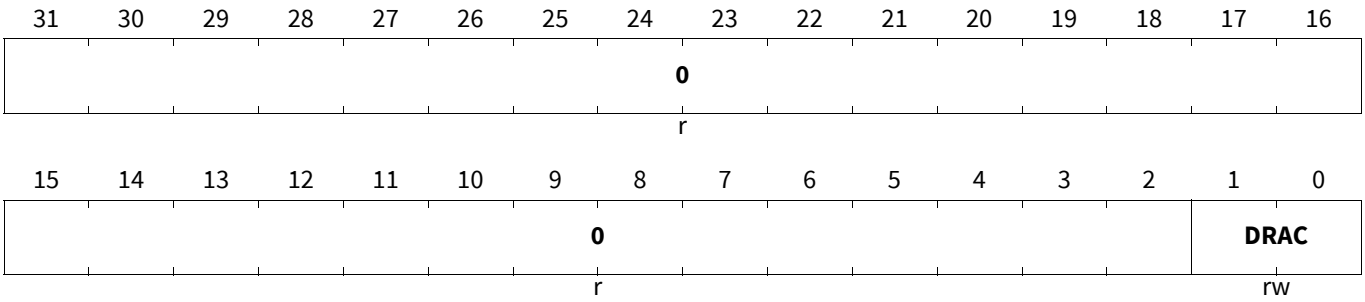
Field	Bits	Type	Description
MCS	3:0	rw	**MCS Channel Selection** Required by TS32_MCA and TS32_MCD. 0_H MCS Channel 0 1_H MCS Channel 1 2_H MCS Channel 2 3_H MCS Channel 3 4_H MCS Channel 4 5_H MCS Channel 5 6_H MCS Channel 6 7_H MCS Channel 7 F_H All MCS Channels **others**, Reserved
MI	7:4	rw	**MCS Instance** Required by TS32_MCA and TS32_MCD. Index of the MCS instance. Index starts with 0, the max. value depends on the GTM configuration.
MOE	9	rw	**MCS Opcode Trace Enable** Required by TS32_MCD. 0_B Trigger Set TS32_MCD excludes opcode fetches 1_B Trigger Set TS32_MCD includes opcode fetches
0	8, 31:10	r	**Reserved** Read as 0, shall be written with 0.

OCDS Debug Access Register

Note: OCS.SUSSTA is set to 1 (module is suspended) up to 2 gtm_clk cycles earlier before GTM changes into soft suspend mode.

ODA

OCDS Debug Access Register **($09FD34_H$)** **Debug Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DRAC	1:0	rw	**Debug Read Access Control** Controls Debug Read Access (DRA) of registers with DRA behavior. 00_B Normal read access for all bus masters, but debug read access for Cerberus 01_B Normal read access for all bus masters 10_B Debug read access for all bus masters 11_B Debug read access for all bus masters
0	31:2	r	**Reserved** Read as 0, shall be written with 0.

Note: When the debug read access is enabled, the registers listed below will behave as follows:

3. *AFD[i]_CH[x]_BUF_ACC: Reading these registers will read only the first data available in the fifo. The fifo pointers and the flags (watermark, fifo empty) will not be changed. On multiple reads every time the first data in the fifo will be read.*
4. *TIM[i]_CH[x]_ECNT: Reading these registers will not reset the register (auto clear) when a debug access is active.*
5. *ATOM[i]_CH[x]_SR[0,1]: Reading these registers in ATOM SOMC mode when captured data are available is possible. While debug read access is enabled, read accesses will not enable the setup of a new compare action compared to the behavior of regular read accesses.*
6. *TIM[i]_CH[x]_GPR0/1: Reading these register will reset the ECNT counter.*

OCDS TBU0 Trigger Register

Note: OCS.SUSSTA is set to 1 (module is suspended) up to 2 gtm_clk cycles earlier before GTM changes into soft suspend mode.

OTBU0T
OCDS TBU0 Trigger Register **($09FD18_H$)** **Debug Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**CM**		**0**	**CV**										
r		rw		r	rw										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV															
rw															

Field	Bits	Type	Description
CV	26:0	rw	**Compare Value** This value is compared to the TBU_CH0_BASE register. As long as both match, the associated TS16_IOS.MISC bit (**Table 174**) is active.
CM	29:28	rw	**Compare Mode** 00_B Disabled 01_B Compare lower 24 bits 10_B Compare upper 24 bits 11_B Compare all 27 bits

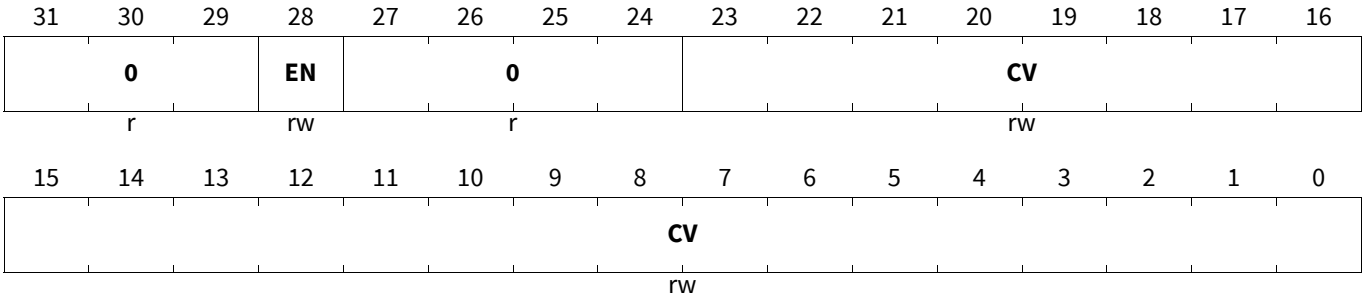
Field	Bits	Type	Description
0	27, 31:30	r	**Reserved** Read as 0, shall be written with 0.

OCDS TBU1 Trigger Register

Note: OCS.SUSSTA is set to 1 (module is suspended) up to 2 gtm_clk cycles earlier before GTM changes into soft suspend mode.

OTBU1T

OCDS TBU1 Trigger Register **(09FD1C$_H$)** **Debug Reset Value: 0000 0000$_H$**



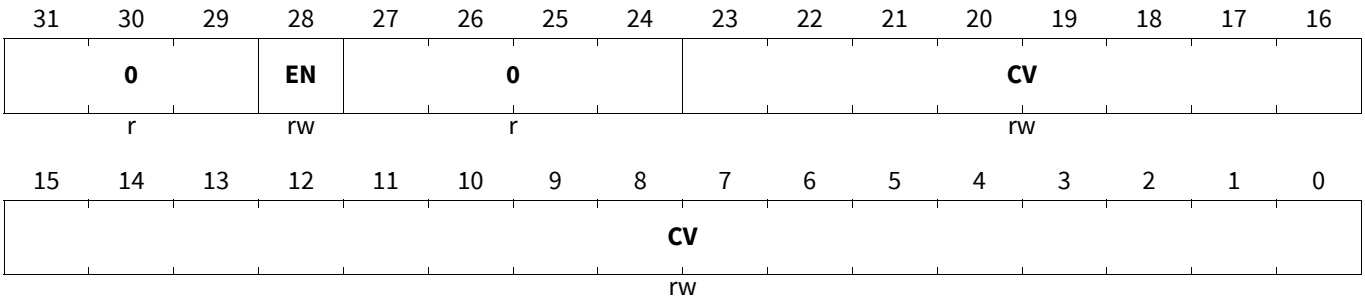
Field	Bits	Type	Description
CV	23:0	rw	**Compare Value** This value is compared to the TBU_CH1_BASE register. As long as both match, the associated TS16_IOS.MISC bit (**Table 174**) is active.
EN	28	rw	**Enable** 0_B Disabled 1_B Enabled
0	27:24, 31:29	r	**Reserved** Read as 0, shall be written with 0.

OCDS TBU2 Trigger Register

Note: OCS.SUSSTA is set to 1 (module is suspended) up to 2 gtm_clk cycles earlier before GTM changes into soft suspend mode.

OTBU2T

OCDS TBU2 Trigger Register **(09FD20$_H$)** **Debug Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CV	23:0	rw	**Compare Value** This value is compared to the TBU_CH2_BASE register. As long as both match, the associated TS16_IOS.MISC bit (**Table 174**) is active.
EN	28	rw	**Enable** 0_B Disabled 1_B Enabled
0	27:24, 31:29	r	**Reserved** Read as 0, shall be written with 0.

OCDS TBU3 Trigger Register

Note: OCS.SUSSTA is set to 1 (module is suspended) up to 2 gtm_clk cycles earlier before GTM changes into soft suspend mode.

This register only exists, if TBU channel 3 exists.

OTBU3T

OCDS TBU3 Trigger Register **($09FD24_H$)** **Debug Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**EN**	**0**				**CV**							
r			rw	r				rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV															
rw															

Field	Bits	Type	Description
CV	23:0	rw	**Compare Value** This value is compared to the TBU_CH3_BASE register. As long as both match, the associated TS16_IOS.MISC bit (**Table 174**) is active.
EN	28	rw	**Enable** 0_B Disabled 1_B Enabled
0	27:24, 31:29	r	**Reserved** Read as 0, shall be written with 0.

28.25.20 GTM Application constraints and limitations

Depending on the specific application, the GTM constraints and limitations listed below need to be taken into account.

Table 181 GTM IP Application Constraints

#	Modules	Description	Required Value	Effect when not considered
1	DPLL	Increment duration (time between two valid inputs of the DPLL: TRIGGER/STATE)	> 23.4 ms	The calculation of at least 3 actions per increment in addition to a proper micro-tick generation (Sub_inc) could not be ensured for a system clock frequency of 100 MHz
2	DPLL	TBU_BASE_CH0 difference for two successive valid input events of the DPLL	256 (for the chosen input resolution)	Overflow in the calculation of the reciprocal values; the increment prediction and all other computations of the DPLL would become wrong
3	DPLL	CMU_CLK0 frequency	less or equal to half the frequency of system clock	The fast pulse correction in automatic end mode could be wrong
4	MCS	Worst Case Execution Time of an N-cycle instruction	less or equal to 9*N system clock periods	MCS program execution may be too long for the application's requirements
5	MCS	If MCS scheduling is configured in round robin mode, the MCS features deterministic program execution (the program of MCS channel x does not modify execution time of MCS channel y). This determinism no longer exists, if an MCS channel of MCS instance 0 is accessing the DPLL RAM with its bus master interface.		No channel independent execution time
6	TOM/ ATOM	If channel is triggered by preceding channel via TRIG_[x-1] signal, the selected FX_CLK/CMU_CLK of both channels has to be the same	Identical FX_CLK/CMU_CLK	The trigger of preceding channel may be lost

Table 181 GTM IP Application Constraints (cont'd)

#	Modules	Description	Required Value	Effect when not considered
7	TIM	If a TIM channel x uses ARU transfer with enabled TDU, the minimal time TIN between two subsequent measurement cycles has to be greater than the maximum time TSAMPLE between two subsequent ARU read request events on TIM channel x	TIN > TSAMPLE (FIFO: TSAMPLE = worst case ARU round trip time, MCS: TSAMPLE = max. time between two (N)ARD(I) instructions for TIM channel x)	The ARU destination of TIM channel x cannot distinguish between a measurement cycle overflow and a time-out with subsequent valid measurement cycle
8	ARU	The ARU round trip time in pure round robin mode is device specific	(ARU_CADDR_END +1) times slow clusters clock period with ARU_CADDR_END = 125 (default value)	If dynamic mode is activated with CLK_WAIT>0, the ARU round trip time becomes longer, and the value is undefined

Table 182 GTM Wrapper Constraints

#	Modules	Description	Required Value	Effect when not considered
1	GTM Wrapper	To switch off (and on again) safely the module, the check of the CLC.DISS (i.e while (CLC.DISS==1) is not enough to guarantee that the module has been effectively switched off..	>= 2 x slow GTM clusters clock period.	Watchdog error

GTM Legacy Addresses

In previous GTM specification, a smaller number of instances of the modules ATOM and MCS were included in the address list. The old address areas of these modules are marked with LEG_ (legacy). These LEG_ areas are mapped to the normal areas.

Accesses to these addresses will not result in a bus error.

The following lists all sub-modules of the which have additional legacy base addresses defined by older GTM Generations. The table is for TC39x, devices with a smaller number on the 4th position only include a subset of the listed features.

Table 183 GTM Legacy Address table

CDTM0_DTM0	0x000E4000	DTM0	0x00013000
CDTM0_DTM1	0x000E4040	DTM1	0x00013040
CDTM0_DTM4	0x000E4100	DTM24	0x00013600
CDTM0_DTM5	0x000E4140	DTM25	0x00013640
CDTM1_DTM0	0x000E4400	DTM4	0x00013100
CDTM1_DTM1	0x000E4440	DTM5	0x00013140
CDTM1_DTM4	0x000E4500	DTM26	0x00013680
CDTM1_DTM5	0x000E4540	DTM27	0x000136C0

Table 183 GTM Legacy Address table (cont'd)

CDTM2_DTM0	0x000E4800	DTM8	0x00013200
CDTM2_DTM1	0x000E4840	DTM9	0x00013240
CDTM2_DTM4	0x000E4900	DTM28	0x00013700
CDTM2_DTM5	0x000E4940	DTM29	0x00013740
CDTM3_DTM0	0x000E4C00	DTM12	0x00013300
CDTM3_DTM1	0x000E4C40	DTM13	0x00013340
CDTM3_DTM4	0x000E4D00	DTM30	0x00013780
CDTM3_DTM5	0x000E4D40	DTM31	0x000137C0
CDTM4_DTM0	0x000E5000	DTM16	0x00013400
CDTM4_DTM1	0x000E5040	DTM17	0x00013440
CDTM4_DTM4	0x000E5100	DTM32	0x00013800
CDTM4_DTM5	0x000E5140	DTM33	0x00013840
CDTM5_DTM4	0x000E5500	DTM34	0x00013880
CDTM5_DTM5	0x000E5540	DTM35	0x000138C0
CDTM6_DTM4	0x000E5900	DTM36	0x00013900
CDTM6_DTM5	0x000E5940	DTM37	0x00013940
ATOM0	0x000E8000	LEG_ATOM0	0x0000D000
ATOM1	0x000E8800	LEG_ATOM1	0x0000D800
ATOM2	0x000E9000	LEG_ATOM2	0x0000E000
ATOM3	0x000E9800	LEG_ATOM3	0x0000E800
ATOM4	0x000EA000	LEG_ATOM4	0x0000F000
ATOM5	0x000EA800	LEG_ATOM5	0x0000F800
ATOM6	0x000EB000	LEG_ATOM6	0x00010000
ATOM7	0x000EB800	LEG_ATOM7	0x00010800
ATOM8	0x000EC000	LEG_ATOM8	0x00011000
ATOM9	0x000EC800	LEG_ATOM9	0x00011800
ATOM10	0x000ED000	LEG_ATOM10	0x00012000
ATOM11	0x000ED800	LEG_ATOM11	0x00012800
MCS0	0x000F0000	LEG_MCS0	0x00030000
MCS1	0x000F1000	LEG_MCS1	0x00031000
MCS2	0x000F2000	LEG_MCS2	0x00032000
MCS3	0x000F3000	LEG_MCS3	0x00033000
MCS4	0x000F4000	LEG_MCS4	0x00034000
MCS5	0x000F5000	LEG_MCS5	0x00035000
MCS6	0x000F6000	LEG_MCS6	0x00036000

Table 184 GTM Legacy Address table

Register Label	Register Address	Legacy Register Label	Legacy Address
CCM0_HW_CONF	0x000E21DC	GTM_HW_CONF	0x00000024
CCM1_HW_CONF	0x000E23DC	GTM_HW_CONF	0x00000024
CCM2_HW_CONF	0x000E25DC	GTM_HW_CONF	0x00000024
CCM3_HW_CONF	0x000E27DC	GTM_HW_CONF	0x00000024
CCM4_HW_CONF	0x000E29DC	GTM_HW_CONF	0x00000024
CCM5_HW_CONF	0x000E2BDC	GTM_HW_CONF	0x00000024
CCM0_TIM_AUX_IN_SRC	0x000E21E0	GTM_TIM0_AUX_IN_SRC	0x00000040
CCM1_TIM_AUX_IN_SRC	0x000E23E0	GTM_TIM1_AUX_IN_SRC	0x00000044
CCM2_TIM_AUX_IN_SRC	0x000E25E0	GTM_TIM2_AUX_IN_SRC	0x00000048
CCM3_TIM_AUX_IN_SRC	0x000E27E0	GTM_TIM3_AUX_IN_SRC	0x0000004C
CCM4_TIM_AUX_IN_SRC	0x000E29E0	GTM_TIM4_AUX_IN_SRC	0x00000050
CCM5_TIM_AUX_IN_SRC	0x000E2BE0	GTM_TIM5_AUX_IN_SRC	0x00000054
CCM0_EXT_CAP_EN	0x000E21E4	GTM_EXT_CAP_EN_0	0x0000005C
CCM1_EXT_CAP_EN	0x000E23E4	GTM_EXT_CAP_EN_1	0x00000060
CCM2_EXT_CAP_EN	0x000E25E4	GTM_EXT_CAP_EN_2	0x00000064
CCM3_EXT_CAP_EN	0x000E27E4	GTM_EXT_CAP_EN_3	0x00000068
CCM4_EXT_CAP_EN	0x000E29E4	GTM_EXT_CAP_EN_4	0x0000006C
CCM0_TOM_OUT	0x000E21E8	GTM_TOM0_OUT	0x00000080
CCM1_TOM_OUT	0x000E23E8	GTM_TOM1_OUT	0x00000084
CCM2_TOM_OUT	0x000E25E8	GTM_TOM2_OUT	0x00000088
CCM0_ATOM_OUT	0x000E21EC	GTM_ATOM0_OUT	0x00000098
CCM1_ATOM_OUT	0x000E23EC	---	---
CCM2_ATOM_OUT	0x000E25EC	GTM_ATOM2_OUT	0x0000009C
CCM3_ATOM_OUT	0x000E27EC	---	---
CCM4_ATOM_OUT	0x000E2BEC	GTM_ATOM4_OUT	0x000000A0
CCM5_ATOM_OUT	0x000E2DEC	---	---
ICM_IRQG_MCS0_CI	0x00000720	LEG_ICM_IRQG_MCS0_CI	0x00000648
ICM_IRQG_MCS1_CI	0x00000724	LEG_ICM_IRQG_MCS1_CI	0x0000064C
ICM_IRQG_MCS2_CI	0x00000728	LEG_ICM_IRQG_MCS2_CI	0x00000650
ICM_IRQG_MCS3_CI	0x0000072C	LEG_ICM_IRQG_MCS3_CI	0x00000654
ICM_IRQG_MCS4_CI	0x00000730	LEG_ICM_IRQG_MCS4_CI	0x00000658
CDTM0_DTM0_CTRL	0x000E4000	DTM0_CTRL	0x00013000
CDTM0_DTM0_CH_CTRL1	0x000E4004	DTM0_CH_CTRL1	0x00013004
CDTM0_DTM0_CH_CTRL2	0x000E4008	DTM0_CH_CTRL2	0x00013008
CDTM0_DTM0_CH_CTRL2_SR	0x000E400C	DTM0_CH_CTRL2_SR	0x0001300C
CDTM0_DTM0_PS_CTRL	0x000E4010	DTM0_PS_CTRL	0x00013010

Table 184 GTM Legacy Address table (cont'd)

Register Label	Register Address	Legacy Register Label	Legacy Address
CDTM0_DTM0_CH0_DTV	0x000E4014	DTM0_CH0_DTV	0x00013014
CDTM0_DTM0_CH1_DTV	0x000E4018	DTM0_CH1_DTV	0x00013018
CDTM0_DTM0_CH2_DTV	0x000E401C	DTM0_CH2_DTV	0x0001301C
CDTM0_DTM0_CH3_DTV	0x000E4020	DTM0_CH3_DTV	0x00013020
CDTM0_DTM0_CH_SR	0x000E4024	DTM0_CH_SR	0x00013024
CDTM0_DTM0_CH_CTRL3	0x000E4028	DTM0_CH_CTRL3	0x00013028
CDTM0_DTM1_CTRL	0x000E4040	DTM1_CTRL	0x00013040
CDTM0_DTM1_CH_CTRL1	0x000E4044	DTM1_CH_CTRL1	0x00013044
CDTM0_DTM1_CH_CTRL2	0x000E4048	DTM1_CH_CTRL2	0x00013048
CDTM0_DTM1_CH_CTRL2_SR	0x000E404C	DTM1_CH_CTRL2_SR	0x0001304C
CDTM0_DTM1_PS_CTRL	0x000E4050	DTM1_PS_CTRL	0x00013050
CDTM0_DTM1_CH0_DTV	0x000E4054	DTM1_CH0_DTV	0x00013054
CDTM0_DTM1_CH1_DTV	0x000E4058	DTM1_CH1_DTV	0x00013058
CDTM0_DTM1_CH2_DTV	0x000E405C	DTM1_CH2_DTV	0x0001305C
CDTM0_DTM1_CH3_DTV	0x000E4060	DTM1_CH3_DTV	0x00013060
CDTM0_DTM1_CH_SR	0x000E4064	DTM1_CH_SR	0x00013064
CDTM0_DTM1_CH_CTRL3	0x000E4068	DTM1_CH_CTRL3	0x00013068
CDTM0_DTM4_CTRL	0x000E4100	DTM24_CTRL	0x00013600
CDTM0_DTM4_CH_CTRL1	0x000E4104	DTM24_CH_CTRL1	0x00013604
CDTM0_DTM4_CH_CTRL2	0x000E4108	DTM24_CH_CTRL2	0x00013608
CDTM0_DTM4_CH_CTRL2_SR	0x000E410C	DTM24_CH_CTRL2_SR	0x0001360C
CDTM0_DTM4_PS_CTRL	0x000E4110	DTM24_PS_CTRL	0x00013610
CDTM0_DTM4_CH0_DTV	0x000E4114	DTM24_CH0_DTV	0x00013614
CDTM0_DTM4_CH1_DTV	0x000E4118	DTM24_CH1_DTV	0x00013618
CDTM0_DTM4_CH2_DTV	0x000E411C	DTM24_CH2_DTV	0x0001361C
CDTM0_DTM4_CH3_DTV	0x000E4120	DTM24_CH3_DTV	0x00013620
CDTM0_DTM4_CH_SR	0x000E4124	DTM24_CH_SR	0x00013624
CDTM0_DTM4_CH_CTRL3	0x000E4128	DTM24_CH_CTRL3	0x00013628
CDTM0_DTM5_CTRL	0x000E4140	DTM25_CTRL	0x00013640
CDTM0_DTM5_CH_CTRL1	0x000E4144	DTM25_CH_CTRL1	0x00013644
CDTM0_DTM5_CH_CTRL2	0x000E4148	DTM25_CH_CTRL2	0x00013648
CDTM0_DTM5_CH_CTRL2_SR	0x000E414C	DTM25_CH_CTRL2_SR	0x0001364C
CDTM0_DTM5_PS_CTRL	0x000E4150	DTM25_PS_CTRL	0x00013650
CDTM0_DTM5_CH0_DTV	0x000E4154	DTM25_CH0_DTV	0x00013654
CDTM0_DTM5_CH1_DTV	0x000E4158	DTM25_CH1_DTV	0x00013658
CDTM0_DTM5_CH2_DTV	0x000E415C	DTM25_CH2_DTV	0x0001365C
CDTM0_DTM5_CH3_DTV	0x000E4160	DTM25_CH3_DTV	0x00013660

Table 184 GTM Legacy Address table (cont'd)

Register Label	Register Address	Legacy Register Label	Legacy Address
CDTM0_DTM5_CH_SR	0x000E4164	DTM25_CH_SR	0x00013664
CDTM0_DTM5_CH_CTRL3	0x000E4168	DTM25_CH_CTRL3	0x00013668
CDTM1_DTM0_CTRL	0x000E4400	DTM4_CTRL	0x00013100
CDTM1_DTM0_CH_CTRL1	0x000E4404	DTM4_CH_CTRL1	0x00013104
CDTM1_DTM0_CH_CTRL2	0x000E4408	DTM4_CH_CTRL2	0x00013108
CDTM1_DTM0_CH_CTRL2_SR	0x000E440C	DTM4_CH_CTRL2_SR	0x0001310C
CDTM1_DTM0_PS_CTRL	0x000E4410	DTM4_PS_CTRL	0x00013110
CDTM1_DTM0_CH0_DTV	0x000E4414	DTM4_CH0_DTV	0x00013114
CDTM1_DTM0_CH1_DTV	0x000E4418	DTM4_CH1_DTV	0x00013118
CDTM1_DTM0_CH2_DTV	0x000E441C	DTM4_CH2_DTV	0x0001311C
CDTM1_DTM0_CH3_DTV	0x000E4420	DTM4_CH3_DTV	0x00013120
CDTM1_DTM0_CH_SR	0x000E4424	DTM4_CH_SR	0x00013124
CDTM1_DTM0_CH_CTRL3	0x000E4428	DTM4_CH_CTRL3	0x00013128
CDTM1_DTM1_CTRL	0x000E4440	DTM5_CTRL	0x00013140
CDTM1_DTM1_CH_CTRL1	0x000E4444	DTM5_CH_CTRL1	0x00013144
CDTM1_DTM1_CH_CTRL2	0x000E4448	DTM5_CH_CTRL2	0x00013148
CDTM1_DTM1_CH_CTRL2_SR	0x000E444C	DTM5_CH_CTRL2_SR	0x0001314C
CDTM1_DTM1_PS_CTRL	0x000E4450	DTM5_PS_CTRL	0x00013150
CDTM1_DTM1_CH0_DTV	0x000E4454	DTM5_CH0_DTV	0x00013154
CDTM1_DTM1_CH1_DTV	0x000E4458	DTM5_CH1_DTV	0x00013158
CDTM1_DTM1_CH2_DTV	0x000E445C	DTM5_CH2_DTV	0x0001315C
CDTM1_DTM1_CH3_DTV	0x000E4460	DTM5_CH3_DTV	0x00013160
CDTM1_DTM1_CH_SR	0x000E4464	DTM5_CH_SR	0x00013164
CDTM1_DTM1_CH_CTRL3	0x000E4468	DTM5_CH_CTRL3	0x00013168
CDTM1_DTM4_CTRL	0x000E4500	DTM26_CTRL	0x00013680
CDTM1_DTM4_CH_CTRL1	0x000E4504	DTM26_CH_CTRL1	0x00013684
CDTM1_DTM4_CH_CTRL2	0x000E4508	DTM26_CH_CTRL2	0x00013688
CDTM1_DTM4_CH_CTRL2_SR	0x000E450C	DTM26_CH_CTRL2_SR	0x0001368C
CDTM1_DTM4_PS_CTRL	0x000E4510	DTM26_PS_CTRL	0x00013690
CDTM1_DTM4_CH0_DTV	0x000E4514	DTM26_CH0_DTV	0x00013694
CDTM1_DTM4_CH1_DTV	0x000E4518	DTM26_CH1_DTV	0x00013698
CDTM1_DTM4_CH2_DTV	0x000E451C	DTM26_CH2_DTV	0x0001369C
CDTM1_DTM4_CH3_DTV	0x000E4520	DTM26_CH3_DTV	0x000136A0
CDTM1_DTM4_CH_SR	0x000E4524	DTM26_CH_SR	0x000136A4
CDTM1_DTM4_CH_CTRL3	0x000E4528	DTM26_CH_CTRL3	0x000136A8
CDTM1_DTM5_CTRL	0x000E4540	DTM27_CTRL	0x000136C0
CDTM1_DTM5_CH_CTRL1	0x000E4544	DTM27_CH_CTRL1	0x000136C4

Table 184 GTM Legacy Address table (cont'd)

Register Label	Register Address	Legacy Register Label	Legacy Address
CDTM1_DTM5_CH_CTRL2	0x000E4548	DTM27_CH_CTRL2	0x000136C8
CDTM1_DTM5_CH_CTRL2_SR	0x000E454C	DTM27_CH_CTRL2_SR	0x000136CC
CDTM1_DTM5_PS_CTRL	0x000E4550	DTM27_PS_CTRL	0x000136D0
CDTM1_DTM5_CH0_DTV	0x000E4554	DTM27_CH0_DTV	0x000136D4
CDTM1_DTM5_CH1_DTV	0x000E4558	DTM27_CH1_DTV	0x000136D8
CDTM1_DTM5_CH2_DTV	0x000E455C	DTM27_CH2_DTV	0x000136DC
CDTM1_DTM5_CH3_DTV	0x000E4560	DTM27_CH3_DTV	0x000136E0
CDTM1_DTM5_CH_SR	0x000E4564	DTM27_CH_SR	0x000136E4
CDTM1_DTM5_CH_CTRL3	0x000E4568	DTM27_CH_CTRL3	0x000136E8
CDTM2_DTM0_CTRL	0x000E4800	DTM8_CTRL	0x00013200
CDTM2_DTM0_CH_CTRL1	0x000E4804	DTM8_CH_CTRL1	0x00013204
CDTM2_DTM0_CH_CTRL2	0x000E4808	DTM8_CH_CTRL2	0x00013208
CDTM2_DTM0_CH_CTRL2_SR	0x000E480C	DTM8_CH_CTRL2_SR	0x0001320C
CDTM2_DTM0_PS_CTRL	0x000E4810	DTM8_PS_CTRL	0x00013210
CDTM2_DTM0_CH0_DTV	0x000E4814	DTM8_CH0_DTV	0x00013214
CDTM2_DTM0_CH1_DTV	0x000E4818	DTM8_CH1_DTV	0x00013218
CDTM2_DTM0_CH2_DTV	0x000E481C	DTM8_CH2_DTV	0x0001321C
CDTM2_DTM0_CH3_DTV	0x000E4820	DTM8_CH3_DTV	0x00013220
CDTM2_DTM0_CH_SR	0x000E4824	DTM8_CH_SR	0x00013224
CDTM2_DTM0_CH_CTRL3	0x000E4828	DTM8_CH_CTRL3	0x00013228
CDTM2_DTM1_CTRL	0x000E4840	DTM9_CTRL	0x00013240
CDTM2_DTM1_CH_CTRL1	0x000E4844	DTM9_CH_CTRL1	0x00013244
CDTM2_DTM1_CH_CTRL2	0x000E4848	DTM9_CH_CTRL2	0x00013248
CDTM2_DTM1_CH_CTRL2_SR	0x000E484C	DTM9_CH_CTRL2_SR	0x0001324C
CDTM2_DTM1_PS_CTRL	0x000E4850	DTM9_PS_CTRL	0x00013250
CDTM2_DTM1_CH0_DTV	0x000E4854	DTM9_CH0_DTV	0x00013254
CDTM2_DTM1_CH1_DTV	0x000E4858	DTM9_CH1_DTV	0x00013258
CDTM2_DTM1_CH2_DTV	0x000E485C	DTM9_CH2_DTV	0x0001325C
CDTM2_DTM1_CH3_DTV	0x000E4860	DTM9_CH3_DTV	0x00013260
CDTM2_DTM1_CH_SR	0x000E4864	DTM9_CH_SR	0x00013264
CDTM2_DTM1_CH_CTRL3	0x000E4868	DTM9_CH_CTRL3	0x00013268
CDTM2_DTM4_CTRL	0x000E4900	DTM28_CTRL	0x00013700
CDTM2_DTM4_CH_CTRL1	0x000E4904	DTM28_CH_CTRL1	0x00013704
CDTM2_DTM4_CH_CTRL2	0x000E4908	DTM28_CH_CTRL2	0x00013708
CDTM2_DTM4_CH_CTRL2_SR	0x000E490C	DTM28_CH_CTRL2_SR	0x0001370C
CDTM2_DTM4_PS_CTRL	0x000E4910	DTM28_PS_CTRL	0x00013710
CDTM2_DTM4_CH0_DTV	0x000E4914	DTM28_CH0_DTV	0x00013714

Table 184 GTM Legacy Address table (cont'd)

Register Label	Register Address	Legacy Register Label	Legacy Address
CDTM2_DTM4_CH1_DTV	0x000E4918	DTM28_CH1_DTV	0x00013718
CDTM2_DTM4_CH2_DTV	0x000E491C	DTM28_CH2_DTV	0x0001371C
CDTM2_DTM4_CH3_DTV	0x000E4920	DTM28_CH3_DTV	0x00013720
CDTM2_DTM4_CH_SR	0x000E4924	DTM28_CH_SR	0x00013724
CDTM2_DTM4_CH_CTRL3	0x000E4928	DTM28_CH_CTRL3	0x00013728
CDTM2_DTM5_CTRL	0x000E4940	DTM29_CTRL	0x00013740
CDTM2_DTM5_CH_CTRL1	0x000E4944	DTM29_CH_CTRL1	0x00013744
CDTM2_DTM5_CH_CTRL2	0x000E4948	DTM29_CH_CTRL2	0x00013748
CDTM2_DTM5_CH_CTRL2_SR	0x000E494C	DTM29_CH_CTRL2_SR	0x0001374C
CDTM2_DTM5_PS_CTRL	0x000E4950	DTM29_PS_CTRL	0x00013750
CDTM2_DTM5_CH0_DTV	0x000E4954	DTM29_CH0_DTV	0x00013754
CDTM2_DTM5_CH1_DTV	0x000E4958	DTM29_CH1_DTV	0x00013758
CDTM2_DTM5_CH2_DTV	0x000E495C	DTM29_CH2_DTV	0x0001375C
CDTM2_DTM5_CH3_DTV	0x000E4960	DTM29_CH3_DTV	0x00013760
CDTM2_DTM5_CH_SR	0x000E4964	DTM29_CH_SR	0x00013764
CDTM2_DTM5_CH_CTRL3	0x000E4968	DTM29_CH_CTRL3	0x00013768
CDTM3_DTM4_CTRL	0x000E4D00	DTM30_CTRL	0x00013780
CDTM3_DTM4_CH_CTRL1	0x000E4D04	DTM30_CH_CTRL1	0x00013784
CDTM3_DTM4_CH_CTRL2	0x000E4D08	DTM30_CH_CTRL2	0x00013788
CDTM3_DTM4_CH_CTRL2_SR	0x000E4D0C	DTM30_CH_CTRL2_SR	0x0001378C
CDTM3_DTM4_PS_CTRL	0x000E4D10	DTM30_PS_CTRL	0x00013790
CDTM3_DTM4_CH0_DTV	0x000E4D14	DTM30_CH0_DTV	0x00013794
CDTM3_DTM4_CH1_DTV	0x000E4D18	DTM30_CH1_DTV	0x00013798
CDTM3_DTM4_CH2_DTV	0x000E4D1C	DTM30_CH2_DTV	0x0001379C
CDTM3_DTM4_CH3_DTV	0x000E4D20	DTM30_CH3_DTV	0x000137A0
CDTM3_DTM4_CH_SR	0x000E4D24	DTM30_CH_SR	0x000137A4
CDTM3_DTM4_CH_CTRL3	0x000E4D28	DTM30_CH_CTRL3	0x000137A8
CDTM3_DTM5_CTRL	0x000E4D40	DTM31_CTRL	0x000137C0
CDTM3_DTM5_CH_CTRL1	0x000E4D44	DTM31_CH_CTRL1	0x000137C4
CDTM3_DTM5_CH_CTRL2	0x000E4D48	DTM31_CH_CTRL2	0x000137C8
CDTM3_DTM5_CH_CTRL2_SR	0x000E4D4C	DTM31_CH_CTRL2_SR	0x000137CC
CDTM3_DTM5_PS_CTRL	0x000E4D50	DTM31_PS_CTRL	0x000137D0
CDTM3_DTM5_CH0_DTV	0x000E4D54	DTM31_CH0_DTV	0x000137D4
CDTM3_DTM5_CH1_DTV	0x000E4D58	DTM31_CH1_DTV	0x000137D8
CDTM3_DTM5_CH2_DTV	0x000E4D5C	DTM31_CH2_DTV	0x000137DC
CDTM3_DTM5_CH3_DTV	0x000E4D60	DTM31_CH3_DTV	0x000137E0
CDTM3_DTM5_CH_SR	0x000E4D64	DTM31_CH_SR	0x000137E4

Table 184 GTM Legacy Address table (cont'd)

Register Label	Register Address	Legacy Register Label	Legacy Address
CDTM3_DTM5_CH_CTRL3	0x000E4D68	DTM31_CH_CTRL3	0x000137E8
CDTM4_DTM4_CTRL	0x000E5100	DTM32_CTRL	0x00013800
CDTM4_DTM4_CH_CTRL1	0x000E5104	DTM32_CH_CTRL1	0x00013804
CDTM4_DTM4_CH_CTRL2	0x000E5108	DTM32_CH_CTRL2	0x00013808
CDTM4_DTM4_CH_CTRL2_SR	0x000E510C	DTM32_CH_CTRL2_SR	0x0001380C
CDTM4_DTM4_PS_CTRL	0x000E5110	DTM32_PS_CTRL	0x00013810
CDTM4_DTM4_CH0_DTV	0x000E5114	DTM32_CH0_DTV	0x00013814
CDTM4_DTM4_CH1_DTV	0x000E5118	DTM32_CH1_DTV	0x00013818
CDTM4_DTM4_CH2_DTV	0x000E511C	DTM32_CH2_DTV	0x0001381C
CDTM4_DTM4_CH3_DTV	0x000E5120	DTM32_CH3_DTV	0x00013820
CDTM4_DTM4_CH_SR	0x000E5124	DTM32_CH_SR	0x00013824
CDTM4_DTM4_CH_CTRL3	0x000E5128	DTM32_CH_CTRL3	0x00013828
CDTM4_DTM5_CTRL	0x000E5140	DTM33_CTRL	0x00013840
CDTM4_DTM5_CH_CTRL1	0x000E5144	DTM33_CH_CTRL1	0x00013844
CDTM4_DTM5_CH_CTRL2	0x000E5148	DTM33_CH_CTRL2	0x00013848
CDTM4_DTM5_CH_CTRL2_SR	0x000E514C	DTM33_CH_CTRL2_SR	0x0001384C
CDTM4_DTM5_PS_CTRL	0x000E5150	DTM33_PS_CTRL	0x00013850
CDTM4_DTM5_CH0_DTV	0x000E5154	DTM33_CH0_DTV	0x00013854
CDTM4_DTM5_CH1_DTV	0x000E5158	DTM33_CH1_DTV	0x00013858
CDTM4_DTM5_CH2_DTV	0x000E515C	DTM33_CH2_DTV	0x0001385C
CDTM4_DTM5_CH3_DTV	0x000E5160	DTM33_CH3_DTV	0x00013860
CDTM4_DTM5_CH_SR	0x000E5164	DTM33_CH_SR	0x00013864
CDTM4_DTM5_CH_CTRL3	0x000E5168	DTM33_CH_CTRL3	0x00013868

28.25.21 ARU Write Address Overview

The ARU write address map is specified in the following table.

Table 185 ARU Write Addresses

GTM Data Source	ARU Address
ARU_ACCESS	0x000
TIM0_WRADDR[0..7]	0x001..0x008
TIM1_WRADDR[0..7]	0x009..0x010
TIM2_WRADDR[0..7]	0x011..0x018
TIM3_WRADDR[0..7]	0x019..0x020
TIM4_WRADDR[0..7]	0x021..0x028
TIM5_WRADDR[0..7]	0x029..0x030
TIM6_WRADDR[0..7]	0x031..0x038
unused	0x039..0x050

Table 185 ARU Write Addresses (cont'd)

GTM Data Source	ARU Address
F2A0_WRADDR[0..7]	0x051..0x058
F2A1_WRADDR[0..7]	0x059..0x060
BRC_WRADDR[0..21]	0x061..0x076
MCS0_WRADDR[0..23]	0x077..0x08E
MCS1_WRADDR[0..23]	0x08F..0x0A6
MCS2_WRADDR[0..23]	0x0A7..0x0BE
MCS3_WRADDR[0..23]	0x0BF..0x0D6
MCS4_WRADDR[0..23]	0x0D7..0x0EE
MCS5_WRADDR[0..23]	0x0EF..0x106
MCS6_WRADDR[0..23]	0x107..0x11E
ATOM0_WRADDR[0..7]	0x11F..0x126
ATOM1_WRADDR[0..7]	0x127..0x12E
ATOM2_WRADDR[0..7]	0x12F..0x136
ATOM3_WRADDR[0..7]	0x137..0x13E
ATOM4_WRADDR[0..7]	0x13F..0x146
ATOM5_WRADDR[0..7]	0x147..0x14E
ATOM6_WRADDR[0..7]	0x14F..0x156
ATOM7_WRADDR[0..7]	0x157..0x15E
ATOM8_WRADDR[0..7]	0x15F..0x166
ATOM9_WRADDR[0..7]	0x167..0x16E
ATOM10_WRADDR[0..7]	0x16F..0x176
ATOM11_WRADDR[0..7]	0x177..0x17E
DPLL_WRADDR[0..31]	0x17F..0x19E
TIM7_WRADDR[0..7]	0x19F..0x1A6
F2A2_WRADDR[0..7]	0x1A7..0x1AE
MCS7_WRADDR[0..23]	0x1AF..0x1C6
MCS8_WRADDR[0..23]	0x1C7..0x1DE
MCS9_WRADDR[0..23]	0x1DF..0x1F6
unused	0x1F7..0x1FD
ARU_EMPTY_ADDR	0x1FE
ARU_FULL_ADDR	0x1FF

28.25.22 ARU Port Partitioning

All GTM sub-modules which are reading from ARU can be connected to one of two ARU read ports. Therefore, it can be read from two different ARU addresses in parallel.

Table 186 GTM ARU Partitioning

Modules	ARU-0 port	ARU-1 port
ATOM0	X	
ATOM1		X
ATOM2	X	
ATOM3		X
ATOM4	X	
ATOM5		X
ATOM6	X	
ATOM7		X
ATOM8	X	
ATOM9	X	
ATOM10	X	
ATOM11	X	
MCS0	X	
MCS1		X
MCS2	X	
MCS3		X
MCS4	X	
MCS5		X
MCS6	X	
MCS7		X
MCS8	X	
MCS9		X
DPLL		X
BRC		X
PSM0	X	
PSM1		X
PSM2	X	

28.25.23 ARU Read ID

Each ARU connected data destination is defined by a combination of ARU port (ARU0 or ARU1) and an ARU read ID. The two ARU counter are addressing two ARU read IDs in parallel. Depending on the ARU mode, both counter may have different values at different point in time (i.e. in dynamic routing mode). The maximum ARU round-trip time is determined by the value of the last ARU read ID. The following table describes the detailed addressing of GTM sub-modules by ARU read IDs.

The following table shows the ARU read IDs for the superset. It is identical to the table shown in the appendix for the TC39x silicon. The unused IDs are marked with “-”.

Table 187 GTM Read IDs

ARU read ID (dec)	ARU0	ARU1		GTM read ID (dec)	ARU0	ARU1
0	reserved	reserved		64	ATOM4 channel 3	DPLL action 31
1	**ARU0**	**ARU1**		65	MCS4 channel 7	MCS3 channel 7
2	**BRC** channel 0	**DPLL** action0		66	ATOM4 channel 4	**ATOM3** channel 0
3	**PSM0** channel 0	**PSM1** channel 0		67	**MCS6** channel 0	**MCS5** channel 0
4	BRC channel 1	DPLL action 1		68	ATOM4 channel 5	ATOM3 channel 1
5	PSM0 channel 1	PSM1 channel 1		69	MCS6 channel 1	MCS5 channel 1
6	BRC channel 2	DPLL action 2		70	ATOM4 channel 6	ATOM3 channel 2
7	PSM0 channel 2	PSM1 channel 2		71	MCS6 channel 2	MCS5 channel 2
8	BRC channel 3	DPLL action 3		72	ATOM4 channel 7	ATOM3 channel 3
9	PSM0 channel 3	PSM1 channel 3		73	MCS6 channel 3	MCS5 channel 3
10	BRC channel 4	DPLL action 4		74	**ATOM6** channel 0	ATOM3 channel 4
11	PSM0 channel 4	PSM1 channel 4		75	MCS6 channel 4	MCS5 channel 4
12	BRC channel 5	DPLL action 5		76	ATOM6 channel 1	ATOM3 channel 5
13	PSM0 channel 5	PSM1 channel 5		77	MCS6 channel 5	MCS5 channel 5
14	BRC channel 6	DPLL action 6		78	ATOM6 channel 2	ATOM3 channel 6
15	PSM0 channel 6	PSM1 channel 6		79	MCS6 channel 6	MCS5 channel 6
16	BRC channel 7	DPLL action 7		80	ATOM6 channel 3	ATOM3 channel 7
17	PSM0 channel 7	PSM1 channel 7		81	MCS6 channel 7	MCS5 channel 7
18	BRC channel 8	DPLL action 8		82	ATOM6 channel 4	**ATOM5** channel 0
19	**MCS0** channel 0	**MCS1** channel 0		83	**ATOM8** channel 0	**ATOM7** channel 0
20	BRC channel 9	DPLL action 9		84	ATOM6 channel 5	ATOM5 channel 1
21	MCS0 channel 1	MCS1 channel 1		85	ATOM8 channel 1	ATOM7 channel 1
22	BRC channel 10	DPLL action 10		86	ATOM6 channel 6	ATOM5 channel 2
23	MCS0 channel 2	MCS1 channel 2		87	ATOM8 channel 2	ATOM7 channel 2
24	BRC channel 11	DPLL action 11		88	ATOM6 channel 7	ATOM5 channel 3
25	MCS0 channel 3	MCS1 channel 3		89	ATOM8 channel 3	ATOM7 channel 3
26	**ATOM0** channel 0	DPLL action 12		90	**MCS8** channel 0	ATOM5 channel 4
27	MCS0 channel 4	MCS1 channel 4		91	ATOM8 channel 4	ATOM7 channel 4
28	ATOM0 channel 1	DPLL action 13		92	MCS8 channel 1	ATOM5 channel 5
29	MCS0 channel 5	MCS1 channel 5		93	ATOM8 channel 5	ATOM7 channel 5
30	ATOM0 channel 2	DPLL action 14		94	MCS8 channel 2	ATOM5 channel 6
31	MCS0 channel 6	MCS1 channel 6		95	ATOM8 channel 6	ATOM7 channel 6
32	ATOM0 channel 3	DPLL action 15		96	MCS8 channel 3	ATOM5 channel 7
33	MCS0 channel 7	MCS1 channel 7		97	ATOM8 channel 7	ATOM7 channel 7
34	ATOM0 channel 4	DPLL action 16		98	MCS8 channel 4	**MCS7** channel 0
35	MCS2 channel 0	**ATOM1** channel 0		99	**ATOM10** channel 0	**ATOM9** channel 0

Table 187 GTM Read IDs (cont'd)

ARU read ID (dec)	ARU0	ARU1		GTM read ID (dec)	ARU0	ARU1
36	ATOM0 channel 5	DPLL action 17		100	MCS8 channel 5	MCS7 channel 1
37	MCS2 channel 1	ATOM1 channel 1		101	ATOM10 channel 1	ATOM9 channel 1
38	ATOM0 channel 6	DPLL action 18		102	MCS8 channel 6	MCS7 channel 2
39	MCS2 channel 2	ATOM1 channel 2		103	ATOM10 channel 2	ATOM9 channel 2
40	ATOM0 channel 7	DPLL action 19		104	MCS8 channel 7	MCS7 channel 3
41	MCS2 channel 3	ATOM1 channel 3		105	ATOM10 channel 3	ATOM9 channel 3
42	**ATOM2** channel 0	DPLL action 20		106	**PSM2** channel 0	MCS7 channel 4
43	MCS2 channel 4	ATOM1 channel 4		107	ATOM10 channel 4	ATOM9 channel 4
44	ATOM2 channel 1	DPLL action 21		108	PSM2 channel 1	MCS7 channel 5
45	MCS2 channel 5	ATOM1 channel 5		109	ATOM10 channel 5	ATOM9 channel 5
46	ATOM2 channel 2	DPLL action 22		110	PSM2 channel 2	MCS7 channel 6
47	MCS2 channel 6	ATOM1 channel 6		111	ATOM10 channel 6	ATOM9 channel 6
48	ATOM2 channel 3	DPLL action 23		112	PSM2 channel 3	MCS7 channel 7
49	MCS2 channel 7	ATOM1 channel 7		113	ATOM10 channel 7	ATOM9 channel 6
50	ATOM2 channel 4	DPLL action 24		114	PSM2 channel 4	**MCS9** channel 0
51	**MCS4** channel 0	**MCS3** channel 0		115	**ATOM11** channel 0	MCS9 channel 1
52	ATOM2 channel 5	DPLL action 25		116	PSM2 channel 5	MCS9 channel 2
53	MCS4 channel 1	MCS3 channel 1		117	ATOM11 channel 1	-
54	ATOM2 channel 6	DPLL action 26		118	PSM2 channel 6	MCS9 channel 3
55	MCS4 channel 2	MCS3 channel 2		119	ATOM11 channel 2	-
56	ATOM2 channel 7	DPLL action 27		120	PSM2 channel 7	MCS9 channel 4
57	MCS4 channel 3	MCS3 channel 3		121	ATOM11 channel 3	-
58	**ATOM4** channel 0	DPLL action 28		122	ATOM11 channel 4	MCS9 channel 5
59	MCS4 channel 4	MCS3 channel 4		123	ATOM11 channel 5	-
60	ATOM4 channel 1	DPLL action 29		124	-	MCS9 channel 6
61	MCS4 channel 5	MCS3 channel 5		125	ATOM11 channel 6	-
62	ATOM4 channel 2	DPLL action 30		126	-	MCS9 channel 7
63	MCS4 channel 6	MCS3 channel 6		127	ATOM11 channel 7	-

28.25.24 MCS Master Interface Address Map

Inside a specific GTM cluster, the MCS can access all the other GTM module registers. The address map for all the GTM registers is the same for each cluster. If a module is not implemented in a cluster, these addresses are reserved.

Table 188 MCS Master Interface Address Map

Register Label	Register Address
TIM_CH0_GPR0	0x00000000
TIM_CH0_GPR1	0x00000004
TIM_CH0_CNT	0x00000008
TIM_CH0_ECNT	0x0000000C
TIM_CH0_CNTS	0x00000010
TIM_CH0_TDUC	0x00000014
TIM_CH0_TDUV	0x00000018
TIM_CH0_FLT_RE	0x0000001C
TIM_CH0_FLT_FE	0x00000020
TIM_CH0_CTRL	0x00000024
TIM_CH0_ECTRL	0x00000028
TIM_CH0_IRQ_NOTIFY	0x0000002C
TIM_CH0_IRQ_EN	0x00000030
TIM_CH0_IRQ_FORCINT	0x00000034
TIM_CH0_IRQ_MODE	0x00000038
TIM_CH0_EIRQ_EN	0x0000003C
TIM_INP_VAL	0x00000074
TIM_IN_SRC	0x00000078
TIM_RST	0x0000007C
TIM_CH1_GPR0	0x00000080
TIM_CH1_GPR1	0x00000084
TIM_CH1_CNT	0x00000088
TIM_CH1_ECNT	0x0000008C
TIM_CH1_CNTS	0x00000090
TIM_CH1_TDUC	0x00000094
TIM_CH1_TDUV	0x00000098
TIM_CH1_FLT_RE	0x0000009C
TIM_CH1_FLT_FE	0x000000A0
TIM_CH1_CTRL	0x000000A4
TIM_CH1_ECTRL	0x000000A8
TIM_CH1_IRQ_NOTIFY	0x000000AC
TIM_CH1_IRQ_EN	0x000000B0
TIM_CH1_IRQ_FORCINT	0x000000B4
TIM_CH1_IRQ_MODE	0x000000B8
TIM_CH1_EIRQ_EN	0x000000BC
TIM_CH2_GPR0	0x00000100
TIM_CH2_GPR1	0x00000104

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
TIM_CH2_CNT	0x00000108
TIM_CH2_ECNT	0x0000010C
TIM_CH2_CNTS	0x00000110
TIM_CH2_TDUC	0x00000114
TIM_CH2_TDUV	0x00000118
TIM_CH2_FLT_RE	0x0000011C
TIM_CH2_FLT_FE	0x00000120
TIM_CH2_CTRL	0x00000124
TIM_CH2_ECTRL	0x00000128
TIM_CH2_IRQ_NOTIFY	0x0000012C
TIM_CH2_IRQ_EN	0x00000130
TIM_CH2_IRQ_FORCINT	0x00000134
TIM_CH2_IRQ_MODE	0x00000138
TIM_CH2_EIRQ_EN	0x0000013C
TIM_CH3_GPR0	0x00000180
TIM_CH3_GPR1	0x00000184
TIM_CH3_CNT	0x00000188
TIM_CH3_ECNT	0x0000018C
TIM_CH3_CNTS	0x00000190
TIM_CH3_TDUC	0x00000194
TIM_CH3_TDUV	0x00000198
TIM_CH3_FLT_RE	0x0000019C
TIM_CH3_FLT_FE	0x000001A0
TIM_CH3_CTRL	0x000001A4
TIM_CH3_ECTRL	0x000001A8
TIM_CH3_IRQ_NOTIFY	0x000001AC
TIM_CH3_IRQ_EN	0x000001B0
TIM_CH3_IRQ_FORCINT	0x000001B4
TIM_CH3_IRQ_MODE	0x000001B8
TIM_CH3_EIRQ_EN	0x000001BC
TIM_CH4_GPR0	0x00000200
TIM_CH4_GPR1	0x00000204
TIM_CH4_CNT	0x00000208
TIM_CH4_ECNT	0x0000020C
TIM_CH4_CNTS	0x00000210
TIM_CH4_TDUC	0x00000214
TIM_CH4_TDUV	0x00000218
TIM_CH4_FLT_RE	0x0000021C

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
TIM_CH4_FLT_FE	0x00000220
TIM_CH4_CTRL	0x00000224
TIM_CH4_ECTRL	0x00000228
TIM_CH4_IRQ_NOTIFY	0x0000022C
TIM_CH4_IRQ_EN	0x00000230
TIM_CH4_IRQ_FORCINT	0x00000234
TIM_CH4_IRQ_MODE	0x00000238
TIM_CH4_EIRQ_EN	0x0000023C
TIM_CH5_GPR0	0x00000280
TIM_CH5_GPR1	0x00000284
TIM_CH5_CNT	0x00000288
TIM_CH5_ECNT	0x0000028C
TIM_CH5_CNTS	0x00000290
TIM_CH5_TDUC	0x00000294
TIM_CH5_TDUV	0x00000298
TIM_CH5_FLT_RE	0x0000029C
TIM_CH5_FLT_FE	0x000002A0
TIM_CH5_CTRL	0x000002A4
TIM_CH5_ECTRL	0x000002A8
TIM_CH5_IRQ_NOTIFY	0x000002AC
TIM_CH5_IRQ_EN	0x000002B0
TIM_CH5_IRQ_FORCINT	0x000002B4
TIM_CH5_IRQ_MODE	0x000002B8
TIM_CH5_EIRQ_EN	0x000002BC
TIM_CH6_GPR0	0x00000300
TIM_CH6_GPR1	0x00000304
TIM_CH6_CNT	0x00000308
TIM_CH6_ECNT	0x0000030C
TIM_CH6_CNTS	0x00000310
TIM_CH6_TDUC	0x00000314
TIM_CH6_TDUV	0x00000318
TIM_CH6_FLT_RE	0x0000031C
TIM_CH6_FLT_FE	0x00000320
TIM_CH6_CTRL	0x00000324
TIM_CH6_ECTRL	0x00000328
TIM_CH6_IRQ_NOTIFY	0x0000032C
TIM_CH6_IRQ_EN	0x00000330
TIM_CH6_IRQ_FORCINT	0x00000334

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
TIM_CH6_IRQ_MODE	0x00000338
TIM_CH6_EIRQ_EN	0x0000033C
TIM_CH7_GPR0	0x00000380
TIM_CH7_GPR1	0x00000384
TIM_CH7_CNT	0x00000388
TIM_CH7_ECNT	0x0000038C
TIM_CH7_CNTS	0x00000390
TIM_CH7_TDUC	0x00000394
TIM_CH7_TDUV	0x00000398
TIM_CH7_FLT_RE	0x0000039C
TIM_CH7_FLT_FE	0x000003A0
TIM_CH7_CTRL	0x000003A4
TIM_CH7_ECTRL	0x000003A8
TIM_CH7_IRQ_NOTIFY	0x000003AC
TIM_CH7_IRQ_EN	0x000003B0
TIM_CH7_IRQ_FORCINT	0x000003B4
TIM_CH7_IRQ_MODE	0x000003B8
TIM_CH7_EIRQ_EN	0x000003BC
TOM_CH0_CTRL	0x00000C00
TOM_CH0_SR0	0x00000C04
TOM_CH0_SR1	0x00000C08
TOM_CH0_CM0	0x00000C0C
TOM_CH0_CM1	0x00000C10
TOM_CH0_CN0	0x00000C14
TOM_CH0_STAT	0x00000C18
TOM_CH0_IRQ_NOTIFY	0x00000C1C
TOM_CH0_IRQ_EN	0x00000C20
TOM_CH0_IRQ_FORCINT	0x00000C24
TOM_CH0_IRQ_MODE	0x00000C28
TOM_TGC0_GLB_CTRL	0x00000C30
TOM_TGC0_ACT_TB	0x00000C34
TOM_TGC0_FUPD_CTRL	0x00000C38
TOM_TGC0_INT_TRIG	0x00000C3C
TOM_CH1_CTRL	0x00000C40
TOM_CH1_SR0	0x00000C44
TOM_CH1_SR1	0x00000C48
TOM_CH1_CM0	0x00000C4C
TOM_CH1_CM1	0x00000C50

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
TOM_CH1_CN0	0x00000C54
TOM_CH1_STAT	0x00000C58
TOM_CH1_IRQ_NOTIFY	0x00000C5C
TOM_CH1_IRQ_EN	0x00000C60
TOM_CH1_IRQ_FORCINT	0x00000C64
TOM_CH1_IRQ_MODE	0x00000C68
TOM_TGC0_ENDIS_CTRL	0x00000C70
TOM_TGC0_ENDIS_STAT	0x00000C74
TOM_TGC0_OUTEN_CTRL	0x00000C78
TOM_TGC0_OUTEN_STAT	0x00000C7C
TOM_CH2_CTRL	0x00000C80
TOM_CH2_SR0	0x00000C84
TOM_CH2_SR1	0x00000C88
TOM_CH2_CM0	0x00000C8C
TOM_CH2_CM1	0x00000C90
TOM_CH2_CN0	0x00000C94
TOM_CH2_STAT	0x00000C98
TOM_CH2_IRQ_NOTIFY	0x00000C9C
TOM_CH2_IRQ_EN	0x00000CA0
TOM_CH2_IRQ_FORCINT	0x00000CA4
TOM_CH2_IRQ_MODE	0x00000CA8
TOM_CH3_CTRL	0x00000CC0
TOM_CH3_SR0	0x00000CC4
TOM_CH3_SR1	0x00000CC8
TOM_CH3_CM0	0x00000CCC
TOM_CH3_CM1	0x00000CD0
TOM_CH3_CN0	0x00000CD4
TOM_CH3_STAT	0x00000CD8
TOM_CH3_IRQ_NOTIFY	0x00000CDC
TOM_CH3_IRQ_EN	0x00000CE0
TOM_CH3_IRQ_FORCINT	0x00000CE4
TOM_CH3_IRQ_MODE	0x00000CE8
TOM_CH4_CTRL	0x00000D00
TOM_CH4_SR0	0x00000D04
TOM_CH4_SR1	0x00000D08
TOM_CH4_CM0	0x00000D0C
TOM_CH4_CM1	0x00000D10
TOM_CH4_CN0	0x00000D14

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
TOM_CH4_STAT	0x00000D18
TOM_CH4_IRQ_NOTIFY	0x00000D1C
TOM_CH4_IRQ_EN	0x00000D20
TOM_CH4_IRQ_FORCINT	0x00000D24
TOM_CH4_IRQ_MODE	0x00000D28
TOM_CH5_CTRL	0x00000D40
TOM_CH5_SR0	0x00000D44
TOM_CH5_SR1	0x00000D48
TOM_CH5_CM0	0x00000D4C
TOM_CH5_CM1	0x00000D50
TOM_CH5_CN0	0x00000D54
TOM_CH5_STAT	0x00000D58
TOM_CH5_IRQ_NOTIFY	0x00000D5C
TOM_CH5_IRQ_EN	0x00000D60
TOM_CH5_IRQ_FORCINT	0x00000D64
TOM_CH5_IRQ_MODE	0x00000D68
TOM_CH6_CTRL	0x00000D80
TOM_CH6_SR0	0x00000D84
TOM_CH6_SR1	0x00000D88
TOM_CH6_CM0	0x00000D8C
TOM_CH6_CM1	0x00000D90
TOM_CH6_CN0	0x00000D94
TOM_CH6_STAT	0x00000D98
TOM_CH6_IRQ_NOTIFY	0x00000D9C
TOM_CH6_IRQ_EN	0x00000DA0
TOM_CH6_IRQ_FORCINT	0x00000DA4
TOM_CH6_IRQ_MODE	0x00000DA8
TOM_CH7_CTRL	0x00000DC0
TOM_CH7_SR0	0x00000DC4
TOM_CH7_SR1	0x00000DC8
TOM_CH7_CM0	0x00000DCC
TOM_CH7_CM1	0x00000DD0
TOM_CH7_CN0	0x00000DD4
TOM_CH7_STAT	0x00000DD8
TOM_CH7_IRQ_NOTIFY	0x00000DDC
TOM_CH7_IRQ_EN	0x00000DE0
TOM_CH7_IRQ_FORCINT	0x00000DE4
TOM_CH7_IRQ_MODE	0x00000DE8

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
TOM_CH8_CTRL	0x00000E00
TOM_CH8_SR0	0x00000E04
TOM_CH8_SR1	0x00000E08
TOM_CH8_CM0	0x00000E0C
TOM_CH8_CM1	0x00000E10
TOM_CH8_CN0	0x00000E14
TOM_CH8_STAT	0x00000E18
TOM_CH8_IRQ_NOTIFY	0x00000E1C
TOM_CH8_IRQ_EN	0x00000E20
TOM_CH8_IRQ_FORCINT	0x00000E24
TOM_CH8_IRQ_MODE	0x00000E28
TOM_TGC1_GLB_CTRL	0x00000E30
TOM_TGC1_ACT_TB	0x00000E34
TOM_TGC1_FUPD_CTRL	0x00000E38
TOM_TGC1_INT_TRIG	0x00000E3C
TOM_CH9_CTRL	0x00000E40
TOM_CH9_SR0	0x00000E44
TOM_CH9_SR1	0x00000E48
TOM_CH9_CM0	0x00000E4C
TOM_CH9_CM1	0x00000E50
TOM_CH9_CN0	0x00000E54
TOM_CH9_STAT	0x00000E58
TOM_CH9_IRQ_NOTIFY	0x00000E5C
TOM_CH9_IRQ_EN	0x00000E60
TOM_CH9_IRQ_FORCINT	0x00000E64
TOM_CH9_IRQ_MODE	0x00000E68
TOM_TGC1_ENDIS_CTRL	0x00000E70
TOM_TGC1_ENDIS_STAT	0x00000E74
TOM_TGC1_OUTEN_CTRL	0x00000E78
TOM_TGC1_OUTEN_STAT	0x00000E7C
TOM_CH10_CTRL	0x00000E80
TOM_CH10_SR0	0x00000E84
TOM_CH10_SR1	0x00000E88
TOM_CH10_CM0	0x00000E8C
TOM_CH10_CM1	0x00000E90
TOM_CH10_CN0	0x00000E94
TOM_CH10_STAT	0x00000E98
TOM_CH10_IRQ_NOTIFY	0x00000E9C

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
TOM_CH10_IRQ_EN	0x00000EA0
TOM_CH10_IRQ_FORCINT	0x00000EA4
TOM_CH10_IRQ_MODE	0x00000EA8
TOM_CH11_CTRL	0x00000EC0
TOM_CH11_SR0	0x00000EC4
TOM_CH11_SR1	0x00000EC8
TOM_CH11_CM0	0x00000ECC
TOM_CH11_CM1	0x00000ED0
TOM_CH11_CN0	0x00000ED4
TOM_CH11_STAT	0x00000ED8
TOM_CH11_IRQ_NOTIFY	0x00000EDC
TOM_CH11_IRQ_EN	0x00000EE0
TOM_CH11_IRQ_FORCINT	0x00000EE4
TOM_CH11_IRQ_MODE	0x00000EE8
TOM_CH12_CTRL	0x00000F00
TOM_CH12_SR0	0x00000F04
TOM_CH12_SR1	0x00000F08
TOM_CH12_CM0	0x00000F0C
TOM_CH12_CM1	0x00000F10
TOM_CH12_CN0	0x00000F14
TOM_CH12_STAT	0x00000F18
TOM_CH12_IRQ_NOTIFY	0x00000F1C
TOM_CH12_IRQ_EN	0x00000F20
TOM_CH12_IRQ_FORCINT	0x00000F24
TOM_CH12_IRQ_MODE	0x00000F28
TOM_CH13_CTRL	0x00000F40
TOM_CH13_SR0	0x00000F44
TOM_CH13_SR1	0x00000F48
TOM_CH13_CM0	0x00000F4C
TOM_CH13_CM1	0x00000F50
TOM_CH13_CN0	0x00000F54
TOM_CH13_STAT	0x00000F58
TOM_CH13_IRQ_NOTIFY	0x00000F5C
TOM_CH13_IRQ_EN	0x00000F60
TOM_CH13_IRQ_FORCINT	0x00000F64
TOM_CH13_IRQ_MODE	0x00000F68
TOM_CH14_CTRL	0x00000F80
TOM_CH14_SR0	0x00000F84

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
TOM_CH14_SR1	0x00000F88
TOM_CH14_CM0	0x00000F8C
TOM_CH14_CM1	0x00000F90
TOM_CH14_CN0	0x00000F94
TOM_CH14_STAT	0x00000F98
TOM_CH14_IRQ_NOTIFY	0x00000F9C
TOM_CH14_IRQ_EN	0x00000FA0
TOM_CH14_IRQ_FORCINT	0x00000FA4
TOM_CH14_IRQ_MODE	0x00000FA8
TOM_CH15_CTRL	0x00000FC0
TOM_CH15_SR0	0x00000FC4
TOM_CH15_SR1	0x00000FC8
TOM_CH15_CM0	0x00000FCC
TOM_CH15_CM1	0x00000FD0
TOM_CH15_CN0	0x00000FD4
TOM_CH15_STAT	0x00000FD8
TOM_CH15_IRQ_NOTIFY	0x00000FDC
TOM_CH15_IRQ_EN	0x00000FE0
TOM_CH15_IRQ_FORCINT	0x00000FE4
TOM_CH15_IRQ_MODE	0x00000FE8
ATOM_CH0_RDADDR	0x00001800
ATOM_CH0_CTRL	0x00001804
ATOM_CH0_SR0	0x00001808
ATOM_CH0_SR1	0x0000180C
ATOM_CH0_CM0	0x00001810
ATOM_CH0_CM1	0x00001814
ATOM_CH0_CN0	0x00001818
ATOM_CH0_STAT	0x0000181C
ATOM_CH0_IRQ_NOTIFY	0x00001820
ATOM_CH0_IRQ_EN	0x00001824
ATOM_CH0_IRQ_FORCINT	0x00001828
ATOM_CH0_IRQ_MODE	0x0000182C
ATOM_AGC_GLB_CTRL	0x00001840
ATOM_AGC_ENDIS_CTRL	0x00001844
ATOM_AGC_ENDIS_STAT	0x00001848
ATOM_AGC_ACT_TB	0x0000184C
ATOM_AGC_OUTEN_CTRL	0x00001850
ATOM_AGC_OUTEN_STAT	0x00001854

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
ATOM_AGC_FUPD_CTRL	0x00001858
ATOM_AGC_INT_TRIG	0x0000185C
ATOM_CH1_RDADDR	0x00001880
ATOM_CH1_CTRL	0x00001884
ATOM_CH1_SR0	0x00001888
ATOM_CH1_SR1	0x0000188C
ATOM_CH1_CM0	0x00001890
ATOM_CH1_CM1	0x00001894
ATOM_CH1_CN0	0x00001898
ATOM_CH1_STAT	0x0000189C
ATOM_CH1_IRQ_NOTIFY	0x000018A0
ATOM_CH1_IRQ_EN	0x000018A4
ATOM_CH1_IRQ_FORCINT	0x000018A8
ATOM_CH1_IRQ_MODE	0x000018AC
ATOM_CH2_RDADDR	0x00001900
ATOM_CH2_CTRL	0x00001904
ATOM_CH2_SR0	0x00001908
ATOM_CH2_SR1	0x0000190C
ATOM_CH2_CM0	0x00001910
ATOM_CH2_CM1	0x00001914
ATOM_CH2_CN0	0x00001918
ATOM_CH2_STAT	0x0000191C
ATOM_CH2_IRQ_NOTIFY	0x00001920
ATOM_CH2_IRQ_EN	0x00001924
ATOM_CH2_IRQ_FORCINT	0x00001928
ATOM_CH2_IRQ_MODE	0x0000192C
ATOM_CH3_RDADDR	0x00001980
ATOM_CH3_CTRL	0x00001984
ATOM_CH3_SR0	0x00001988
ATOM_CH3_SR1	0x0000198C
ATOM_CH3_CM0	0x00001990
ATOM_CH3_CM1	0x00001994
ATOM_CH3_CN0	0x00001998
ATOM_CH3_STAT	0x0000199C
ATOM_CH3_IRQ_NOTIFY	0x000019A0
ATOM_CH3_IRQ_EN	0x000019A4
ATOM_CH3_IRQ_FORCINT	0x000019A8
ATOM_CH3_IRQ_MODE	0x000019AC

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
ATOM_CH4_RDADDR	0x00001A00
ATOM_CH4_CTRL	0x00001A04
ATOM_CH4_SR0	0x00001A08
ATOM_CH4_SR1	0x00001A0C
ATOM_CH4_CM0	0x00001A10
ATOM_CH4_CM1	0x00001A14
ATOM_CH4_CN0	0x00001A18
ATOM_CH4_STAT	0x00001A1C
ATOM_CH4_IRQ_NOTIFY	0x00001A20
ATOM_CH4_IRQ_EN	0x00001A24
ATOM_CH4_IRQ_FORCINT	0x00001A28
ATOM_CH4_IRQ_MODE	0x00001A2C
ATOM_CH5_RDADDR	0x00001A80
ATOM_CH5_CTRL	0x00001A84
ATOM_CH5_SR0	0x00001A88
ATOM_CH5_SR1	0x00001A8C
ATOM_CH5_CM0	0x00001A90
ATOM_CH5_CM1	0x00001A94
ATOM_CH5_CN0	0x00001A98
ATOM_CH5_STAT	0x00001A9C
ATOM_CH5_IRQ_NOTIFY	0x00001AA0
ATOM_CH5_IRQ_EN	0x00001AA4
ATOM_CH5_IRQ_FORCINT	0x00001AA8
ATOM_CH5_IRQ_MODE	0x00001AAC
ATOM_CH6_RDADDR	0x00001B00
ATOM_CH6_CTRL	0x00001B04
ATOM_CH6_SR0	0x00001B08
ATOM_CH6_SR1	0x00001B0C
ATOM_CH6_CM0	0x00001B10
ATOM_CH6_CM1	0x00001B14
ATOM_CH6_CN0	0x00001B18
ATOM_CH6_STAT	0x00001B1C
ATOM_CH6_IRQ_NOTIFY	0x00001B20
ATOM_CH6_IRQ_EN	0x00001B24
ATOM_CH6_IRQ_FORCINT	0x00001B28
ATOM_CH6_IRQ_MODE	0x00001B2C
ATOM_CH7_RDADDR	0x00001B80
ATOM_CH7_CTRL	0x00001B84

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
ATOM_CH7_SR0	0x00001B88
ATOM_CH7_SR1	0x00001B8C
ATOM_CH7_CM0	0x00001B90
ATOM_CH7_CM1	0x00001B94
ATOM_CH7_CN0	0x00001B98
ATOM_CH7_STAT	0x00001B9C
ATOM_CH7_IRQ_NOTIFY	0x00001BA0
ATOM_CH7_IRQ_EN	0x00001BA4
ATOM_CH7_IRQ_FORCINT	0x00001BA8
ATOM_CH7_IRQ_MODE	0x00001BAC
CDTM_DTM0_CTRL	0x00003000
CDTM_DTM0_CH_CTRL1	0x00003004
CDTM_DTM0_CH_CTRL2	0x00003008
CDTM_DTM0_CH_CTRL2_SR	0x0000300C
CDTM_DTM0_PS_CTRL	0x00003010
CDTM_DTM0_CH0_DTV	0x00003014
CDTM_DTM0_CH1_DTV	0x00003018
CDTM_DTM0_CH2_DTV	0x0000301C
CDTM_DTM0_CH3_DTV	0x00003020
CDTM_DTM0_CH_SR	0x00003024
CDTM_DTM0_CH_CTRL3	0x00003028
CDTM_DTM1_CTRL	0x00003040
CDTM_DTM1_CH_CTRL1	0x00003044
CDTM_DTM1_CH_CTRL2	0x00003048
CDTM_DTM1_CH_CTRL2_SR	0x0000304C
CDTM_DTM1_PS_CTRL	0x00003050
CDTM_DTM1_CH0_DTV	0x00003054
CDTM_DTM1_CH1_DTV	0x00003058
CDTM_DTM1_CH2_DTV	0x0000305C
CDTM_DTM1_CH3_DTV	0x00003060
CDTM_DTM1_CH_SR	0x00003064
CDTM_DTM1_CH_CTRL3	0x00003068
CDTM_DTM2_CTRL	0x00003080
CDTM_DTM2_CH_CTRL1	0x00003084
CDTM_DTM2_CH_CTRL2	0x00003088
CDTM_DTM2_CH_CTRL2_SR	0x0000308C
CDTM_DTM2_PS_CTRL	0x00003090
CDTM_DTM2_CH0_DTV	0x00003094

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
CDTM_DTM2_CH1_DTV	0x00003098
CDTM_DTM2_CH2_DTV	0x0000309C
CDTM_DTM2_CH3_DTV	0x000030A0
CDTM_DTM2_CH_SR	0x000030A4
CDTM_DTM2_CH_CTRL3	0x000030A8
CDTM_DTM3_CTRL	0x000030C0
CDTM_DTM3_CH_CTRL1	0x000030C4
CDTM_DTM3_CH_CTRL2	0x000030C8
CDTM_DTM3_CH_CTRL2_SR	0x000030CC
CDTM_DTM3_PS_CTRL	0x000030D0
CDTM_DTM3_CH0_DTV	0x000030D4
CDTM_DTM3_CH1_DTV	0x000030D8
CDTM_DTM3_CH2_DTV	0x000030DC
CDTM_DTM3_CH3_DTV	0x000030E0
CDTM_DTM3_CH_SR	0x000030E4
CDTM_DTM3_CH_CTRL3	0x000030E8
CDTM_DTM4_CTRL	0x00003100
CDTM_DTM4_CH_CTRL1	0x00003104
CDTM_DTM4_CH_CTRL2	0x00003108
CDTM_DTM4_CH_CTRL2_SR	0x0000310C
CDTM_DTM4_PS_CTRL	0x00003110
CDTM_DTM4_CH0_DTV	0x00003114
CDTM_DTM4_CH1_DTV	0x00003118
CDTM_DTM4_CH2_DTV	0x0000311C
CDTM_DTM4_CH3_DTV	0x00003120
CDTM_DTM4_CH_SR	0x00003124
CDTM_DTM4_CH_CTRL3	0x00003128
CDTM_DTM5_CTRL	0x00003140
CDTM_DTM5_CH_CTRL1	0x00003144
CDTM_DTM5_CH_CTRL2	0x00003148
CDTM_DTM5_CH_CTRL2_SR	0x0000314C
CDTM_DTM5_PS_CTRL	0x00003150
CDTM_DTM5_CH0_DTV	0x00003154
CDTM_DTM5_CH1_DTV	0x00003158
CDTM_DTM5_CH2_DTV	0x0000315C
CDTM_DTM5_CH3_DTV	0x00003160
CDTM_DTM5_CH_SR	0x00003164
CDTM_DTM5_CH_CTRL3	0x00003168

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
MCS_CH0_R0	0x00003800
MCS_CH0_R1	0x00003804
MCS_CH0_R2	0x00003808
MCS_CH0_R3	0x0000380C
MCS_CH0_R4	0x00003810
MCS_CH0_R5	0x00003814
MCS_CH0_R6	0x00003818
MCS_CH0_R7	0x0000381C
MCS_CH0_CTRL	0x00003820
MCS_CH0_ACB	0x00003824
MCS_CTRG	0x00003828
MCS_STRG	0x0000382C
MCS_CH0_MHB	0x0000383C
MCS_CH0_PC	0x00003840
MCS_CH0_IRQ_NOTIFY	0x00003844
MCS_CH0_IRQ_EN	0x00003848
MCS_CH0_IRQ_FORCINT	0x0000384C
MCS_CH0_IRQ_MODE	0x00003850
MCS_CH0_EIRQ_EN	0x00003854
MCS_REG_PROT	0x00003860
MCS_CTRL_STAT	0x00003864
MCS_RESET	0x00003868
MCS_CAT	0x0000386C
MCS_CWT	0x00003870
MCS_ERR	0x0000387C
MCS_CH1_R0	0x00003880
MCS_CH1_R1	0x00003884
MCS_CH1_R2	0x00003888
MCS_CH1_R3	0x0000388C
MCS_CH1_R4	0x00003890
MCS_CH1_R5	0x00003894
MCS_CH1_R6	0x00003898
MCS_CH1_R7	0x0000389C
MCS_CH1_CTRL	0x000038A0
MCS_CH1_ACB	0x000038A4
MCS_CH1_MHB	0x000038BC
MCS_CH1_PC	0x000038C0
MCS_CH1_IRQ_NOTIFY	0x000038C4

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
MCS_CH1_IRQ_EN	0x000038C8
MCS_CH1_IRQ_FORCINT	0x000038CC
MCS_CH1_IRQ_MODE	0x000038D0
MCS_CH1_EIRQ_EN	0x000038D4
MCS_CH2_R0	0x00003900
MCS_CH2_R1	0x00003904
MCS_CH2_R2	0x00003908
MCS_CH2_R3	0x0000390C
MCS_CH2_R4	0x00003910
MCS_CH2_R5	0x00003914
MCS_CH2_R6	0x00003918
MCS_CH2_R7	0x0000391C
MCS_CH2_CTRL	0x00003920
MCS_CH2_ACB	0x00003924
MCS_CH2_MHB	0x0000393C
MCS_CH2_PC	0x00003940
MCS_CH2_IRQ_NOTIFY	0x00003944
MCS_CH2_IRQ_EN	0x00003948
MCS_CH2_IRQ_FORCINT	0x0000394C
MCS_CH2_IRQ_MODE	0x00003950
MCS_CH2_EIRQ_EN	0x00003954
MCS_CH3_R0	0x00003980
MCS_CH3_R1	0x00003984
MCS_CH3_R2	0x00003988
MCS_CH3_R3	0x0000398C
MCS_CH3_R4	0x00003990
MCS_CH3_R5	0x00003994
MCS_CH3_R6	0x00003998
MCS_CH3_R7	0x0000399C
MCS_CH3_CTRL	0x000039A0
MCS_CH3_ACB	0x000039A4
MCS_CH3_MHB	0x000039BC
MCS_CH3_PC	0x000039C0
MCS_CH3_IRQ_NOTIFY	0x000039C4
MCS_CH3_IRQ_EN	0x000039C8
MCS_CH3_IRQ_FORCINT	0x000039CC
MCS_CH3_IRQ_MODE	0x000039D0
MCS_CH3_EIRQ_EN	0x000039D4

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
MCS_CH4_R0	0x00003A00
MCS_CH4_R1	0x00003A04
MCS_CH4_R2	0x00003A08
MCS_CH4_R3	0x00003A0C
MCS_CH4_R4	0x00003A10
MCS_CH4_R5	0x00003A14
MCS_CH4_R6	0x00003A18
MCS_CH4_R7	0x00003A1C
MCS_CH4_CTRL	0x00003A20
MCS_CH4_ACB	0x00003A24
MCS_CH4_MHB	0x00003A3C
MCS_CH4_PC	0x00003A40
MCS_CH4_IRQ_NOTIFY	0x00003A44
MCS_CH4_IRQ_EN	0x00003A48
MCS_CH4_IRQ_FORCINT	0x00003A4C
MCS_CH4_IRQ_MODE	0x00003A50
MCS_CH4_EIRQ_EN	0x00003A54
MCS_CH5_R0	0x00003A80
MCS_CH5_R1	0x00003A84
MCS_CH5_R2	0x00003A88
MCS_CH5_R3	0x00003A8C
MCS_CH5_R4	0x00003A90
MCS_CH5_R5	0x00003A94
MCS_CH5_R6	0x00003A98
MCS_CH5_R7	0x00003A9C
MCS_CH5_CTRL	0x00003AA0
MCS_CH5_ACB	0x00003AA4
MCS_CH5_MHB	0x00003ABC
MCS_CH5_PC	0x00003AC0
MCS_CH5_IRQ_NOTIFY	0x00003AC4
MCS_CH5_IRQ_EN	0x00003AC8
MCS_CH5_IRQ_FORCINT	0x00003ACC
MCS_CH5_IRQ_MODE	0x00003AD0
MCS_CH5_EIRQ_EN	0x00003AD4
MCS_CH6_R0	0x00003B00
MCS_CH6_R1	0x00003B04
MCS_CH6_R2	0x00003B08
MCS_CH6_R3	0x00003B0C

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
MCS_CH6_R4	0x00003B10
MCS_CH6_R5	0x00003B14
MCS_CH6_R6	0x00003B18
MCS_CH6_R7	0x00003B1C
MCS_CH6_CTRL	0x00003B20
MCS_CH6_ACB	0x00003B24
MCS_CH6_MHB	0x00003B3C
MCS_CH6_PC	0x00003B40
MCS_CH6_IRQ_NOTIFY	0x00003B44
MCS_CH6_IRQ_EN	0x00003B48
MCS_CH6_IRQ_FORCINT	0x00003B4C
MCS_CH6_IRQ_MODE	0x00003B50
MCS_CH6_EIRQ_EN	0x00003B54
MCS_CH7_R0	0x00003B80
MCS_CH7_R1	0x00003B84
MCS_CH7_R2	0x00003B88
MCS_CH7_R3	0x00003B8C
MCS_CH7_R4	0x00003B90
MCS_CH7_R5	0x00003B94
MCS_CH7_R6	0x00003B98
MCS_CH7_R7	0x00003B9C
MCS_CH7_CTRL	0x00003BA0
MCS_CH7_ACB	0x00003BA4
MCS_CH7_MHB	0x00003BBC
MCS_CH7_PC	0x00003BC0
MCS_CH7_IRQ_NOTIFY	0x00003BC4
MCS_CH7_IRQ_EN	0x00003BC8
MCS_CH7_IRQ_FORCINT	0x00003BCC
MCS_CH7_IRQ_MODE	0x00003BD0
MCS_CH7_EIRQ_EN	0x00003BD4
SPE_CTRL_STAT	0x00005000
SPE_PAT	0x00005004
SPE_OUT_PAT0	0x00005008
SPE_OUT_PAT1	0x0000500C
SPE_OUT_PAT2	0x00005010
SPE_OUT_PAT3	0x00005014
SPE_OUT_PAT4	0x00005018
SPE_OUT_PAT5	0x0000501C

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
SPE_OUT_PAT6	0x00005020
SPE_OUT_PAT7	0x00005024
SPE_OUT_CTRL	0x00005028
SPE_IRQ_NOTIFY	0x0000502C
SPE_IRQ_EN	0x00005030
SPE_IRQ_FORCINT	0x00005034
SPE_IRQ_MODE	0x00005038
SPE_EIRQ_EN	0x0000503C
SPE_REV_CNT	0x00005040
SPE_REV_CMP	0x00005044
SPE_CTRL_STAT2	0x00005048
SPE_CMD	0x0000504C
CCM_ARP0_CTRL	0x00005200
CCM_ARP0_PROT	0x00005204
CCM_ARP1_CTRL	0x00005208
CCM_ARP1_PROT	0x0000520C
CCM_ARP2_CTRL	0x00005210
CCM_ARP2_PROT	0x00005214
CCM_ARP3_CTRL	0x00005218
CCM_ARP3_PROT	0x0000521C
CCM_ARP4_CTRL	0x00005220
CCM_ARP4_PROT	0x00005224
CCM_ARP5_CTRL	0x00005228
CCM_ARP5_PROT	0x0000522C
CCM_ARP6_CTRL	0x00005230
CCM_ARP6_PROT	0x00005234
CCM_ARP7_CTRL	0x00005238
CCM_ARP7_PROT	0x0000523C
CCM_ARP8_CTRL	0x00005240
CCM_ARP8_PROT	0x00005244
CCM_ARP9_CTRL	0x00005248
CCM_ARP9_PROT	0x0000524C
CCM_AEIM_STA	0x000053D8
CCM_HW_CONF	0x000053DC
CCM_TIM_AUX_IN_SRC	0x000053E0
CCM_EXT_CAP_EN	0x000053E4
CCM_TOM_OUT	0x000053E8
CCM_ATOM_OUT	0x000053EC

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
CCM_CMU_CLK_CFG	0x000053F0
CCM_CMU_FXCLK_CFG	0x000053F4
CCM_CFG	0x000053F8
CCM_PROT	0x000053FC
ADC_CH0_DATA	0x00005400
ADC_CH0_STA	0x00005404
ADC_CH1_DATA	0x00005408
ADC_CH1_STA	0x0000540C
ADC_CH2_DATA	0x00005410
ADC_CH2_STA	0x00005414
ADC_CH3_DATA	0x00005418
ADC_CH3_STA	0x0000541C
ADC_CH4_DATA	0x00005420
ADC_CH4_STA	0x00005424
ADC_CH5_DATA	0x00005428
ADC_CH5_STA	0x0000542C
ADC_CH6_DATA	0x00005430
ADC_CH6_STA	0x00005434
ADC_CH7_DATA	0x00005438
ADC_CH7_STA	0x0000543C
ADC_CH8_DATA	0x00005440
ADC_CH8_STA	0x00005444
ADC_CH9_DATA	0x00005448
ADC_CH9_STA	0x0000544C
ADC_CH10_DATA	0x00005450
ADC_CH10_STA	0x00005454
ADC_CH11_DATA	0x00005458
ADC_CH11_STA	0x0000545C
ADC_CH12_DATA	0x00005460
ADC_CH12_STA	0x00005464
ADC_CH13_DATA	0x00005468
ADC_CH13_STA	0x0000546C
ADC_CH14_DATA	0x00005470
ADC_CH14_STA	0x00005474
ADC_CH15_DATA	0x00005478
ADC_CH15_STA	0x0000547C
ADC_CH16_DATA	0x00005480
ADC_CH16_STA	0x00005484

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
ADC_CH17_DATA	0x00005488
ADC_CH17_STA	0x0000548C
ADC_CH18_DATA	0x00005490
ADC_CH18_STA	0x00005494
ADC_CH19_DATA	0x00005498
ADC_CH19_STA	0x0000549C
ADC_CH20_DATA	0x000054A0
ADC_CH20_STA	0x000054A4
ADC_CH21_DATA	0x000054A8
ADC_CH21_STA	0x000054AC
ADC_CH22_DATA	0x000054B0
ADC_CH22_STA	0x000054B4
ADC_CH23_DATA	0x000054B8
ADC_CH23_STA	0x000054BC
ADC_CH24_DATA	0x000054C0
ADC_CH24_STA	0x000054C4
ADC_CH25_DATA	0x000054C8
ADC_CH25_STA	0x000054CC
ADC_CH26_DATA	0x000054D0
ADC_CH26_STA	0x000054D4
ADC_CH27_DATA	0x000054D8
ADC_CH27_STA	0x000054DC
ADC_CH28_DATA	0x000054E0
ADC_CH28_STA	0x000054E4
ADC_CH29_DATA	0x000054E8
ADC_CH29_STA	0x000054EC
ADC_CH30_DATA	0x000054F0
ADC_CH30_STA	0x000054F4
ADC_CH31_DATA	0x000054F8
ADC_CH31_STA	0x000054FC
ADC_CH32_DATA	0x00005500
ADC_CH32_STA	0x00005504
ADC_CH33_DATA	0x00005508
ADC_CH33_STA	0x0000550C
ADC_CH34_DATA	0x00005510
ADC_CH34_STA	0x00005514
ADC_CH35_DATA	0x00005518
ADC_CH35_STA	0x0000551C

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
ADC_CH36_DATA	0x00005520
ADC_CH36_STA	0x00005524
ADC_CH37_DATA	0x00005528
ADC_CH37_STA	0x0000552C
ADC_CH38_DATA	0x00005530
ADC_CH38_STA	0x00005534
ADC_CH39_DATA	0x00005538
ADC_CH39_STA	0x0000553C
ADC_CH40_DATA	0x00005540
ADC_CH40_STA	0x00005544
ADC_CH41_DATA	0x00005548
ADC_CH41_STA	0x0000554C
ADC_CH42_DATA	0x00005550
ADC_CH42_STA	0x00005554
ADC_CH43_DATA	0x00005558
ADC_CH43_STA	0x0000555C
ADC_CH44_DATA	0x00005560
ADC_CH44_STA	0x00005564
ADC_CH45_DATA	0x00005568
ADC_CH45_STA	0x0000556C
ADC_CH46_DATA	0x00005570
ADC_CH46_STA	0x00005574
ADC_CH47_DATA	0x00005578
ADC_CH47_STA	0x0000557C
ADC_CH48_DATA	0x00005580
ADC_CH48_STA	0x00005584
ADC_CH49_DATA	0x00005588
ADC_CH49_STA	0x0000558C
ADC_CH50_DATA	0x00005590
ADC_CH50_STA	0x00005594
ADC_CH51_DATA	0x00005598
ADC_CH51_STA	0x0000559C
ADC_CH52_DATA	0x000055A0
ADC_CH52_STA	0x000055A4
ADC_CH53_DATA	0x000055A8
ADC_CH53_STA	0x000055AC
ADC_CH54_DATA	0x000055B0
ADC_CH54_STA	0x000055B4

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
ADC_CH55_DATA	0x000055B8
ADC_CH55_STA	0x000055BC
ADC_CH56_DATA	0x000055C0
ADC_CH56_STA	0x000055C4
ADC_CH57_DATA	0x000055C8
ADC_CH57_STA	0x000055CC
ADC_CH58_DATA	0x000055D0
ADC_CH58_STA	0x000055D4
ADC_CH59_DATA	0x000055D8
ADC_CH59_STA	0x000055DC
ADC_CH60_DATA	0x000055E0
ADC_CH60_STA	0x000055E4
ADC_CH61_DATA	0x000055E8
ADC_CH61_STA	0x000055EC
ADC_CH62_DATA	0x000055F0
ADC_CH62_STA	0x000055F4
ADC_CH63_DATA	0x000055F8
ADC_CH63_STA	0x000055FC
F2A_CH0_ARU_RD_FIFO	0x00005800
F2A_CH1_ARU_RD_FIFO	0x00005804
F2A_CH2_ARU_RD_FIFO	0x00005808
F2A_CH3_ARU_RD_FIFO	0x0000580C
F2A_CH4_ARU_RD_FIFO	0x00005810
F2A_CH5_ARU_RD_FIFO	0x00005814
F2A_CH6_ARU_RD_FIFO	0x00005818
F2A_CH7_ARU_RD_FIFO	0x0000581C
F2A_CH0_STR_CFG	0x00005820
F2A_CH1_STR_CFG	0x00005824
F2A_CH2_STR_CFG	0x00005828
F2A_CH3_STR_CFG	0x0000582C
F2A_CH4_STR_CFG	0x00005830
F2A_CH5_STR_CFG	0x00005834
F2A_CH6_STR_CFG	0x00005838
F2A_CH7_STR_CFG	0x0000583C
F2A_ENABLE	0x00005840
F2A_CTRL	0x00005844
AFD_CH0_BUF_ACC	0x00005880
AFD_CH1_BUF_ACC	0x00005890

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
AFD_CH2_BUF_ACC	0x000058A0
AFD_CH3_BUF_ACC	0x000058B0
AFD_CH4_BUF_ACC	0x000058C0
AFD_CH5_BUF_ACC	0x000058D0
AFD_CH6_BUF_ACC	0x000058E0
AFD_CH7_BUF_ACC	0x000058F0
FIFO_CH0_CTRL	0x00005C00
FIFO_CH0_END_ADDR	0x00005C04
FIFO_CH0_START_ADDR	0x00005C08
FIFO_CH0_UPPER_WM	0x00005C0C
FIFO_CH0_LOWER_WM	0x00005C10
FIFO_CH0_STATUS	0x00005C14
FIFO_CH0_FILL_LEVEL	0x00005C18
FIFO_CH0_WR_PTR	0x00005C1C
FIFO_CH0_RD_PTR	0x00005C20
FIFO_CH0_IRQ_NOTIFY	0x00005C24
FIFO_CH0_IRQ_EN	0x00005C28
FIFO_CH0_IRQ_FORCINT	0x00005C2C
FIFO_CH0_IRQ_MODE	0x00005C30
FIFO_CH0_EIRQ_EN	0x00005C34
FIFO_CH1_CTRL	0x00005C40
FIFO_CH1_END_ADDR	0x00005C44
FIFO_CH1_START_ADDR	0x00005C48
FIFO_CH1_UPPER_WM	0x00005C4C
FIFO_CH1_LOWER_WM	0x00005C50
FIFO_CH1_STATUS	0x00005C54
FIFO_CH1_FILL_LEVEL	0x00005C58
FIFO_CH1_WR_PTR	0x00005C5C
FIFO_CH1_RD_PTR	0x00005C60
FIFO_CH1_IRQ_NOTIFY	0x00005C64
FIFO_CH1_IRQ_EN	0x00005C68
FIFO_CH1_IRQ_FORCINT	0x00005C6C
FIFO_CH1_IRQ_MODE	0x00005C70
FIFO_CH1_EIRQ_EN	0x00005C74
FIFO_CH2_CTRL	0x00005C80
FIFO_CH2_END_ADDR	0x00005C84
FIFO_CH2_START_ADDR	0x00005C88
FIFO_CH2_UPPER_WM	0x00005C8C

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
FIFO_CH2_LOWER_WM	0x00005C90
FIFO_CH2_STATUS	0x00005C94
FIFO_CH2_FILL_LEVEL	0x00005C98
FIFO_CH2_WR_PTR	0x00005C9C
FIFO_CH2_RD_PTR	0x00005CA0
FIFO_CH2_IRQ_NOTIFY	0x00005CA4
FIFO_CH2_IRQ_EN	0x00005CA8
FIFO_CH2_IRQ_FORCINT	0x00005CAC
FIFO_CH2_IRQ_MODE	0x00005CB0
FIFO_CH2_EIRQ_EN	0x00005CB4
FIFO_CH3_CTRL	0x00005CC0
FIFO_CH3_END_ADDR	0x00005CC4
FIFO_CH3_START_ADDR	0x00005CC8
FIFO_CH3_UPPER_WM	0x00005CCC
FIFO_CH3_LOWER_WM	0x00005CD0
FIFO_CH3_STATUS	0x00005CD4
FIFO_CH3_FILL_LEVEL	0x00005CD8
FIFO_CH3_WR_PTR	0x00005CDC
FIFO_CH3_RD_PTR	0x00005CE0
FIFO_CH3_IRQ_NOTIFY	0x00005CE4
FIFO_CH3_IRQ_EN	0x00005CE8
FIFO_CH3_IRQ_FORCINT	0x00005CEC
FIFO_CH3_IRQ_MODE	0x00005CF0
FIFO_CH3_EIRQ_EN	0x00005CF4
FIFO_CH4_CTRL	0x00005D00
FIFO_CH4_END_ADDR	0x00005D04
FIFO_CH4_START_ADDR	0x00005D08
FIFO_CH4_UPPER_WM	0x00005D0C
FIFO_CH4_LOWER_WM	0x00005D10
FIFO_CH4_STATUS	0x00005D14
FIFO_CH4_FILL_LEVEL	0x00005D18
FIFO_CH4_WR_PTR	0x00005D1C
FIFO_CH4_RD_PTR	0x00005D20
FIFO_CH4_IRQ_NOTIFY	0x00005D24
FIFO_CH4_IRQ_EN	0x00005D28
FIFO_CH4_IRQ_FORCINT	0x00005D2C
FIFO_CH4_IRQ_MODE	0x00005D30
FIFO_CH4_EIRQ_EN	0x00005D34

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
FIFO_CH5_CTRL	0x00005D40
FIFO_CH5_END_ADDR	0x00005D44
FIFO_CH5_START_ADDR	0x00005D48
FIFO_CH5_UPPER_WM	0x00005D4C
FIFO_CH5_LOWER_WM	0x00005D50
FIFO_CH5_STATUS	0x00005D54
FIFO_CH5_FILL_LEVEL	0x00005D58
FIFO_CH5_WR_PTR	0x00005D5C
FIFO_CH5_RD_PTR	0x00005D60
FIFO_CH5_IRQ_NOTIFY	0x00005D64
FIFO_CH5_IRQ_EN	0x00005D68
FIFO_CH5_IRQ_FORCINT	0x00005D6C
FIFO_CH5_IRQ_MODE	0x00005D70
FIFO_CH5_EIRQ_EN	0x00005D74
FIFO_CH6_CTRL	0x00005D80
FIFO_CH6_END_ADDR	0x00005D84
FIFO_CH6_START_ADDR	0x00005D88
FIFO_CH6_UPPER_WM	0x00005D8C
FIFO_CH6_LOWER_WM	0x00005D90
FIFO_CH6_STATUS	0x00005D94
FIFO_CH6_FILL_LEVEL	0x00005D98
FIFO_CH6_WR_PTR	0x00005D9C
FIFO_CH6_RD_PTR	0x00005DA0
FIFO_CH6_IRQ_NOTIFY	0x00005DA4
FIFO_CH6_IRQ_EN	0x00005DA8
FIFO_CH6_IRQ_FORCINT	0x00005DAC
FIFO_CH6_IRQ_MODE	0x00005DB0
FIFO_CH6_EIRQ_EN	0x00005DB4
FIFO_CH7_CTRL	0x00005DC0
FIFO_CH7_END_ADDR	0x00005DC4
FIFO_CH7_START_ADDR	0x00005DC8
FIFO_CH7_UPPER_WM	0x00005DCC
FIFO_CH7_LOWER_WM	0x00005DD0
FIFO_CH7_STATUS	0x00005DD4
FIFO_CH7_FILL_LEVEL	0x00005DD8
FIFO_CH7_WR_PTR	0x00005DDC
FIFO_CH7_RD_PTR	0x00005DE0
FIFO_CH7_IRQ_NOTIFY	0x00005DE4

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
FIFO_CH7_IRQ_EN	0x00005DE8
FIFO_CH7_IRQ_FORCINT	0x00005DEC
FIFO_CH7_IRQ_MODE	0x00005DF0
FIFO_CH7_EIRQ_EN	0x00005DF4
CMU_CLK_EN	0x00007000
CMU_GCLK_NUM	0x00007004
CMU_GCLK_DEN	0x00007008
CMU_CLK_0_CTRL	0x0000700C
CMU_CLK_1_CTRL	0x00007010
CMU_CLK_2_CTRL	0x00007014
CMU_CLK_3_CTRL	0x00007018
CMU_CLK_4_CTRL	0x0000701C
CMU_CLK_5_CTRL	0x00007020
CMU_CLK_6_CTRL	0x00007024
CMU_CLK_7_CTRL	0x00007028
CMU_ECLK_0_NUM	0x0000702C
CMU_ECLK_0_DEN	0x00007030
CMU_ECLK_1_NUM	0x00007034
CMU_ECLK_1_DEN	0x00007038
CMU_ECLK_2_NUM	0x0000703C
CMU_ECLK_2_DEN	0x00007040
CMU_FXCLK_CTRL	0x00007044
CMU_GLB_CTRL	0x00007048
CMU_CLK_CTRL	0x0000704C
TBU_CHEN	0x00007080
TBU_CH0_CTRL	0x00007084
TBU_CH0_BASE	0x00007088
TBU_CH1_CTRL	0x0000708C
TBU_CH1_BASE	0x00007090
TBU_CH2_CTRL	0x00007094
TBU_CH2_BASE	0x00007098
TBU_CH3_CTRL	0x0000709C
TBU_CH3_BASE	0x000070A0
TBU_CH3_BASE_MARK	0x000070A4
TBU_CH3_BASE_CAPTURE	0x000070A8
BRC_SRC_0_ADDR	0x00007100
BRC_SRC_0_DEST	0x00007104
BRC_SRC_1_ADDR	0x00007108

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
BRC_SRC_1_DEST	0x0000710C
BRC_SRC_2_ADDR	0x00007110
BRC_SRC_2_DEST	0x00007114
BRC_SRC_3_ADDR	0x00007118
BRC_SRC_3_DEST	0x0000711C
BRC_SRC_4_ADDR	0x00007120
BRC_SRC_4_DEST	0x00007124
BRC_SRC_5_ADDR	0x00007128
BRC_SRC_5_DEST	0x0000712C
BRC_SRC_6_ADDR	0x00007130
BRC_SRC_6_DEST	0x00007134
BRC_SRC_7_ADDR	0x00007138
BRC_SRC_7_DEST	0x0000713C
BRC_SRC_8_ADDR	0x00007140
BRC_SRC_8_DEST	0x00007144
BRC_SRC_9_ADDR	0x00007148
BRC_SRC_9_DEST	0x0000714C
BRC_SRC_10_ADDR	0x00007150
BRC_SRC_10_DEST	0x00007154
BRC_SRC_11_ADDR	0x00007158
BRC_SRC_11_DEST	0x0000715C
BRC_IRQ_NOTIFY	0x00007160
BRC_IRQ_EN	0x00007164
BRC_IRQ_FORCINT	0x00007168
BRC_IRQ_MODE	0x0000716C
BRC_RST	0x00007170
BRC_EIRQ_EN	0x00007174
ARU_ACCESS	0x00007180
ARU_DATA_H	0x00007184
ARU_DATA_L	0x00007188
ARU_DBG_ACCESS0	0x0000718C
ARU_DBG_DATA0_H	0x00007190
ARU_DBG_DATA0_L	0x00007194
ARU_DBG_ACCESS1	0x00007198
ARU_DBG_DATA1_H	0x0000719C
ARU_DBG_DATA1_L	0x000071A0
ARU_IRQ_NOTIFY	0x000071A4
ARU_IRQ_EN	0x000071A8

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
ARU_IRQ_FORCINT	0x000071AC
ARU_IRQ_MODE	0x000071B0
ARU_CADDR_END	0x000071B4
ARU_reserved	0x000071B8
ARU_CTRL	0x000071BC
ARU_0_DYN_CTRL	0x000071C0
ARU_1_DYN_CTRL	0x000071C4
ARU_0_DYN_ROUTE_LOW	0x000071C8
ARU_1_DYN_ROUTE_LOW	0x000071CC
ARU_0_DYN_ROUTE_HIGH	0x000071D0
ARU_1_DYN_ROUTE_HIGH	0x000071D4
ARU_0_DYN_ROUTE_SR_LOW	0x000071D8
ARU_1_DYN_ROUTE_SR_LOW	0x000071DC
ARU_0_DYN_ROUTE_SR_HIGH	0x000071E0
ARU_1_DYN_ROUTE_SR_HIGH	0x000071E4
ARU_0_DYN_RDADDR	0x000071E8
ARU_1_DYN_RDADDR	0x000071EC
ARU_CADDR	0x000071FC
MCFG_CTRL	0x00007200
MAP_CTRL	0x00007240
MON_STATUS	0x00007280
MON_ACTIVITY_0	0x00007284
MON_ACTIVITY_1	0x00007288
MON_ACTIVITY_MCS0	0x0000728C
MON_ACTIVITY_MCS1	0x00007290
MON_ACTIVITY_MCS2	0x00007294
MON_ACTIVITY_MCS3	0x00007298
MON_ACTIVITY_MCS4	0x0000729C
MON_ACTIVITY_MCS5	0x000072A0
MON_ACTIVITY_MCS6	0x000072A4
MON_ACTIVITY_MCS7	0x000072A8
MON_ACTIVITY_MCS8	0x000072AC
MON_ACTIVITY_MCS9	0x000072B0
CMP_EN	0x000072C0
CMP_IRQ_NOTIFY	0x000072C4
CMP_IRQ_EN	0x000072C8
CMP_IRQ_FORCINT	0x000072CC
CMP_IRQ_MODE	0x000072D0

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
CMP_EIRQ_EN	0x000072D4
GTM_REV	0x00007300
GTM_RST	0x00007304
GTM_CTRL	0x00007308
GTM_AEI_ADDR_XPT	0x0000730C
GTM_IRQ_NOTIFY	0x00007310
GTM_IRQ_EN	0x00007314
GTM_IRQ_FORCINT	0x00007318
GTM_IRQ_MODE	0x0000731C
GTM_EIRQ_EN	0x00007320
LEG_GTM_HW_CONF	0x00007324
GTM_CFG	0x00007328
GTM_AEI_STA_XPT	0x0000732C
GTM_TIM0_AUX_IN_SRC	0x00007340
GTM_TIM1_AUX_IN_SRC	0x00007344
GTM_TIM2_AUX_IN_SRC	0x00007348
GTM_TIM3_AUX_IN_SRC	0x0000734C
GTM_TIM4_AUX_IN_SRC	0x00007350
GTM_TIM5_AUX_IN_SRC	0x00007354
GTM_TIM6_AUX_IN_SRC	0x00007358
GTM_EXT_CAP_EN_0	0x0000735C
GTM_EXT_CAP_EN_1	0x00007360
GTM_EXT_CAP_EN_2	0x00007364
GTM_EXT_CAP_EN_3	0x00007368
GTM_EXT_CAP_EN_4	0x0000736C
GTM_EXT_CAP_EN_5	0x00007370
GTM_EXT_CAP_EN_6	0x00007374
GTM_EXT_CAP_EN_7	0x00007378
GTM_TOM0_OUT	0x00007380
GTM_TOM1_OUT	0x00007384
GTM_TOM2_OUT	0x00007388
GTM_TOM3_OUT	0x0000738C
GTM_TOM4_OUT	0x00007390
GTM_TOM5_OUT	0x00007394
GTM_ATOM0_OUT	0x00007398
GTM_ATOM2_OUT	0x0000739C
GTM_ATOM4_OUT	0x000073A0
GTM_ATOM6_OUT	0x000073A4

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
GTM_ATOM8_OUT	0x000073A8
GTM_ATOM10_OUT	0x000073AC
GTM_CLS_CLK_CFG	0x000073B0
MCS2DPLL_DEB0	0x00007800
MCS2DPLL_DEB1	0x00007804
MCS2DPLL_DEB2	0x00007808
MCS2DPLL_DEB3	0x0000780C
MCS2DPLL_DEB4	0x00007810
MCS2DPLL_DEB5	0x00007814
MCS2DPLL_DEB6	0x00007818
MCS2DPLL_DEB7	0x0000781C
MCS2DPLL_DEB8	0x00007820
MCS2DPLL_DEB9	0x00007824
MCS2DPLL_DEB10	0x00007828
MCS2DPLL_DEB11	0x0000782C
MCS2DPLL_DEB12	0x00007830
MCS2DPLL_DEB13	0x00007834
MCS2DPLL_DEB14	0x00007838
MCS2DPLL_DEB15	0x0000783C
DPLL_CTRL_0	0x00008000
DPLL_CTRL_1	0x00008004
DPLL_CTRL_2	0x00008008
DPLL_CTRL_3	0x0000800C
DPLL_CTRL_4	0x00008010
DPLL_CTRL_5	0x00008014
DPLL_ACT_STA	0x00008018
DPLL_OSW	0x0000801C
DPLL_AOSV_2	0x00008020
DPLL_APT	0x00008024
DPLL_APS	0x00008028
DPLL_APT_2C	0x0000802C
DPLL_APS_1C3	0x00008030
DPLL_NUTC	0x00008034
DPLL_NUSC	0x00008038
DPLL_NTI_CNT	0x0000803C
DPLL_IRQ_NOTIFY	0x00008040
DPLL_IRQ_EN	0x00008044
DPLL_IRQ_FORCINT	0x00008048

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_IRQ_MODE	0x0000804C
DPLL_EIRQ_EN	0x00008050
DPLL_INC_CNT1	0x000080B0
DPLL_INC_CNT2	0x000080B4
DPLL_APT_SYNC	0x000080B8
DPLL_APS_SYNC	0x000080BC
DPLL_TBU_TS0_T	0x000080C0
DPLL_TBU_TS0_S	0x000080C4
DPLL_ADD_IN_LD1	0x000080C8
DPLL_ADD_IN_LD2	0x000080CC
DPLL_STATUS	0x000080FC
DPLL_ID_PMTR_0	0x00008100
DPLL_ID_PMTR_1	0x00008104
DPLL_ID_PMTR_2	0x00008108
DPLL_ID_PMTR_3	0x0000810C
DPLL_ID_PMTR_4	0x00008110
DPLL_ID_PMTR_5	0x00008114
DPLL_ID_PMTR_6	0x00008118
DPLL_ID_PMTR_7	0x0000811C
DPLL_ID_PMTR_8	0x00008120
DPLL_ID_PMTR_9	0x00008124
DPLL_ID_PMTR_10	0x00008128
DPLL_ID_PMTR_11	0x0000812C
DPLL_ID_PMTR_12	0x00008130
DPLL_ID_PMTR_13	0x00008134
DPLL_ID_PMTR_14	0x00008138
DPLL_ID_PMTR_15	0x0000813C
DPLL_ID_PMTR_16	0x00008140
DPLL_ID_PMTR_17	0x00008144
DPLL_ID_PMTR_18	0x00008148
DPLL_ID_PMTR_19	0x0000814C
DPLL_ID_PMTR_20	0x00008150
DPLL_ID_PMTR_21	0x00008154
DPLL_ID_PMTR_22	0x00008158
DPLL_ID_PMTR_23	0x0000815C
DPLL_ID_PMTR_24	0x00008160
DPLL_ID_PMTR_25	0x00008164
DPLL_ID_PMTR_26	0x00008168

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_ID_PMTR_27	0x0000816C
DPLL_ID_PMTR_28	0x00008170
DPLL_ID_PMTR_29	0x00008174
DPLL_ID_PMTR_30	0x00008178
DPLL_ID_PMTR_31	0x0000817C
DPLL_CTRL_0_SHADOW_TRIGGER	0x000081E0
DPLL_CTRL_0_SHADOW_STATE	0x000081E4
DPLL_CTRL_1_SHADOW_TRIGGER	0x000081E8
DPLL_CTRL_1_SHADOW_STATE	0x000081EC
DPLL_RAM_INI	0x000081FC
DPLL_RR1A	0x00008200
DPLL_RR1A_END	0x000083FC
DPLL_TS_T	0x00008400
DPLL_TS_T_OLD	0x00008404
DPLL_FTV_T	0x00008408
DPLL_TS_S	0x00008410
DPLL_TS_S_OLD	0x00008414
DPLL_FTV_S	0x00008418
DPLL_THMI	0x00008420
DPLL_THMA	0x00008424
DPLL_THVAL	0x00008428
DPLL_TOV	0x00008430
DPLL_TOV_S	0x00008434
DPLL_ADD_IN_CAL1	0x00008438
DPLL_ADD_IN_CAL2	0x0000843C
DPLL_MPVAL1	0x00008440
DPLL_MPVAL2	0x00008444
DPLL_NMB_T_TAR	0x00008448
DPLL_NMB_T_TAR_OLD	0x0000844C
DPLL_NMB_S_TAR	0x00008450
DPLL_NMB_S_TAR_OLD	0x00008454
DPLL_RCDT_TX	0x00008460
DPLL_RCDT_SX	0x00008464
DPLL_RCDT_TX_NOM	0x00008468
DPLL_RCDT_SX_NOM	0x0000846C
DPLL_RDT_T_ACT	0x00008470
DPLL_RDT_S_ACT	0x00008474
DPLL_DT_T_ACT	0x00008478

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_DT_S_ACT	0x0000847C
DPLL_EDT_T	0x00008480
DPLL_MEDT_T	0x00008484
DPLL_EDT_S	0x00008488
DPLL_MEDT_S	0x0000848C
DPLL_CDT_TX	0x00008490
DPLL_CDT_SX	0x00008494
DPLL_CDT_TX_NOM	0x00008498
DPLL_CDT_SX_NOM	0x0000849C
DPLL_TLR	0x000084A0
DPLL_SLR	0x000084A4
DPLL_PDT_0	0x00008500
DPLL_PDT_1	0x00008504
DPLL_PDT_2	0x00008508
DPLL_PDT_3	0x0000850C
DPLL_PDT_4	0x00008510
DPLL_PDT_5	0x00008514
DPLL_PDT_6	0x00008518
DPLL_PDT_7	0x0000851C
DPLL_PDT_8	0x00008520
DPLL_PDT_9	0x00008524
DPLL_PDT_10	0x00008528
DPLL_PDT_11	0x0000852C
DPLL_PDT_12	0x00008530
DPLL_PDT_13	0x00008534
DPLL_PDT_14	0x00008538
DPLL_PDT_15	0x0000853C
DPLL_PDT_16	0x00008540
DPLL_PDT_17	0x00008544
DPLL_PDT_18	0x00008548
DPLL_PDT_19	0x0000854C
DPLL_PDT_20	0x00008550
DPLL_PDT_21	0x00008554
DPLL_PDT_22	0x00008558
DPLL_PDT_23	0x0000855C
DPLL_PDT_24	0x00008560
DPLL_PDT_25	0x00008564
DPLL_PDT_26	0x00008568

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_PDT_27	0x0000856C
DPLL_PDT_28	0x00008570
DPLL_PDT_29	0x00008574
DPLL_PDT_30	0x00008578
DPLL_PDT_31	0x0000857C
DPLL_MLS1	0x000085C0
DPLL_MLS2	0x000085C4
DPLL_CNT_NUM_1	0x000085C8
DPLL_CNT_NUM_2	0x000085CC
DPLL_PVT	0x000085D0
DPLL_PSTC	0x000085E0
DPLL_PSSC	0x000085E4
DPLL_PSTM	0x000085E8
DPLL_PSTM_OLD	0x000085EC
DPLL_PSSM	0x000085F0
DPLL_PSSM_OLD	0x000085F4
DPLL_NMB_T	0x000085F8
DPLL_NMB_S	0x000085FC
DPLL_RDT_S0	0x00008600
DPLL_RDT_S1	0x00008604
DPLL_RDT_S2	0x00008608
DPLL_RDT_S3	0x0000860C
DPLL_RDT_S4	0x00008610
DPLL_RDT_S5	0x00008614
DPLL_RDT_S6	0x00008618
DPLL_RDT_S7	0x0000861C
DPLL_RDT_S8	0x00008620
DPLL_RDT_S9	0x00008624
DPLL_RDT_S10	0x00008628
DPLL_RDT_S11	0x0000862C
DPLL_RDT_S12	0x00008630
DPLL_RDT_S13	0x00008634
DPLL_RDT_S14	0x00008638
DPLL_RDT_S15	0x0000863C
DPLL_RDT_S16	0x00008640
DPLL_RDT_S17	0x00008644
DPLL_RDT_S18	0x00008648
DPLL_RDT_S19	0x0000864C

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_RDT_S20	0x00008650
DPLL_RDT_S21	0x00008654
DPLL_RDT_S22	0x00008658
DPLL_RDT_S23	0x0000865C
DPLL_RDT_S24	0x00008660
DPLL_RDT_S25	0x00008664
DPLL_RDT_S26	0x00008668
DPLL_RDT_S27	0x0000866C
DPLL_RDT_S28	0x00008670
DPLL_RDT_S29	0x00008674
DPLL_RDT_S30	0x00008678
DPLL_RDT_S31	0x0000867C
DPLL_RDT_S32	0x00008680
DPLL_RDT_S33	0x00008684
DPLL_RDT_S34	0x00008688
DPLL_RDT_S35	0x0000868C
DPLL_RDT_S36	0x00008690
DPLL_RDT_S37	0x00008694
DPLL_RDT_S38	0x00008698
DPLL_RDT_S39	0x0000869C
DPLL_RDT_S40	0x000086A0
DPLL_RDT_S41	0x000086A4
DPLL_RDT_S42	0x000086A8
DPLL_RDT_S43	0x000086AC
DPLL_RDT_S44	0x000086B0
DPLL_RDT_S45	0x000086B4
DPLL_RDT_S46	0x000086B8
DPLL_RDT_S47	0x000086BC
DPLL_RDT_S48	0x000086C0
DPLL_RDT_S49	0x000086C4
DPLL_RDT_S50	0x000086C8
DPLL_RDT_S51	0x000086CC
DPLL_RDT_S52	0x000086D0
DPLL_RDT_S53	0x000086D4
DPLL_RDT_S54	0x000086D8
DPLL_RDT_S55	0x000086DC
DPLL_RDT_S56	0x000086E0
DPLL_RDT_S57	0x000086E4

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_RDT_S58	0x000086E8
DPLL_RDT_S59	0x000086EC
DPLL_RDT_S60	0x000086F0
DPLL_RDT_S61	0x000086F4
DPLL_RDT_S62	0x000086F8
DPLL_RDT_S63	0x000086FC
DPLL_TSF_S0	0x00008700
DPLL_TSF_S1	0x00008704
DPLL_TSF_S2	0x00008708
DPLL_TSF_S3	0x0000870C
DPLL_TSF_S4	0x00008710
DPLL_TSF_S5	0x00008714
DPLL_TSF_S6	0x00008718
DPLL_TSF_S7	0x0000871C
DPLL_TSF_S8	0x00008720
DPLL_TSF_S9	0x00008724
DPLL_TSF_S10	0x00008728
DPLL_TSF_S11	0x0000872C
DPLL_TSF_S12	0x00008730
DPLL_TSF_S13	0x00008734
DPLL_TSF_S14	0x00008738
DPLL_TSF_S15	0x0000873C
DPLL_TSF_S16	0x00008740
DPLL_TSF_S17	0x00008744
DPLL_TSF_S18	0x00008748
DPLL_TSF_S19	0x0000874C
DPLL_TSF_S20	0x00008750
DPLL_TSF_S21	0x00008754
DPLL_TSF_S22	0x00008758
DPLL_TSF_S23	0x0000875C
DPLL_TSF_S24	0x00008760
DPLL_TSF_S25	0x00008764
DPLL_TSF_S26	0x00008768
DPLL_TSF_S27	0x0000876C
DPLL_TSF_S28	0x00008770
DPLL_TSF_S29	0x00008774
DPLL_TSF_S30	0x00008778
DPLL_TSF_S31	0x0000877C

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_TSF_S32	0x00008780
DPLL_TSF_S33	0x00008784
DPLL_TSF_S34	0x00008788
DPLL_TSF_S35	0x0000878C
DPLL_TSF_S36	0x00008790
DPLL_TSF_S37	0x00008794
DPLL_TSF_S38	0x00008798
DPLL_TSF_S39	0x0000879C
DPLL_TSF_S40	0x000087A0
DPLL_TSF_S41	0x000087A4
DPLL_TSF_S42	0x000087A8
DPLL_TSF_S43	0x000087AC
DPLL_TSF_S44	0x000087B0
DPLL_TSF_S45	0x000087B4
DPLL_TSF_S46	0x000087B8
DPLL_TSF_S47	0x000087BC
DPLL_TSF_S48	0x000087C0
DPLL_TSF_S49	0x000087C4
DPLL_TSF_S50	0x000087C8
DPLL_TSF_S51	0x000087CC
DPLL_TSF_S52	0x000087D0
DPLL_TSF_S53	0x000087D4
DPLL_TSF_S54	0x000087D8
DPLL_TSF_S55	0x000087DC
DPLL_TSF_S56	0x000087E0
DPLL_TSF_S57	0x000087E4
DPLL_TSF_S58	0x000087E8
DPLL_TSF_S59	0x000087EC
DPLL_TSF_S60	0x000087F0
DPLL_TSF_S61	0x000087F4
DPLL_TSF_S62	0x000087F8
DPLL_TSF_S63	0x000087FC
DPLL_ADT_S0	0x00008800
DPLL_ADT_S1	0x00008804
DPLL_ADT_S2	0x00008808
DPLL_ADT_S3	0x0000880C
DPLL_ADT_S4	0x00008810
DPLL_ADT_S5	0x00008814

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_ADT_S6	0x00008818
DPLL_ADT_S7	0x0000881C
DPLL_ADT_S8	0x00008820
DPLL_ADT_S9	0x00008824
DPLL_ADT_S10	0x00008828
DPLL_ADT_S11	0x0000882C
DPLL_ADT_S12	0x00008830
DPLL_ADT_S13	0x00008834
DPLL_ADT_S14	0x00008838
DPLL_ADT_S15	0x0000883C
DPLL_ADT_S16	0x00008840
DPLL_ADT_S17	0x00008844
DPLL_ADT_S18	0x00008848
DPLL_ADT_S19	0x0000884C
DPLL_ADT_S20	0x00008850
DPLL_ADT_S21	0x00008854
DPLL_ADT_S22	0x00008858
DPLL_ADT_S23	0x0000885C
DPLL_ADT_S24	0x00008860
DPLL_ADT_S25	0x00008864
DPLL_ADT_S26	0x00008868
DPLL_ADT_S27	0x0000886C
DPLL_ADT_S28	0x00008870
DPLL_ADT_S29	0x00008874
DPLL_ADT_S30	0x00008878
DPLL_ADT_S31	0x0000887C
DPLL_ADT_S32	0x00008880
DPLL_ADT_S33	0x00008884
DPLL_ADT_S34	0x00008888
DPLL_ADT_S35	0x0000888C
DPLL_ADT_S36	0x00008890
DPLL_ADT_S37	0x00008894
DPLL_ADT_S38	0x00008898
DPLL_ADT_S39	0x0000889C
DPLL_ADT_S40	0x000088A0
DPLL_ADT_S41	0x000088A4
DPLL_ADT_S42	0x000088A8
DPLL_ADT_S43	0x000088AC

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_ADT_S44	0x000088B0
DPLL_ADT_S45	0x000088B4
DPLL_ADT_S46	0x000088B8
DPLL_ADT_S47	0x000088BC
DPLL_ADT_S48	0x000088C0
DPLL_ADT_S49	0x000088C4
DPLL_ADT_S50	0x000088C8
DPLL_ADT_S51	0x000088CC
DPLL_ADT_S52	0x000088D0
DPLL_ADT_S53	0x000088D4
DPLL_ADT_S54	0x000088D8
DPLL_ADT_S55	0x000088DC
DPLL_ADT_S56	0x000088E0
DPLL_ADT_S57	0x000088E4
DPLL_ADT_S58	0x000088E8
DPLL_ADT_S59	0x000088EC
DPLL_ADT_S60	0x000088F0
DPLL_ADT_S61	0x000088F4
DPLL_ADT_S62	0x000088F8
DPLL_ADT_S63	0x000088FC
DPLL_DT_S0	0x00008900
DPLL_DT_S1	0x00008904
DPLL_DT_S2	0x00008908
DPLL_DT_S3	0x0000890C
DPLL_DT_S4	0x00008910
DPLL_DT_S5	0x00008914
DPLL_DT_S6	0x00008918
DPLL_DT_S7	0x0000891C
DPLL_DT_S8	0x00008920
DPLL_DT_S9	0x00008924
DPLL_DT_S10	0x00008928
DPLL_DT_S11	0x0000892C
DPLL_DT_S12	0x00008930
DPLL_DT_S13	0x00008934
DPLL_DT_S14	0x00008938
DPLL_DT_S15	0x0000893C
DPLL_DT_S16	0x00008940
DPLL_DT_S17	0x00008944

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_DT_S18	0x00008948
DPLL_DT_S19	0x0000894C
DPLL_DT_S20	0x00008950
DPLL_DT_S21	0x00008954
DPLL_DT_S22	0x00008958
DPLL_DT_S23	0x0000895C
DPLL_DT_S24	0x00008960
DPLL_DT_S25	0x00008964
DPLL_DT_S26	0x00008968
DPLL_DT_S27	0x0000896C
DPLL_DT_S28	0x00008970
DPLL_DT_S29	0x00008974
DPLL_DT_S30	0x00008978
DPLL_DT_S31	0x0000897C
DPLL_DT_S32	0x00008980
DPLL_DT_S33	0x00008984
DPLL_DT_S34	0x00008988
DPLL_DT_S35	0x0000898C
DPLL_DT_S36	0x00008990
DPLL_DT_S37	0x00008994
DPLL_DT_S38	0x00008998
DPLL_DT_S39	0x0000899C
DPLL_DT_S40	0x000089A0
DPLL_DT_S41	0x000089A4
DPLL_DT_S42	0x000089A8
DPLL_DT_S43	0x000089AC
DPLL_DT_S44	0x000089B0
DPLL_DT_S45	0x000089B4
DPLL_DT_S46	0x000089B8
DPLL_DT_S47	0x000089BC
DPLL_DT_S48	0x000089C0
DPLL_DT_S49	0x000089C4
DPLL_DT_S50	0x000089C8
DPLL_DT_S51	0x000089CC
DPLL_DT_S52	0x000089D0
DPLL_DT_S53	0x000089D4
DPLL_DT_S54	0x000089D8
DPLL_DT_S55	0x000089DC

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_DT_S56	0x000089E0
DPLL_DT_S57	0x000089E4
DPLL_DT_S58	0x000089E8
DPLL_DT_S59	0x000089EC
DPLL_DT_S60	0x000089F0
DPLL_DT_S61	0x000089F4
DPLL_DT_S62	0x000089F8
DPLL_DT_S63	0x000089FC
DPLL_TSAC0	0x00008E00
DPLL_TSAC1	0x00008E04
DPLL_TSAC2	0x00008E08
DPLL_TSAC3	0x00008E0C
DPLL_TSAC4	0x00008E10
DPLL_TSAC5	0x00008E14
DPLL_TSAC6	0x00008E18
DPLL_TSAC7	0x00008E1C
DPLL_TSAC8	0x00008E20
DPLL_TSAC9	0x00008E24
DPLL_TSAC10	0x00008E28
DPLL_TSAC11	0x00008E2C
DPLL_TSAC12	0x00008E30
DPLL_TSAC13	0x00008E34
DPLL_TSAC14	0x00008E38
DPLL_TSAC15	0x00008E3C
DPLL_TSAC16	0x00008E40
DPLL_TSAC17	0x00008E44
DPLL_TSAC18	0x00008E48
DPLL_TSAC19	0x00008E4C
DPLL_TSAC20	0x00008E50
DPLL_TSAC21	0x00008E54
DPLL_TSAC22	0x00008E58
DPLL_TSAC23	0x00008E5C
DPLL_TSAC24	0x00008E60
DPLL_TSAC25	0x00008E64
DPLL_TSAC26	0x00008E68
DPLL_TSAC27	0x00008E6C
DPLL_TSAC28	0x00008E70
DPLL_TSAC29	0x00008E74

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_TSAC30	0x00008E78
DPLL_TSAC31	0x00008E7C
DPLL_PSAC0	0x00008E80
DPLL_PSAC1	0x00008E84
DPLL_PSAC2	0x00008E88
DPLL_PSAC3	0x00008E8C
DPLL_PSAC4	0x00008E90
DPLL_PSAC5	0x00008E94
DPLL_PSAC6	0x00008E98
DPLL_PSAC7	0x00008E9C
DPLL_PSAC8	0x00008EA0
DPLL_PSAC9	0x00008EA4
DPLL_PSAC10	0x00008EA8
DPLL_PSAC11	0x00008EAC
DPLL_PSAC12	0x00008EB0
DPLL_PSAC13	0x00008EB4
DPLL_PSAC14	0x00008EB8
DPLL_PSAC15	0x00008EBC
DPLL_PSAC16	0x00008EC0
DPLL_PSAC17	0x00008EC4
DPLL_PSAC18	0x00008EC8
DPLL_PSAC19	0x00008ECC
DPLL_PSAC20	0x00008ED0
DPLL_PSAC21	0x00008ED4
DPLL_PSAC22	0x00008ED8
DPLL_PSAC23	0x00008EDC
DPLL_PSAC24	0x00008EE0
DPLL_PSAC25	0x00008EE4
DPLL_PSAC26	0x00008EE8
DPLL_PSAC27	0x00008EEC
DPLL_PSAC28	0x00008EF0
DPLL_PSAC29	0x00008EF4
DPLL_PSAC30	0x00008EF8
DPLL_PSAC31	0x00008EFC
DPLL_ACB_0	0x00008F00
DPLL_ACB_1	0x00008F04
DPLL_ACB_2	0x00008F08
DPLL_ACB_3	0x00008F0C

Table 188 MCS Master Interface Address Map (cont'd)

Register Label	Register Address
DPLL_ACB_4	0x00008F10
DPLL_ACB_5	0x00008F14
DPLL_ACB_6	0x00008F18
DPLL_ACB_7	0x00008F1C
DPLL_CTRL_11	0x00008F20
DPLL_THVAL2	0x00008F24
DPLL_TIDEL	0x00008F28
DPLL_SIDEL	0x00008F2C
DPLL_APS_SYNC_EXT	0x00008F30
DPLL_CTRL_EXT	0x00008F34
DPLL_APS_EXT	0x00008F38
DPLL_APS_1C3_EXT	0x00008F3C
DPLL_STA	0x00008F40
DPLL_INCF1_OFFSET	0x00008F44
DPLL_INCF2_OFFSET	0x00008F48
DPLL_DT_T_START	0x00008F4C
DPLL_DT_S_START	0x00008F50
DPLL_STA_MASK	0x00008F54
DPLL_STA_FLAG	0x00008F58
DPLL_INC_CNT1_MASK	0x00008F5C
DPLL_INC_CNT2_MASK	0x00008F60
DPLL_NUSC_EXT1	0x00008F64
DPLL_NUSC_EXT2	0x00008F68
DPLL_CTN_MIN	0x00008F6C
DPLL_CTN_MAX	0x00008F70
DPLL_CSN_MIN	0x00008F74
DPLL_CSN_MAX	0x00008F7C
DPLL_RR2	0x0000C000
DPLL_RR2_END	0x0000FFFC

28.25.25 General remarks

The GTM kernel is a pure 32bit module. Therefore accessing the GTM, shall be not bytewise or half word. This shall not be done and might have strange effects on the module like overwriting registers with scrapped information. For TriCore architecture conform accesses, no error will be shown to the outside.

28.26 Revision History

Table 189 Revision History

Reference	Change to Previous Version	Comment
v2.2.9		
	The GTM family specification has been part of a continuous improvement process. Due to the fact, that there have several changes, this chapter can be regarded as completely new. reworked	
v2.2.10		
	Cleanup of document: Inserted missing links to chapters; correction of typos; added missing frames for figures; format cleanup for notes in register and bit descriptions	
see next column for search text	Wrong name FXCMU_CLK replaced by CMU_FXCLK (see end of section “TGC Sub-unit”, and third paragraph in section “Continuous Counting Up-Down Mode” in chapter “Timer Output Module (TOM)”	
	Further description added to bitfields CCUxTC in registers TOM[i]_CH[x]_IRQ_NOTIFY and ATOM[i]_CH[x]_IRQ_NOTIFY	
	Extended GTM Legacy Address table (see “GTM Legacy Address table” in chapter “GTM Application constraints and limitations”)	
	Updated Table 186, GTM ARU Read IDs (see table “GTM Read IDs” in chapter “ARU Read ID”)	
	Corrected table on GTM Configuration (see table “GTM Configuration by AURIX TC3xx Product” in chapter “GTM Implementation”)	
	Corrected description for MCS Data Input Path; updated figure (see figure “MCS Data Input Path” and paragraph after this figure in chapter “GTM Data Exchange Registers”)	
v2.2.11		
	Cleaning up the document: In case of missinglinks, replaced by full text to be able to search for.	
Blockdiagram page 1	Removed QSPI, as not existing in any silicon.	
ARU Routing Concept 3 paragraphs below figure	Maximum Throuput Mode (MTM), before wrongly named.	
BRC_SRC_z_DEST	Bitfield EN_DEST runvariable changed	
BRC_IRQ_NOTIF	Data inconsistency occurred (new!) for channel x (new end)in MTM mode	

Table 189 Revision History (cont'd)

Reference	Change to Previous Version	Comment
BRC_IRQ_EN	DID_IRQ_ENx (x=0-11) new description	
CMU_CLK_EN	DID_IRQ_ENx (x=0-11) new Enable clock source 0x - now with x insteadof 0.	
CCMi_HW_CONF	Precise reset value	
TOM[i]_TGC[y]_FUPD_CTRL	RSTCN0_CHx (x=0-7) now with x instead of 0	
MCS2DPLL_DEBn	corrected addresses	
ICM_IRQG_ATOM_[k]_CI	now for all ATOMs	
ICM_IRQG_CLS_[k]_MEI	corrected index	
CMP_IRQ_FORCINT	corrected index in bitfield TRG_TBWCx	
RESET1	corrected spelling	
Port Connections	exchanged to see appendix, as port information has been moved	
GTM Outputs to Port Connections	chapter removed as not valid for low end devices.	
table MCSx (x=0..9) Trigger Assignments	MCSTRIG replaced by MCS_TRIG	
OTSC1	corrected spelling	
MCS Master Interface Address Map	corrected addresses and naming	
v2.2.11		
	no relevant changes for family specification	
v2.2.12		
	no relevant changes for family specification	

Table 189 Revision History (cont'd)

Reference	Change to Previous Version	Comment
v2.2.13		
	Changed overview table as TC33x will have fast clusters only.	
v2.2.14		
	Remarks inside the OCDS Registers, concerning status, if TBU channel 3, DPLL or MCS do not exist on a device.	
	CCMi_HW_CONF gets proper bit descriptions, as constants have been all 0x0.	
v2.2.15		
	No changes in this part of the specification	
v2.2.16		
	EXT_TRIG bitfield description changed in Register: TOM[i]_CH[x]_CTRL, ATOM[i]_CH[x]_CTRL in SOMP mode and ATOM[i]_CH[x]_CTRL. TIM_[x-1] exchanged by TRIG_[x-1]	
v2.2.17		
	Removed unmotivated questionmarks and replaced ist by is.	
v2.2.18		
	Corrected Spelling bugs	
	IRQ_Notify providing hint, that this bit is rw and has to be written.	
v2.2.19		
	SPE bit ADIR also now in description as previously described as DIR (identical with signal name)	
	EDIS description including details	
	GTM_CCMi_TIM_AUX_IN_SRC including details	
	CCMi_CFG now included for all clusters	
	CDTM[i]_DTM[j]_CH_CTRL1.I1SEL_n registers corrected to proper bitfield meaning.	
	Minor changes, without any impact on functionality	
v2.2.20		
	Corrected description for ATOM[i]_AGC_ENDIS_CTRL.	
v2.2.21		
	CCMi_TIM_AUX_IN_SRC now the setting for 0x1 is properly setup	
	ADC interface remark, as only 32addresses exist, but 64 are specified. This is due to address reservation for future extensions.	
	MCSINTSTAT? due to very fuzzy description, some clarification and corrections were necessary. Now all signals mentioned can be found.	
v2.2.22		
	Correcting interrupt table, to proper naming	
v2.2.23		
	TC3xx GTM TIM Interrupt Bundling: Note added that according to MIG it is 8 interrupts per TIM module.	

Table 189 Revision History (cont'd)

Reference	Change to Previous Version	Comment
v2.2.24		
	(A)TOMi_CHx_CTRL.EXTTRIGOUT TRIGOUT corrected to TRIGOUT=0	
	ENDIS_CTRL/STAT text improved	
	MCSi_CHx_CTRL access rights corrected	
	DPLL_CTRL1.TSL bits do run from 0 to 3	

29 Capture/Compare Unit 6 (CCU6)

The CCU6 is a high-resolution 16-bit capture and compare unit with application-specific modes, mainly for AC drive control. Special operating modes support the control of Brushless DC-motors using Hall sensors or Back-EMF detection. Furthermore, block commutation and control mechanisms for multi-phase machines are supported. It also supports inputs to start several timers synchronously, an important feature in devices with several CCU6 kernels. The CCU6 module consists of two identical kernels CCU60 and CCU61.

29.1 Feature List

This section gives an overview over the different building blocks and their main features.

Timer 12 Block Features

- Three capture/compare channels, each channel can be used either as capture or as compare channel
- Supports generation of three-phase PWM (six outputs, individual signals for high-side and low-side switches)
- 16-bit resolution, maximum count frequency = peripheral clock
- Dead-time control for each channel to avoid short-circuits in the power stage
- Concurrent update of T12 registers
- Center-aligned and edge-aligned PWM can be generated
- Single-shot mode supported
- Start can be controlled by external events
- Capability of counting external events
- Many interrupt request sources
- Hysteresis-like control mode

Timer 13 Block Features

- One independent compare channel with one output
- 16-bit resolution, maximum count frequency = peripheral clock
- Concurrent update of T13 registers
- Can be synchronized to T12
- Interrupt generation at period-match and compare-match
- Single-shot mode supported
- Start can be controlled by external events
- Capability of counting external events

Additional Specific Functions

- Block commutation for Brushless DC-drives implemented
- Position detection via Hall-sensor pattern
- Noise filter supported for position input signals
- Automatic rotational speed measurement and commutation control for block commutation
- Integrated error handling
- Fast emergency stop without CPU load via external signal ($\overline{\text{CTRAP}}$)
- Control modes for multi-channel AC-drives
- Output levels can be selected and adapted to the power stage

29.2 Overview

The CCU6 is made up of a Timer T12 Block with three capture/compare channels and a Timer T13 Block with one compare channel. The T12 channels can independently generate PWM signals or accept capture triggers, or they can jointly generate control signal patterns to drive AC-motors or inverters.

This chapter is structured as follows:

29.2.1 Functional Overview

A rich set of status bits, synchronized updating of parameter values via shadow registers, and flexible generation of interrupt request signals provide means for efficient software-control.

The Timer T12 can operate in capture and/or compare mode for its three channels. The modes can also be combined (e.g. a channel operates in compare mode, whereas another channel operates in capture mode). The Timer T13 can operate in compare mode only. The multi-channel control unit generates output patterns which can be modulated by T12 and/or T13. The modulation sources can be selected and combined for the signal modulation.

Note: The capture/compare module itself is named CCU6 (capture/compare unit 6).
The first instance of a CCU6 module, including kernels CCU60 and CCU61, is named CCU6061.
A capture/compare channel inside this module is named CC6x.
For easier readability, the term “CCU6” is used to refer to a CCU6 module as well as to a CCU6 kernel, unless a distinct differentiation is required.

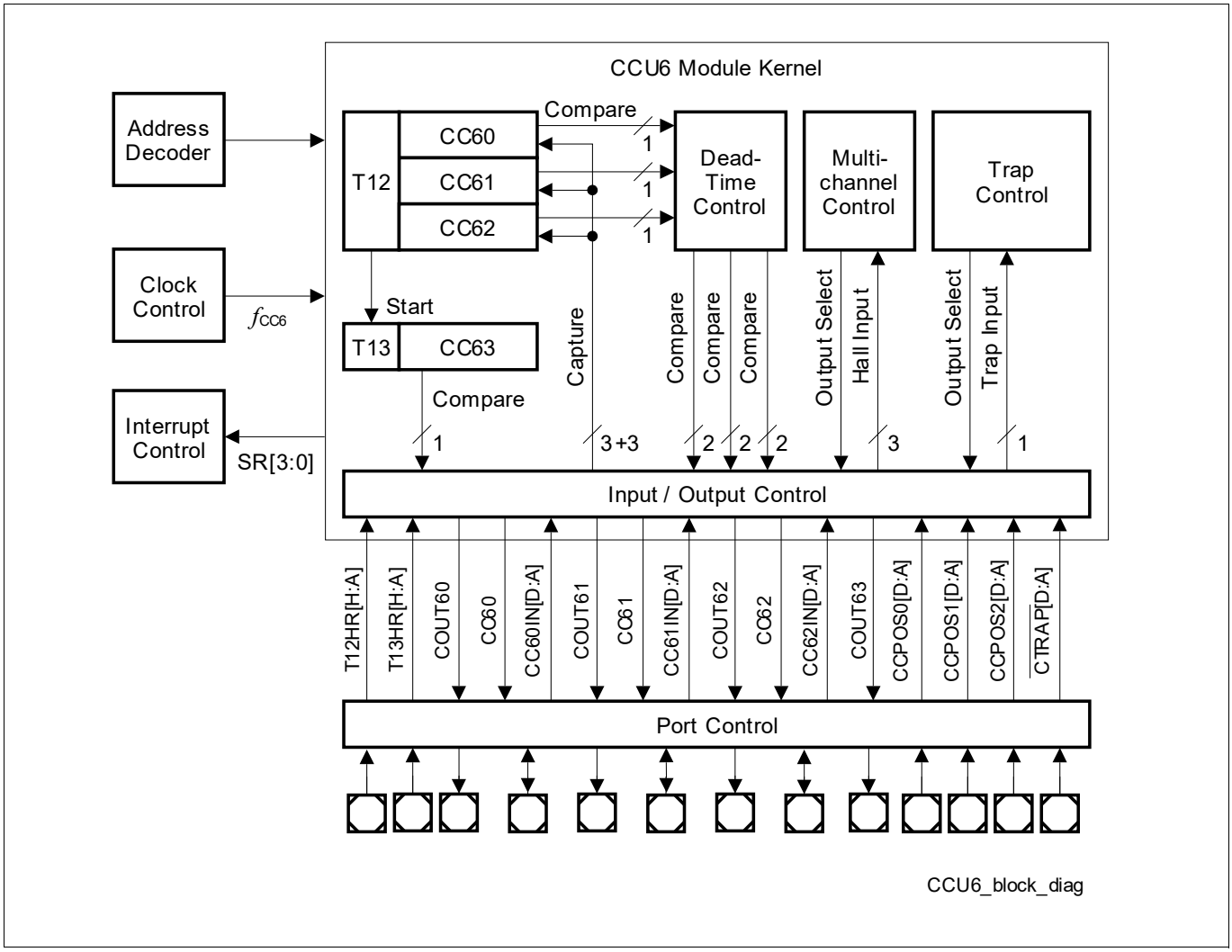


Figure 155 CCU6 Block Diagram

29.2.2 CCU6 Register Overview

Figure 156 shows the kernel registers of a CCU6 . BPI registers are described separately in **Section 29.11.6**, the Module Output Select register MOSEL is described in **Section 29.11.5**.

CCU6 Kernel Register Overview

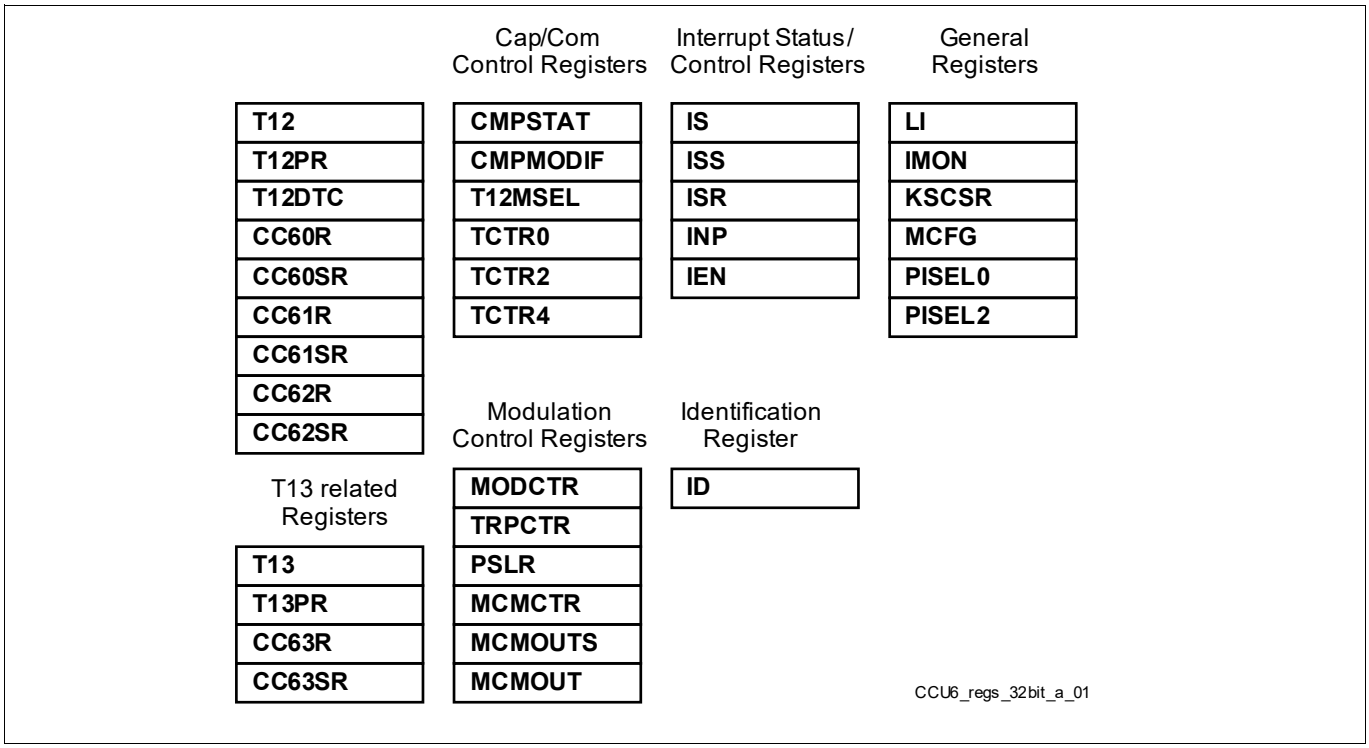


Figure 156 CCU6 Registers

Table 190 summarizes all registers required for programming of a CCU6 . It includes the CCU6 kernel registers, BPI and Module Output Select register, and defines their offset addresses and reset.

For the specific register table, the prefix “CCU6x_” has to be added to the register names in this table to identify the registers of different CCU6 kernels that are implemented. In this naming convention, x indicates the kernel number.

The CCU6 module has no destructive register read mechanisms.

In the case of a write access to addresses inside the address range (that is covered by the same chip select signal), but that are not the addresses explicitly mentioned for the module, the write access is not taken into account for the module. The same principle is valid for read accesses. In case of a read access to another address, the module does not react.

*Note: The exact register address is given by the relative address of the register (given in **Table 190**) plus the kernel base address (given in the appendix).*

Note: The Module Output Select Register MOSEL controls the trigger signals from both CCU6 kernels (CCU60 and CCU61) and is only available in the address space of kernel CCU60.

Table 190 Register Overview - CCU6 (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	0000_H	U,SV	SV,E,P	Application Reset	97
MCFG	Module Configuration Register	0004_H	U,SV	U,SV,P	See page 89	89
ID	Module Identification Register	0008_H	U,SV	BE	See page 87	87
MOSEL	CCU60 Module Output Select Register	$000C_H$	U,SV	U,SV,P	Application Reset	95
PISEL0	Port Input Select Register 0	0010_H	U,SV	U,SV,P	Application Reset	87
PISEL2	Port Input Select Register 2	0014_H	U,SV	U,SV,P	Application Reset	88
KSCSR	Kernel State Control Sensitivity Register	$001C_H$	U,SV	U,SV,P,OEN	See page 93	93
T12	Timer T12 Counter Register	0020_H	U,SV	U,SV,P	Application Reset	28
T12PR	Timer 12 Period Register	0024_H	U,SV	U,SV,P	Application Reset	28
T12DTC	Dead-Time Control Register for Timer12	0028_H	U,SV	U,SV,P	Application Reset	30
CC6xR	Capture/Compare Register for Channel CC6x	0030_H+x *4	U,SV	U,SV,P	Application Reset	29
CC6xSR	Capture/Compare Shadow Reg. for Channel CC6x	0040_H+x *4	U,SV	U,SV,P	Application Reset	29
T13	Timer T13 Counter Register	0050_H	U,SV	U,SV,P	Application Reset	51
T13PR	Timer 13 Period Register	0054_H	U,SV	U,SV,P	Application Reset	51
CC63R	Compare Register for T13	0058_H	U,SV	U,SV,P	Application Reset	52
CC63SR	Compare Shadow Register for T13	$005C_H$	U,SV	U,SV,P	Application Reset	52
CMPSTAT	Compare State Register	0060_H	U,SV	U,SV,P	Application Reset	32
CMPMODIF	Compare State Modification Register	0064_H	U,SV	U,SV,P	Application Reset	33
T12MSEL	T12 Mode Select Register	0068_H	U,SV	U,SV,P	Application Reset	35
TCTR0	Timer Control Register 0	0070_H	U,SV	U,SV,P	Application Reset	35

Table 190 Register Overview - CCU6 (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
TCTR2	Timer Control Register 2	0074_H	U,SV	U,SV,P	Application Reset	37
TCTR4	Timer Control Register 4	0078_H	U,SV	U,SV,P	Application Reset	39
MODCTR	Modulation Control Register	0080_H	U,SV	U,SV,P	Application Reset	64
TRPCTR	Trap Control Register	0084_H	U,SV	U,SV,P	Application Reset	65
PSLR	Passive State Level Register	0088_H	U,SV	U,SV,P	Application Reset	67
MCMOUTS	Multi-Channel Mode Output Shadow Register	$008C_H$	U,SV	U,SV,P	Application Reset	70
MCMOUT	Multi-Channel Mode Output Register	0090_H	U,SV	U,SV,P	Application Reset	71
MCMCTR	Multi-Channel Mode Control Register	0094_H	U,SV	U,SV,P	Application Reset	68
IMON	Input Monitoring Register	0098_H	U,SV	U,SV,P	Application Reset	90
LI	Lost Indicator Register	$009C_H$	U,SV	U,SV,P	Application Reset	92
IS	Interrupt Status Register	$00A0_H$	U,SV	U,SV,P	Application Reset	75
ISS	Interrupt Status Set Register	$00A4_H$	U,SV	U,SV,P	Application Reset	77
ISR	Interrupt Status Reset Register	$00A8_H$	U,SV	U,SV,P	Application Reset	78
INP	Interrupt Node Pointer Register	$00AC_H$	U,SV	U,SV,P	Application Reset	81
IEN	Interrupt Enable Register	$00B0_H$	U,SV	U,SV,P	Application Reset	79
OCS	OCDS Control and Status Register	$00E8_H$	U,SV	SV,P,OEN	See page 98	98
KRSTCLR	Kernel Reset Status Clear Register	$00EC_H$	U,SV	SV,E,P	Application Reset	102
KRST1	Kernel Reset Register 1	$00F0_H$	U,SV	SV,E,P	Application Reset	101
KRST0	Kernel Reset Register 0	$00F4_H$	U,SV	SV,E,P	Application Reset	100
ACCEN0	Access Enable Register 0	$00FC_H$	U,SV	SV,SE	Application Reset	99

29.3 Operating Timer T12

The timer T12 block is the main unit to generate the 3-phase PWM signals. A 16-bit counter is connected to 3 channel registers via comparators, that generate a signal when the counter contents match one of the channel register contents. A variety of control functions facilitate the adaptation of the T12 structure to different application needs.
Besides the 3-phase PWM generation, the T12 block offers options for individual compare and capture functions, as well as dead-time control and hysteresis-like compare mode.

This section provides information about:

- T12 overview (see **Section 29.3.1**)
- Counting scheme (see **Section 29.3.2**)
- Compare modes (see **Section 29.3.3**)
- Compare mode output path (see **Section 29.3.4**)
- Capture modes (see **Section 29.3.5**)
- Shadow transfer (see **Section 29.3.6**)
- T12 operating mode selection (see **Section 29.3.7**)
- T12 register description (see **Section 29.3.8**)

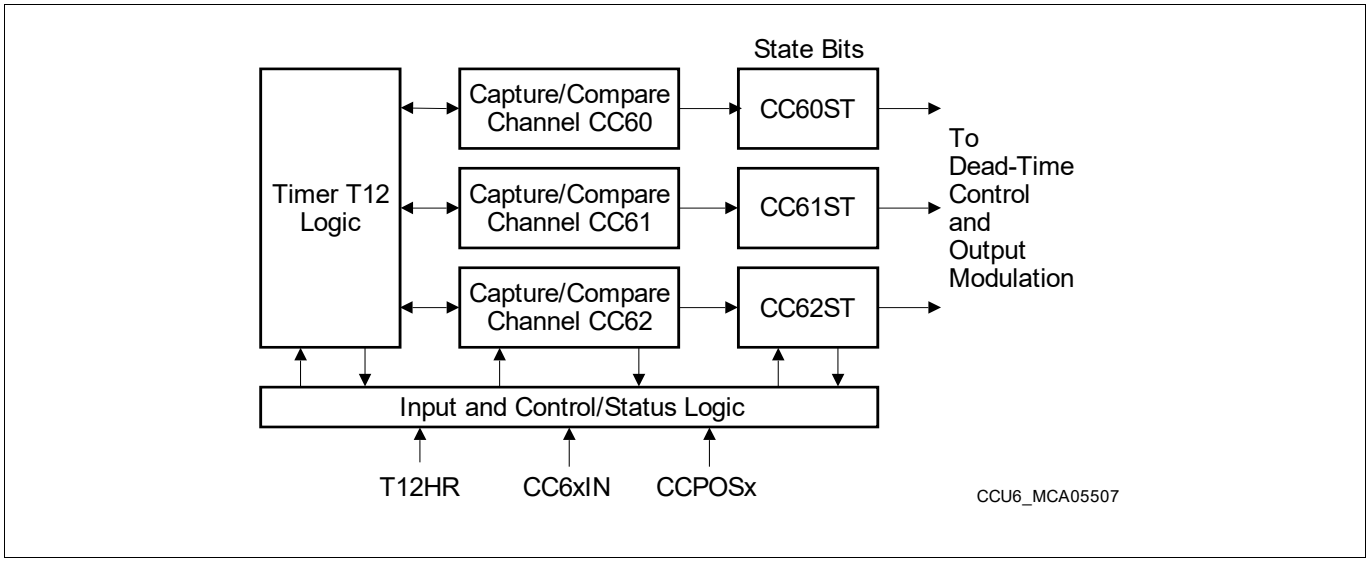


Figure 157 Overview Diagram of the Timer T12 Block

29.3.1 T12 Overview

Figure 158 shows a detailed block diagram of Timer T12. The functions of the timer T12 block are controlled by bits in registers **TCTR0**, **TCTR2**, and **PISEL0**.

Timer T12 receives its input clock (f_{T12}) from the module clock f_{CC6} via a programmable prescaler and an optional 1/256 divider or from an input signal T12HR. These options are controlled via bit fields T12CLK and T12PRE (see **Table 191**). T12 can count up or down, depending on the selected operation mode. A direction flag, CDIR, indicates the current counting direction.

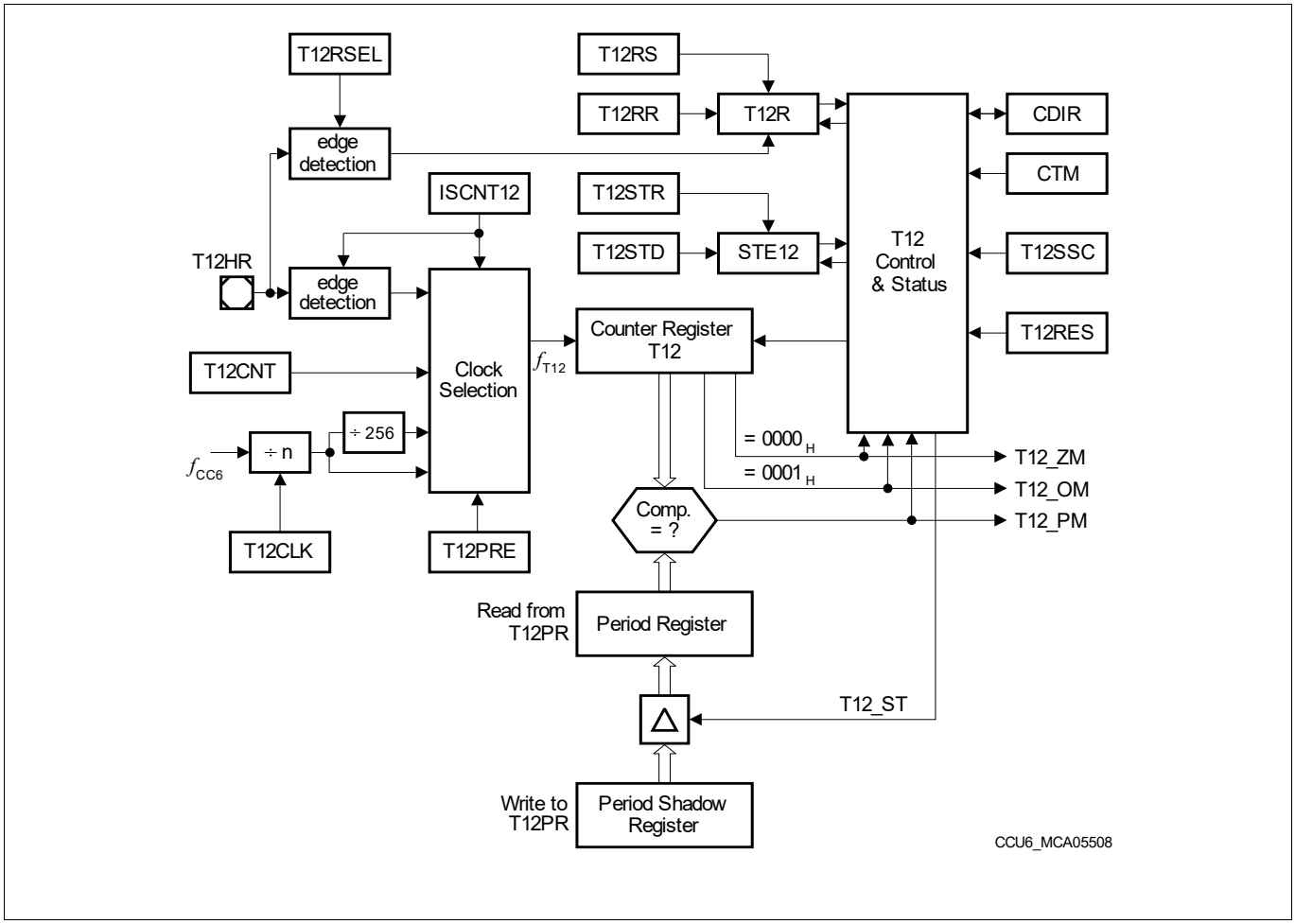


Figure 158 Timer T12 Logic and Period Comparators

Via a comparator, the T12 counter register **T12** is connected to a Period Register **T12PR**. This register determines the maximum count value for T12.

In Edge-Aligned mode, T12 is cleared to 0000_H after it has reached the period value defined by T12PR. In Center-Aligned mode, the count direction of T12 is set from ‘up’ to ‘down’ after it has reached the period value (please note that in this mode, T12 exceeds the period value by one before counting down). In both cases, signal T12_PM (T12 Period Match) is generated. The Period Register receives a new period value from its Shadow Period Register.

A read access to T12PR delivers the current period value at the comparator, whereas a write access targets the Shadow Period Register to prepare another period value. The transfer of a new period value from the Shadow Period Register into the Period Register (see **Section 29.3.6**) is controlled via the ‘T12 Shadow Transfer’ control signal, T12_ST. The generation of this signal depends on the operating mode and on the shadow transfer enable bit STE12. Providing a shadow register for the period value as well as for other values related to the generation of the PWM signal allows a concurrent update by software for all relevant parameters.

Two further signals indicate whether the counter contents are equal to 0000_H (T12_ZM = zero match) or 0001_H (T12_OM = one match). These signals control the counting and switching behavior of T12.

The basic operating mode of T12, either Edge-Aligned mode (**Figure 159**) or Center-Aligned mode (**Figure 160**), is selected via bit CTM. A Single-Shot control bit, T12SSC, enables an automatic stop of the timer when the current counting period is finished (see **Figure 161** and **Figure 162**).

The start or stop of T12 is controlled by the Run bit T12R that can be modified by bits in register **TCTR4**. The run bit can be set/cleared by software via the associated set/clear bits T12RS or T12RR, it can be set by a selectable edge of the input signal T12HR (**TCTR2**.T12RSEL), or it is cleared by hardware according to preselected conditions.
The timer T12 run bit T12R must not be set while the applied T12 period value is zero.
Timer T12 can be cleared via control bit T12RES. Setting this write-only bit does only clear the timer contents, but has no further effects, for example, it does not stop the timer.

The generation of the T12 shadow transfer control signal, T12_ST, is enabled via bit STE12. This bit can be set or reset by software indirectly through its associated set/clear control bits T12STR and T12STD.
While Timer T12 is running, write accesses to the count register T12 are not taken into account. If T12 is stopped and the Dead-Time counters are 0, write actions to register T12 are immediately taken into account.

29.3.2 T12 Counting Scheme

This section describes the clocking and counting capabilities of T12.

29.3.2.1 Clock Selection

In **Timer Mode** (**PISEL2**.ISCNT12 = 00_B), the input clock f_{T12} of Timer T12 is derived from the internal module clock f_{CC6} through a programmable prescaler and an optional 1/256 divider. The resulting prescaler factors are listed in **Table 191**. The prescaler of T12 is cleared while T12 is not running (**TCTR0**.T12R = 0) to ensure reproducible timings and delays.

Table 191 Timer T12 Input Frequency Options

T12CLK	Resulting Input Clock *f*T12 Prescaler Off (T12PRE = 0)	Resulting Input Clock *f*T12 Prescaler On (T12PRE = 1)
000_B	f_{CC6}	f_{CC6} / 256
001_B	f_{CC6} / 2	f_{CC6} / 512
010_B	f_{CC6} / 4	f_{CC6} / 1024
011_B	f_{CC6} / 8	f_{CC6} / 2048
100_B	f_{CC6} / 16	f_{CC6} / 4096
101_B	f_{CC6} / 32	f_{CC6} / 8192
110_B	f_{CC6} / 64	f_{CC6} / 16384
111_B	f_{CC6} / 128	f_{CC6} / 32768

In **Counter Mode**, timer T12 counts one step:

- If a 1 is written to **TCTR4**.T12CNT and **PISEL2**.ISCNT12 = 01_B
- If a rising edge of input signal T12HR is detected and **PISEL2**.ISCNT12 = 10_B
- If a falling edge of input signal T12HR is detected and **PISEL2**.ISCNT12 = 11_B

29.3.2.2 Edge-Aligned / Center-Aligned Mode

In **Edge-Aligned Mode** (CTM = 0), timer T12 is always counting upwards (CDIR = 0). When reaching the value given by the period register (period-match T12_PM), the value of T12 is cleared with the next counting step (saw tooth shape).

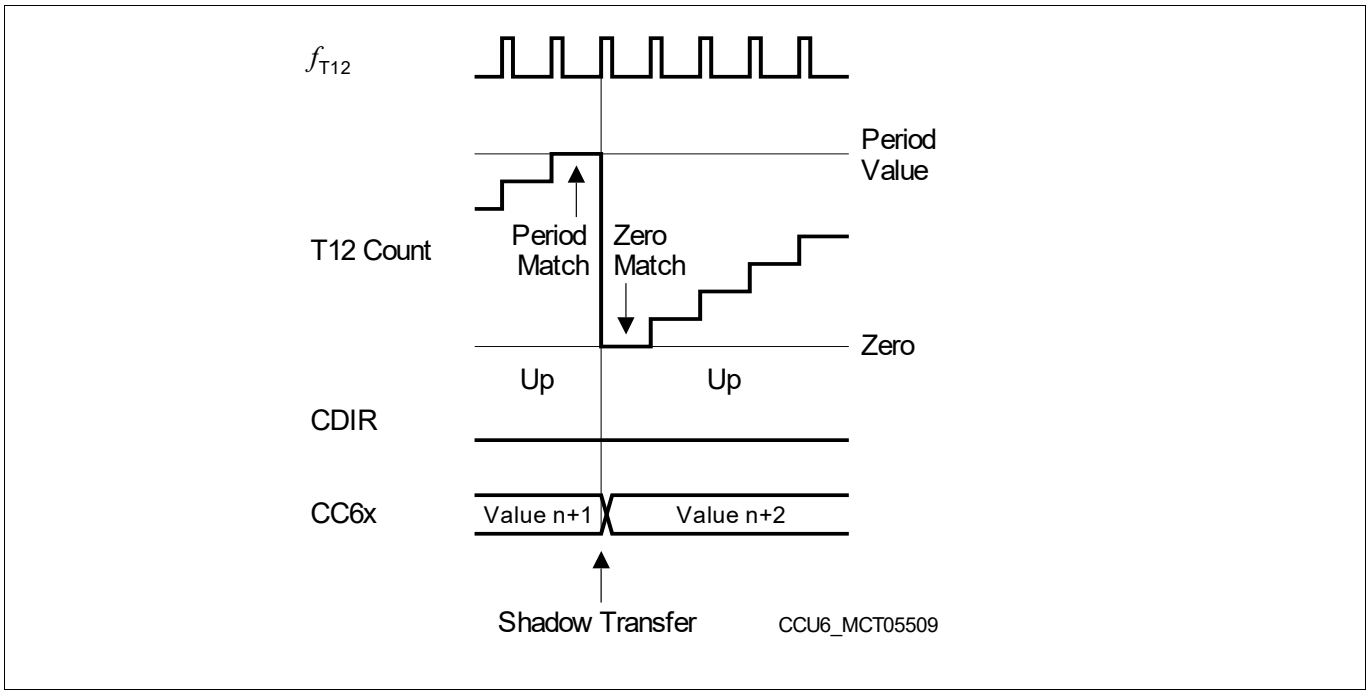


Figure 159 T12 Operation in Edge-Aligned Mode

As a result, in Edge-Aligned mode, the timer period is given by:

$T12_{PER}$ = <Period-Value> + 1; in T12 clocks (f_{T12}) (29.1)

In **Center-Aligned Mode** (CTM = 1), timer T12 is counting upwards or downwards (triangular shape). When reaching the value given by the period register (period-match T12_PM) while counting upwards (CDIR = 0), the counting direction control bit CDIR is changed to downwards (CDIR = 1) with the next counting step.
When reaching the value 0001_H (one-match T12_OM) while counting downwards, the counting direction control bit CDIR is changed to upwards with the next counting step.

As a result, in Center.Aligned mode, the timer period is given by:

$T12_{PER}$ = (<Period-Value> + 1) x 2; in T12 clocks (f_{T12}) (29.2)

- With the next clock event of f_{T12} the count direction is set to counting up (CDIR = 0) when the counter reaches 0001_H while counting down.
- With the next clock event of f_{T12} the count direction is set to counting down (CDIR = 1) when the Period-Match is detected while counting up.
- With the next clock event of f_{T12} the counter counts up while CDIR = 0 and it counts down while CDIR = 1.

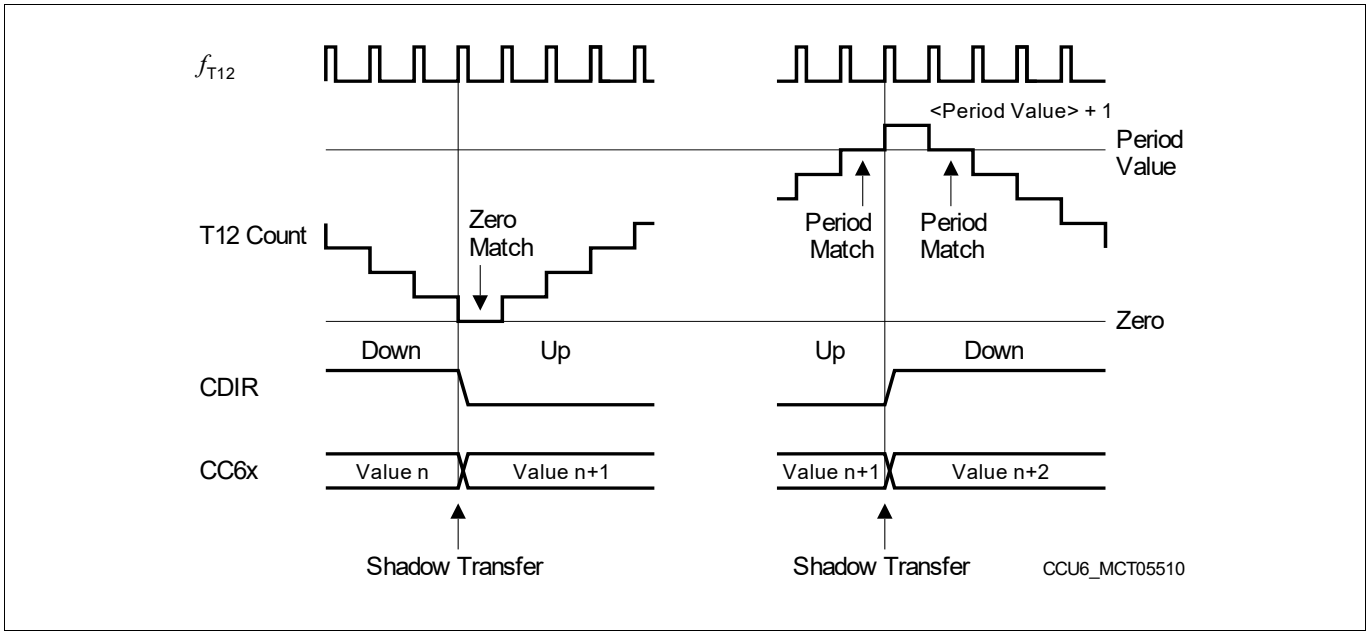


Figure 160 T12 Operation in Center-Aligned Mode

*Note: Bit CDIR changes with the next timer clock event after the one-match or the period-match. Therefore, the timer continues counting in the previous direction for one cycle before actually changing its direction (see **Figure 160**).*

29.3.2.3 Single-Shot Mode

In Single-Shot Mode, the timer run bit T12R is cleared by hardware. If bit T12SSC = 1, the timer T12 will stop when the current timer period is finished.

In Edge-Aligned mode, T12R is cleared when the timer becomes zero after having reached the period value (see **Figure 161**).

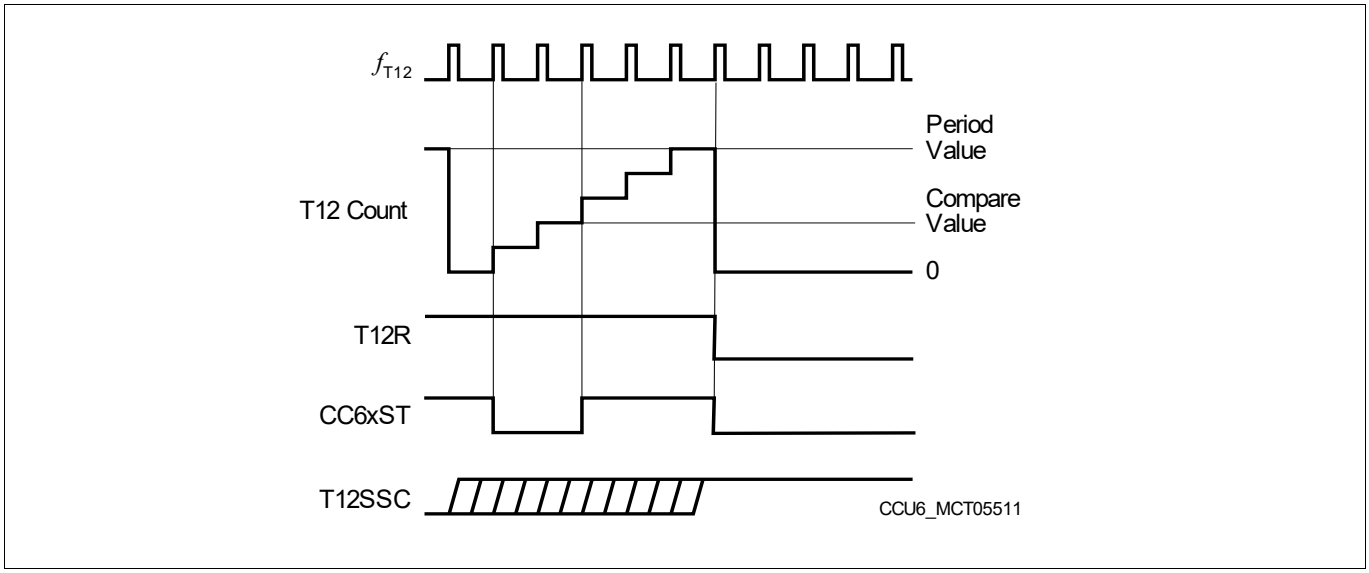


Figure 161 Single-Shot Operation in Edge-Aligned Mode

In Center-Aligned mode, the period is finished when the timer has counted down to zero (one clock cycle after the one-match while counting down, see **Figure 162**).

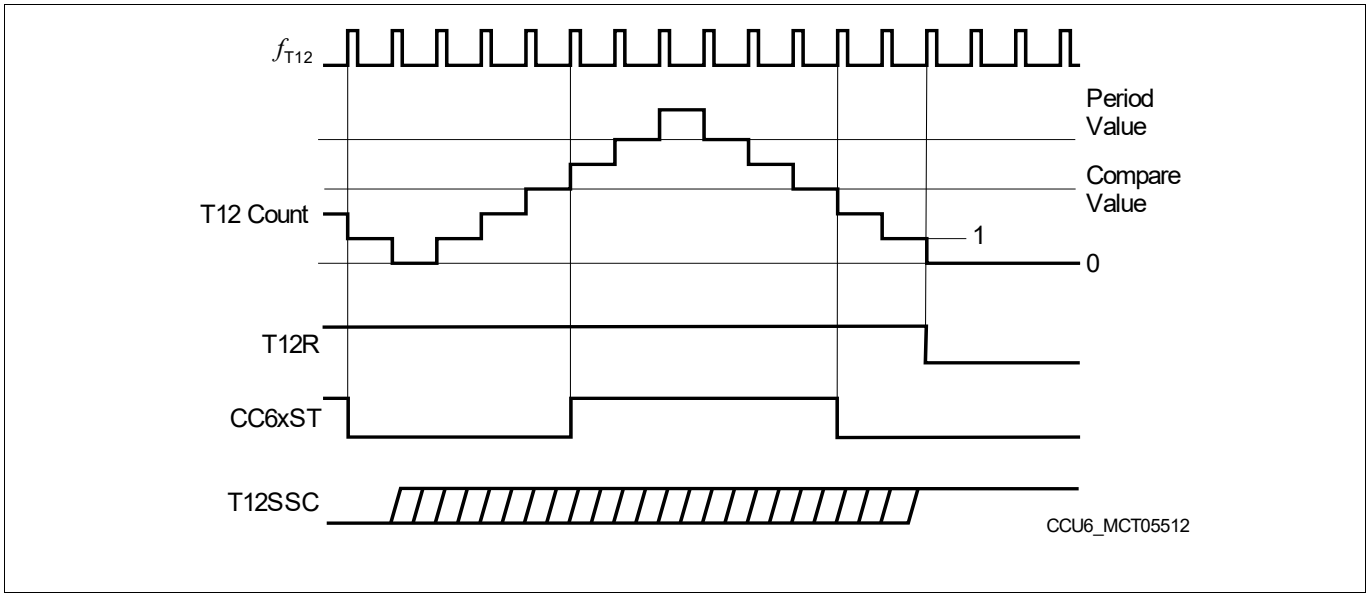


Figure 162 Single-Shot Operation in Center-Aligned Mode

29.3.3 T12 Compare Mode

Associated with Timer T12 are three individual capture/compare channels, that can perform compare or capture operations with regard to the contents of the T12 counter. The capture functions are explained in **Section 29.3.5**.

29.3.3.1 Compare Channels

In Compare Mode (see **Figure 163**), the three individual compare channels CC60 CC61, and CC62 can generate a three-phase PWM pattern.

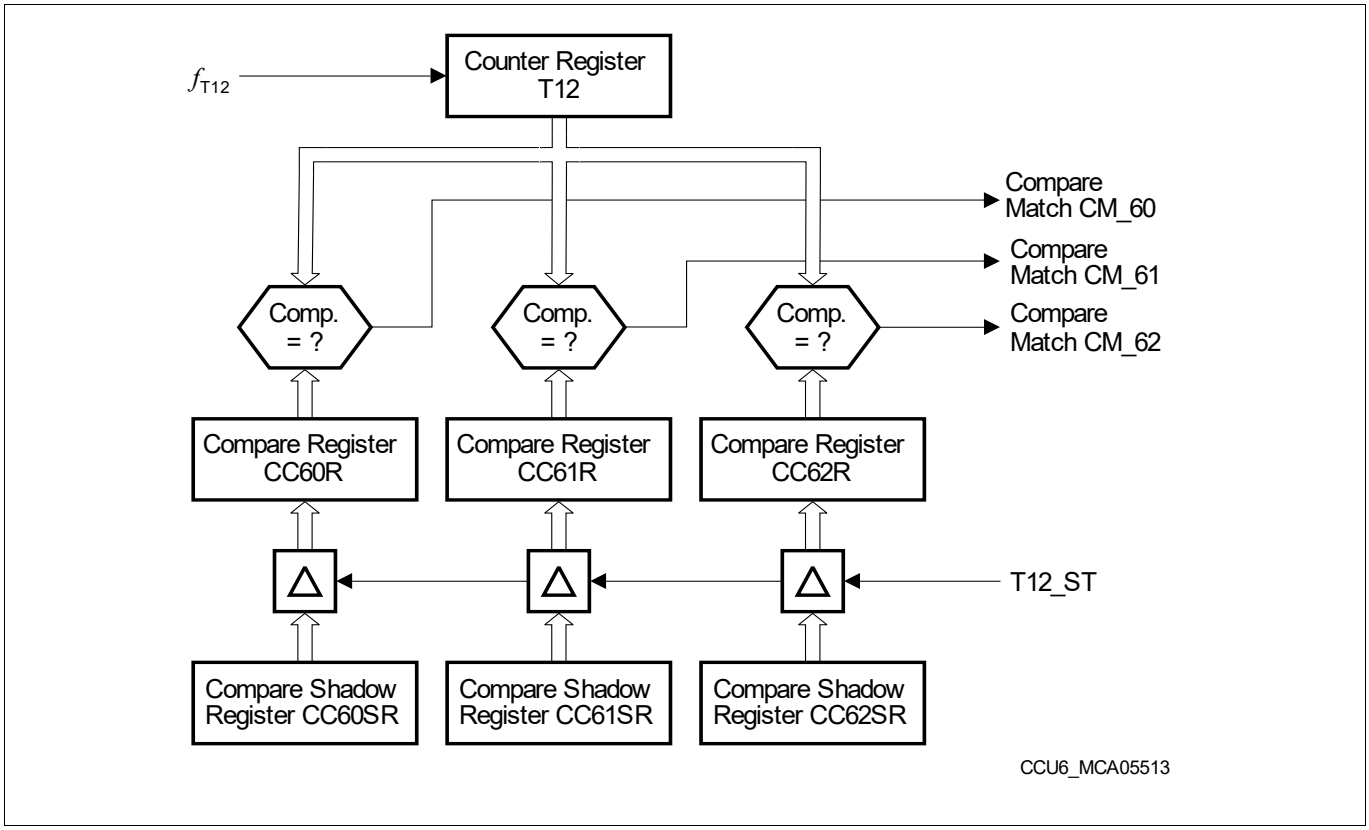


Figure 163 T12 Channel Comparators

Each compare channel is connected to the T12 counter register via its individual equal-to comparator, generating a match signal when the contents of the counter matches the contents of the associated compare register. Each channel consists of the comparator and a double register structure - the actual compare register CC6xR, feeding the comparator, and an associated shadow register CC6xSR, that is preloaded by software and transferred into the compare register when signal T12 shadow transfer, T12_ST, gets active. Providing a shadow register for the compare value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters of a three-phase PWM.

29.3.3.2 Channel State Bits

Associated with each (compare) channel is a State Bit, **CMPSTAT**.CC6xST, holding the status of the compare (or capture) operation (see **Figure 164**). In compare mode, the State Bits are modified according to a set of switching rules, depending on the current status of timer T12.

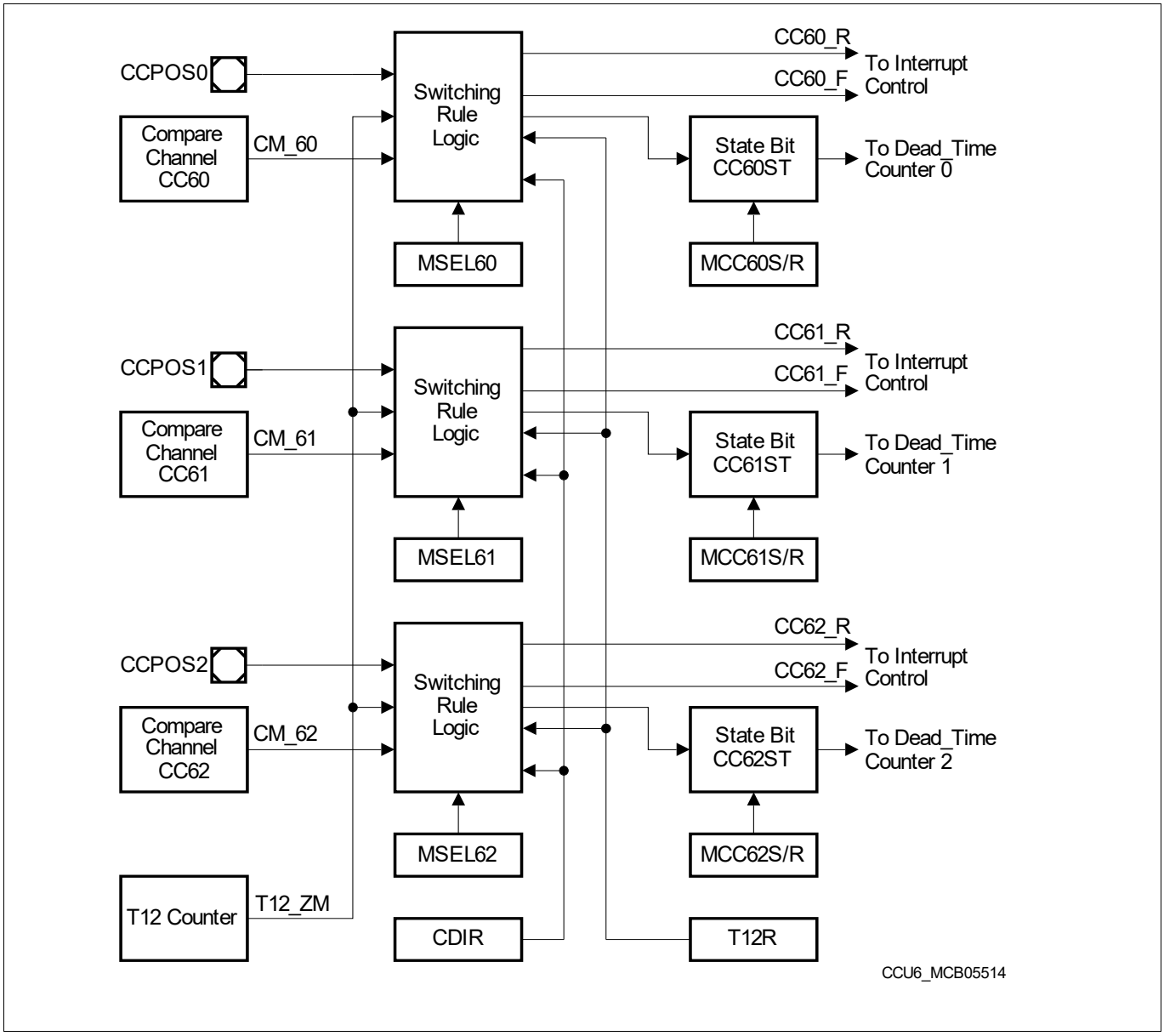


Figure 164 Compare State Bits for Compare Mode

The inputs to the switching rule logic for the CC6xST bits are the timer direction (CDIR), the timer run bit (T12R), the timer T12 zero-match signal (T12_ZM), and the actual individual compare-match signals CM_6x as well as the mode control bits, **T12MSEL**.MSEL6x.
In addition, each state bit can be set or cleared by software via the appropriate set and reset bits in register **CMPMODIF**, MCC6xS and MCC6xR. The input signals CCPOSx are used in hysteresis-like compare mode, whereas in normal compare mode, these inputs are ignored.

Note: In Hall Sensor, single shot or capture modes, additional/different rules are taken into account (see related sections).

A compare interrupt event CC6x_R is signaled when a compare match is detected while counting upwards, whereas the compare interrupt event CC6x_F is signaled when a compare match is detected while counting down. The actual setting of a State Bit has no influence on the interrupt generation in compare mode.

A modification of a State Bit CC6xST by the switching rule logic due to a compare action is only possible while Timer T12 is running (T12R = 1). If this is the case, the following switching rules apply for setting and clearing the State Bits in Compare Mode (illustrated in **Figure 165** and **Figure 166**):

A State Bit **CC6xST is set** to 1:

- with the next T12 clock (f_{T12}) after a compare-match when T12 is counting up (i.e., when the counter is incremented above the compare value);
- with the next T12 clock (f_{T12}) after a zero-match AND a parallel compare-match when T12 is counting up.

A State Bit **CC6xST is cleared** to 0:

- with the next T12 clock (f_{T12}) after a compare-match when T12 is counting down (i.e., when the counter is decremented below the compare value in center-aligned mode);
- with the next T12 clock (f_{T12}) after a zero-match AND NO parallel compare-match when T12 is counting up.

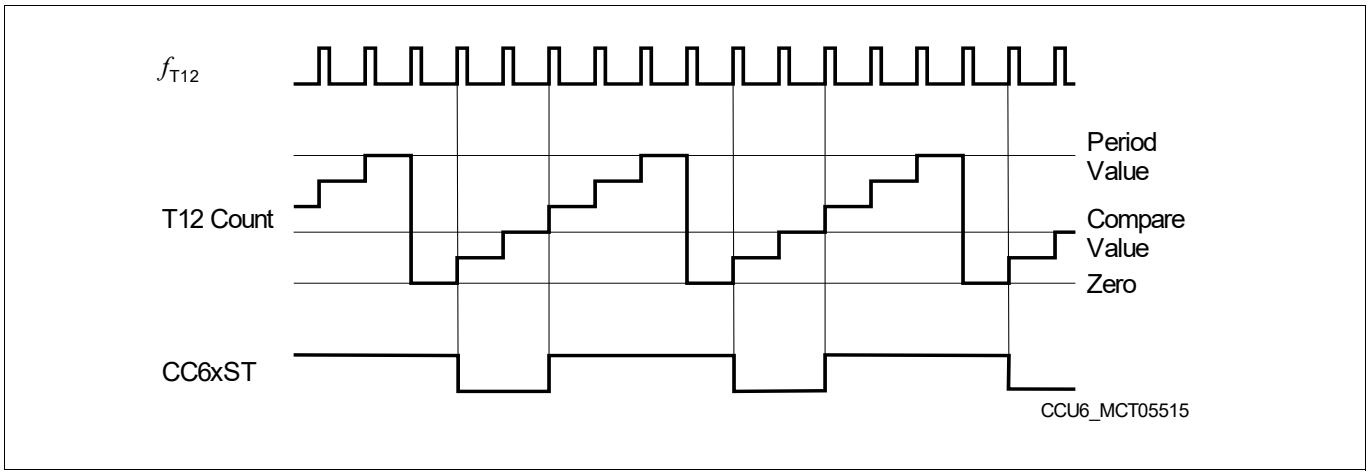


Figure 165 Compare Operation, Edge-Aligned Mode

Figure 167 illustrates some more examples for compare waveforms. It is important to note that in these examples, it is assumed that some of the compare values are changed while the timer is running. This change is performed via a software preload of the Shadow Register, CC6xSR. The value is transferred to the actual Compare Register CC6xR with the T12 Shadow Transfer signal, T12_ST, that is assumed to be enabled.

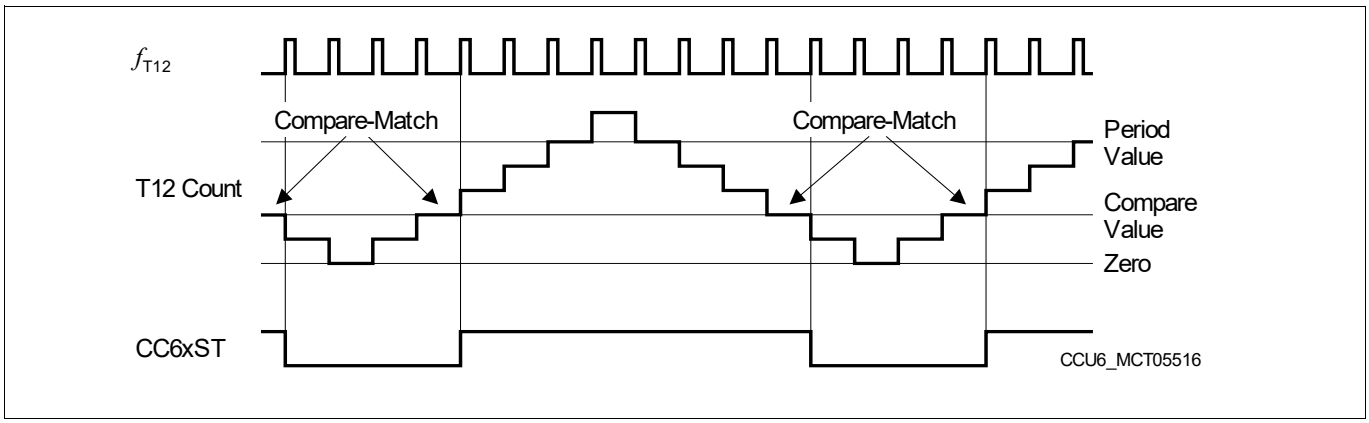


Figure 166 Compare Operation, Center-Aligned Mode

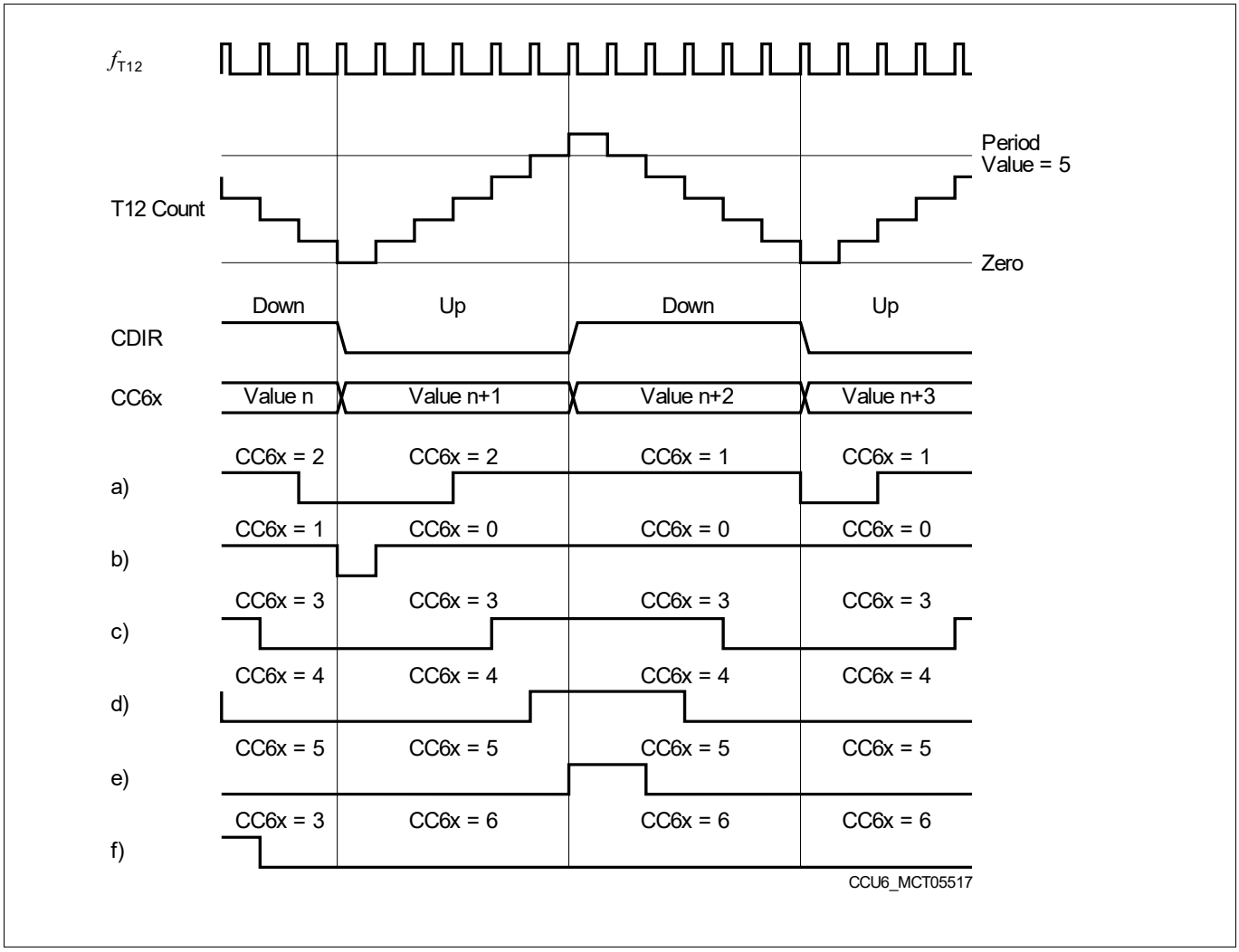


Figure 167 Compare Waveform Examples

Example b) illustrates the transition to a duty cycle of 100%. First, a compare value of 0001_H is used, then changed to 0000_H. Please note that a low pulse with the length of one T12 clock is still produced in the cycle where the new value 0000_H is in effect; this pulse originates from the previous value 0001_H. In the following timer cycles, the State Bit CC6xST remains at 1, producing a 100% duty cycle signal. In this case, the compare rule ‘zero-match AND compare-match’ is in effect.

Example f) shows the transition to a duty cycle of 0%. The new compare value is set to <Period-Value> + 1, and the State Bit CC6ST remains cleared.

Figure 168 illustrates an example for the waveforms of all three channels. With the appropriate dead-time control and output modulation, a very efficient 3-phase PWM signal can be generated.

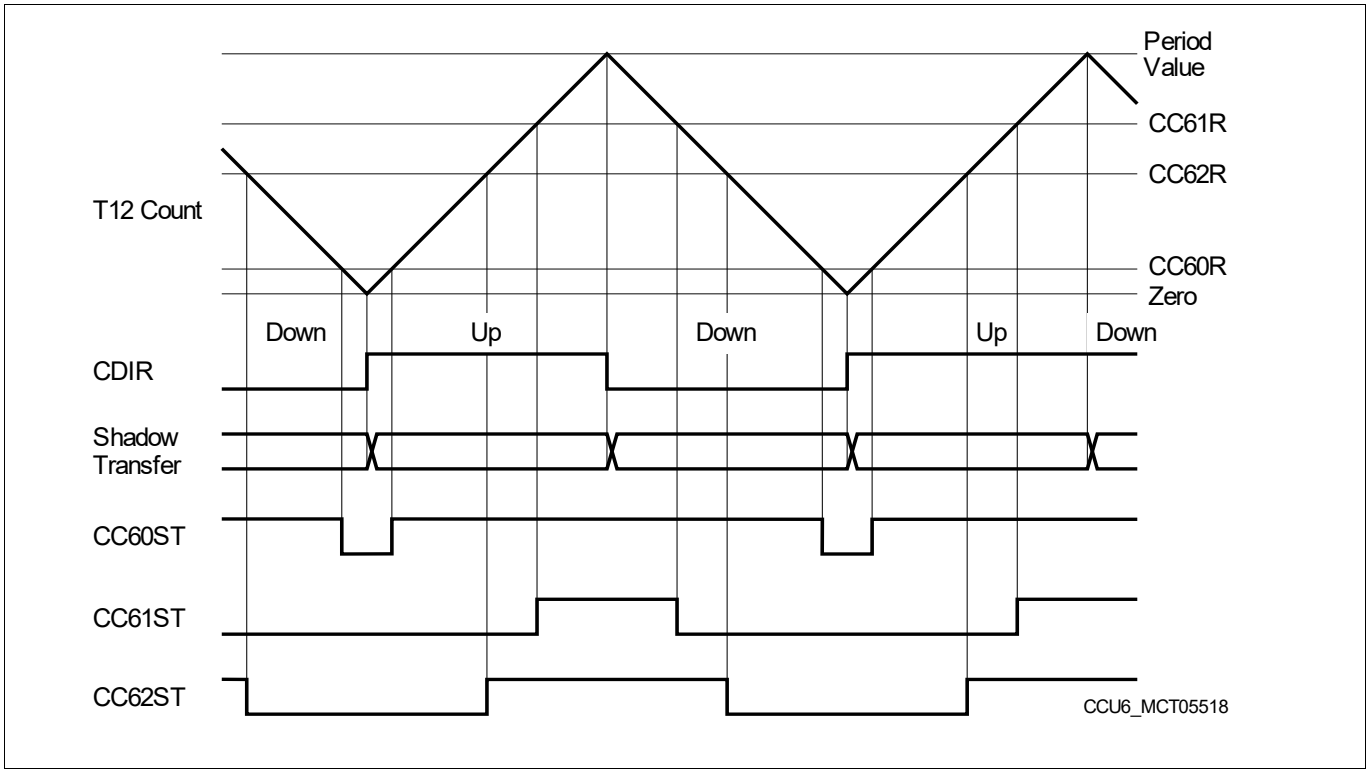


Figure 168 Three-Channel Compare Waveforms

29.3.3.3 Hysteresis-Like Control Mode

The hysteresis-like control mode (**T12MSEL**.MSEL6x = 1001_B) offers the possibility to switch off the PWM output if the input CCPOSx becomes 0 by clearing the State Bit CC6xST. This can be used as a simple motor control feature by using a comparator indicating, e.g., overcurrent. While CCPOSx = 0, the PWM outputs of the corresponding channel are driving their passive levels, because the setting of bit CC6xST is only possible while CCPOSx = 1.

As long as input CCPOSx is 0, the corresponding State Bit is held 0. When CCPOSx is at high level, the outputs can be in active state and are determined by bit CC6xST (see **Figure 164** for the state bit logic and **Figure 169** for the output paths).

The CCPOSx inputs are evaluated with f_{CC6}.

This mode can be used to introduce a timing-related behavior to a hysteresis controller. A standard hysteresis controller detects if a value exceeds a limit and switches its output according to the compare result. Depending on the operating conditions, the switching frequency and the duty cycle are not fixed, but change permanently.

If (outer) time-related control loops based on a hysteresis controller in an inner loop should be implemented, the outer loops show a better behavior if they are synchronized to the inner loops. Therefore, the hysteresis-like mode can be used, that combines timer-related switching with a hysteresis controller behavior. For example, in this mode, an output can be switched on according to a fixed time base, but it is switched off as soon as a falling edge is detected at input CCPOSx.

This mode can also be used for standard PWM with overcurrent protection. As long as there is no low level signal at pin CCPOSx, the output signals are generated in the normal manner as described in the previous sections. Only if input CCPOSx shows a low level, e.g. due to the detection of overcurrent, the outputs are shut off to avoid harmful stress to the system.

29.3.4 Compare Mode Output Path

Figure 169 gives an overview on the signal path from a channel State Bit to its output pin in its simplest form. As illustrated, a user has a variety of controls to determine the desired output signal switching behavior in relation to the current state of the State Bit, CC6xST. Please refer to **Section 29.3.4.3** for details on the output modulation.

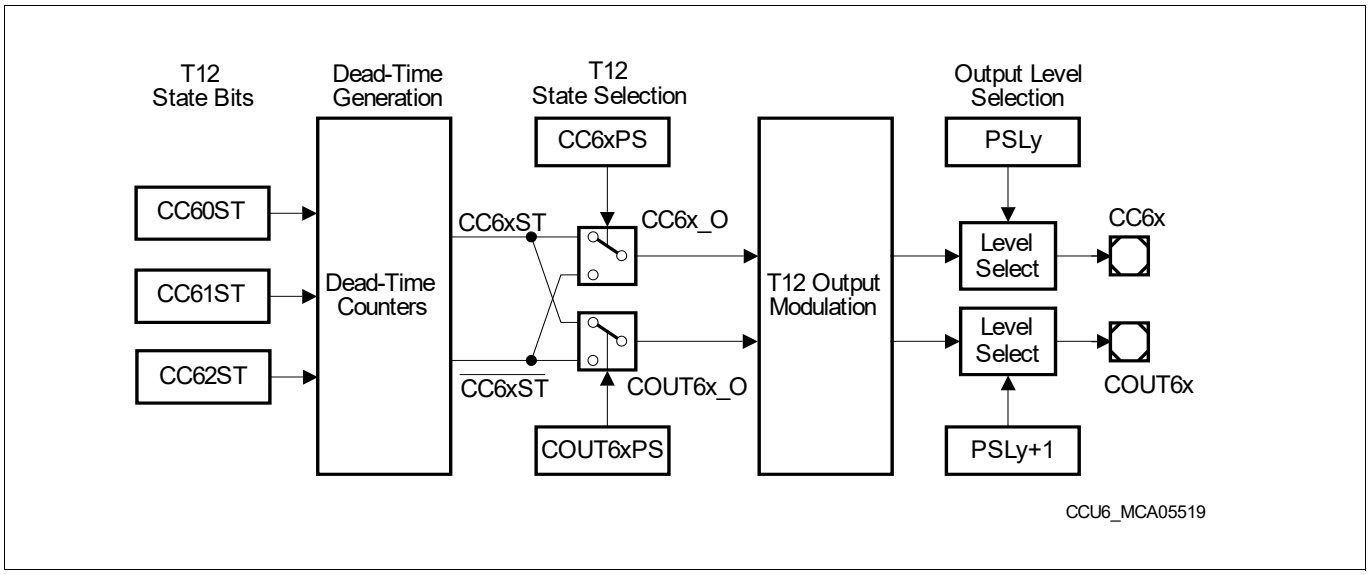


Figure 169 Compare Mode Simplified Output Path Diagram

The output path is based on signals that are defined as active or passive. The terms active and passive are not related to output levels, but to internal actions. This mainly applies for the modulation, where T12 and T13 signals are combined with the multi-channel signals and the trap function. The Output level Selection allows the user to define the output level at the output pin for the passive state (inverted level for the active state). It is recommended to configure this block in a way that an external power switch is switched off while the CCU6 delivers an output signal in the passive state.

29.3.4.1 Dead-Time Generation

The generation of (complementary) signals for the high-side and the low-side switches of one power inverter phase is based on the same compare channel. For example, if the high-side switch should be active while the T12 counter value is above the compare value (State Bit = 1), then the low-side switch should be active while the counter value is below the compare value (State Bit = 0).
In most cases, the switching behavior of the connected power switches is not symmetrical concerning the switch-on and switch-off times. A general problem arises if the time for switch-on is smaller than the time for switch-off of the power device. In this case, a short-circuit can occur in the inverter bridge leg, which may damage the complete system. In order to solve this problem by HW, this capture/compare unit contains a programmable Dead-Time Generation Block, that delays the passive to active edge of the switching signals by a programmable time (the active to passive edge is not delayed).

The Dead-Time Generation Block, illustrated in **Figure 170**, is built in a similar way for all three channels of T12. It is controlled by bits in register **T12DTC**. Any change of a CC6xST State Bit activates the corresponding Dead-Time Counter, that is clocked with the same input clock as T12 (f_{T12}). The length of the dead-time can be programmed by bit field DTM. This value is identical for all three channels. Writing **TCTR4**.DTRES = 1 sets all dead-times to passive.

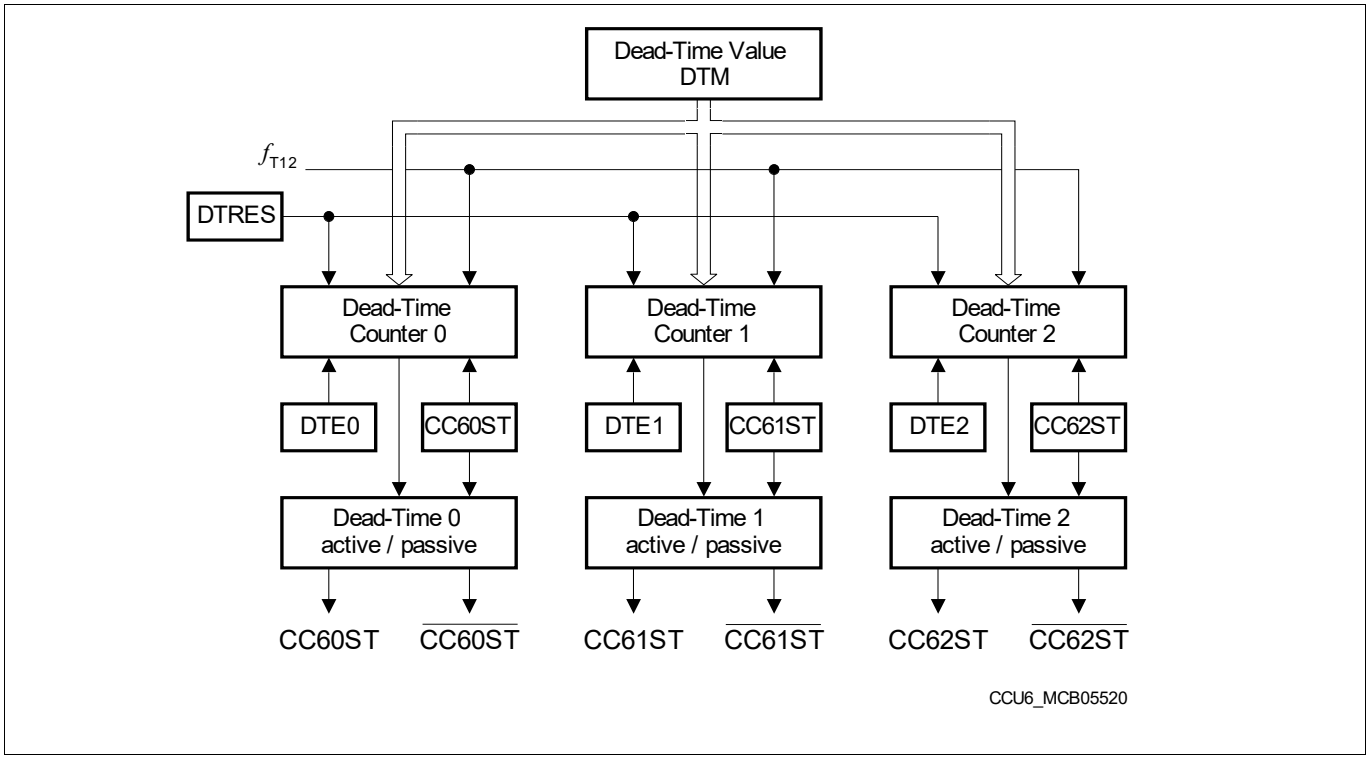


Figure 170 Dead-Time Generation Block Diagram

Each of the three dead-time counters has its individual dead-time enable bit, DTEx. An enabled dead-time counter generates a dead-time delaying the passive-to-active edge of the channel output signal. The change in a State Bit CC6xST is not taken into account while the dead-time generation of this channel is currently in progress (active). This avoids an unintentional additional dead-time if a State Bit CC6xST changes too early.
A disabled dead-time counter is always considered as passive and does not delay any edge of CC6xST.
Based on the State Bits CC6xST, the Dead-Time Generation Block outputs a direct signal CC6xST and an inverted signal $\overline{CC6xST}$ for each compare channel, each masked with the effect of the related Dead-Time Counters (waveforms illustrated in **Figure 171**).

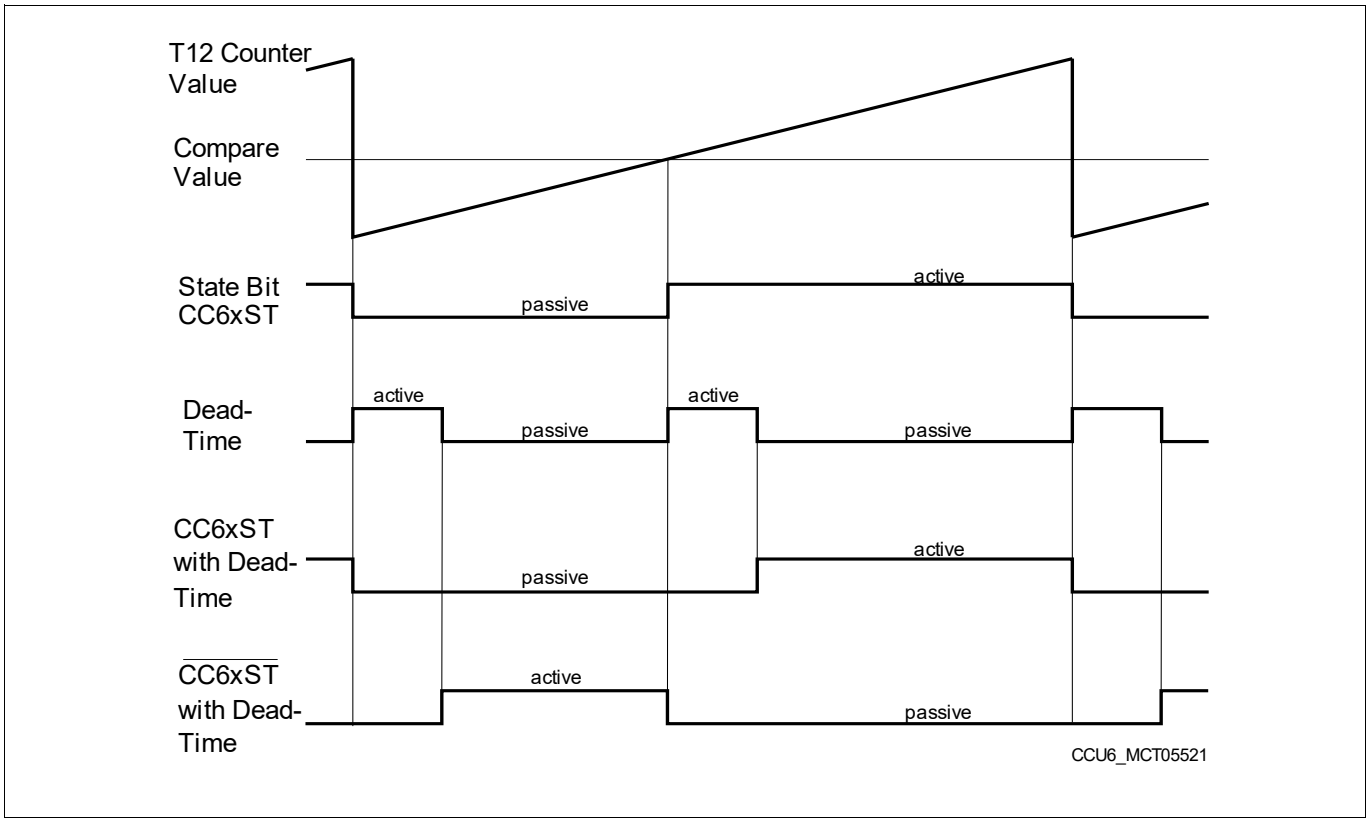


Figure 171 Dead-Time Generation Waveforms

29.3.4.2 State Selection

To support a wide range of power switches and drivers, the state selection offers the flexibility to define when an output can be active and can be modulated, especially useful for **complementary or multi-phase PWM** signals. The state selection is based on the signals CC6xST and $\overline{\text{CC6xST}}$ delivered by the dead-time generator (see **Figure 169**). Both signals are never active at the same time, but can be passive at the same time. This happens during the dead-time of each compare channel after a change of the corresponding State Bit CC6xST.
The user can select independently for each output signal CC6xO and COUT6xO if it should be active before or after the compare value has been reached (see register **CMPSTAT**). With this selection, the active (conducting) phases of complementary power switches in a power inverter bridge leg can be positioned with respect to the compare value (e.g. signal CC6xO can be active before, whereas COUT6xO can be active after the compare value is reached). Like this, the output modulation, the trap logic and the output level selection can be programmed independently for each output signal, although two output signals are referring to the same compare channel.

29.3.4.3 Output Modulation and Level Selection

The last block of the data path is the Output Modulation block. Here, all the modulation sources and the trap functionality are combined and control the actual level of the output pins (controlled by the modulation enable bits T1xMODENy and MCMEN in register **MODCTR**). The following signal sources can be combined here **for each T12 output signal** (see **Figure 172** for compare channel CC60):

- A **T12 related compare signal** CC6x_O (for outputs CC6x) or COUT6x_O (for outputs COUT6x) delivered by the T12 block (state selection with dead-time) with an individual enable bit T12MODENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)
- The **T13 related compare signal** CC63_O delivered by the T13 state selection with an individual enable bit T13MODENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)
- A **multi-channel output signal** MCMPy (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x) with a common enable bit MCMEN
- The **trap state** TRPS with an individual enable bit TRPENy per output signal (y = 0, 2, 4 for outputs CC6x and y = 1, 3, 5 for outputs COUT6x)

If one of the modulation input signals CC6x_O/COUT6x_O, CC63_O, or MCMPy of an output modulation block is enabled and is at passive state, the modulated is also in passive state, regardless of the state of the other signals that are enabled. Only if all enabled signals are in active state the modulated output shows an active state. If no modulation input is enabled, the output is in passive state.
If the Trap State is active (TRPS = 1), then the outputs that are enabled for the trap signal (by TRPENy = 1) are set to the passive state.

The output of each of the modulation control blocks is connected to a level select block that is configured by register **PSLR**. It offers the option to determine the actual output level of a pin, depending on the state of the output line (decoupling of active/passive state and output polarity) as specified by the Passive State Select bit PSLy. If the modulated output signal is in the passive state, the level specified directly by PSLy is output. If it is in the active state, the inverted level of PSLy is output. This allows the user to adapt the polarity of an active output signal to the connected circuitry.

The PSLy bits have shadow registers to allow for updates without undesired pulses on the output lines. The bits related to CC6x and COUT6x (x = 0, 1, 2) are updated with the T12 shadow transfer signal (T12_ST). A read action returns the actually used values, whereas a write action targets the shadow bits. Providing a shadow register for the PSL value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

Figure 172 shows the output modulation structure for compare channel CC60 (output signals CC60 and COUT60). A similar structure is implemented for the other two compare channels CC61 and CC62.

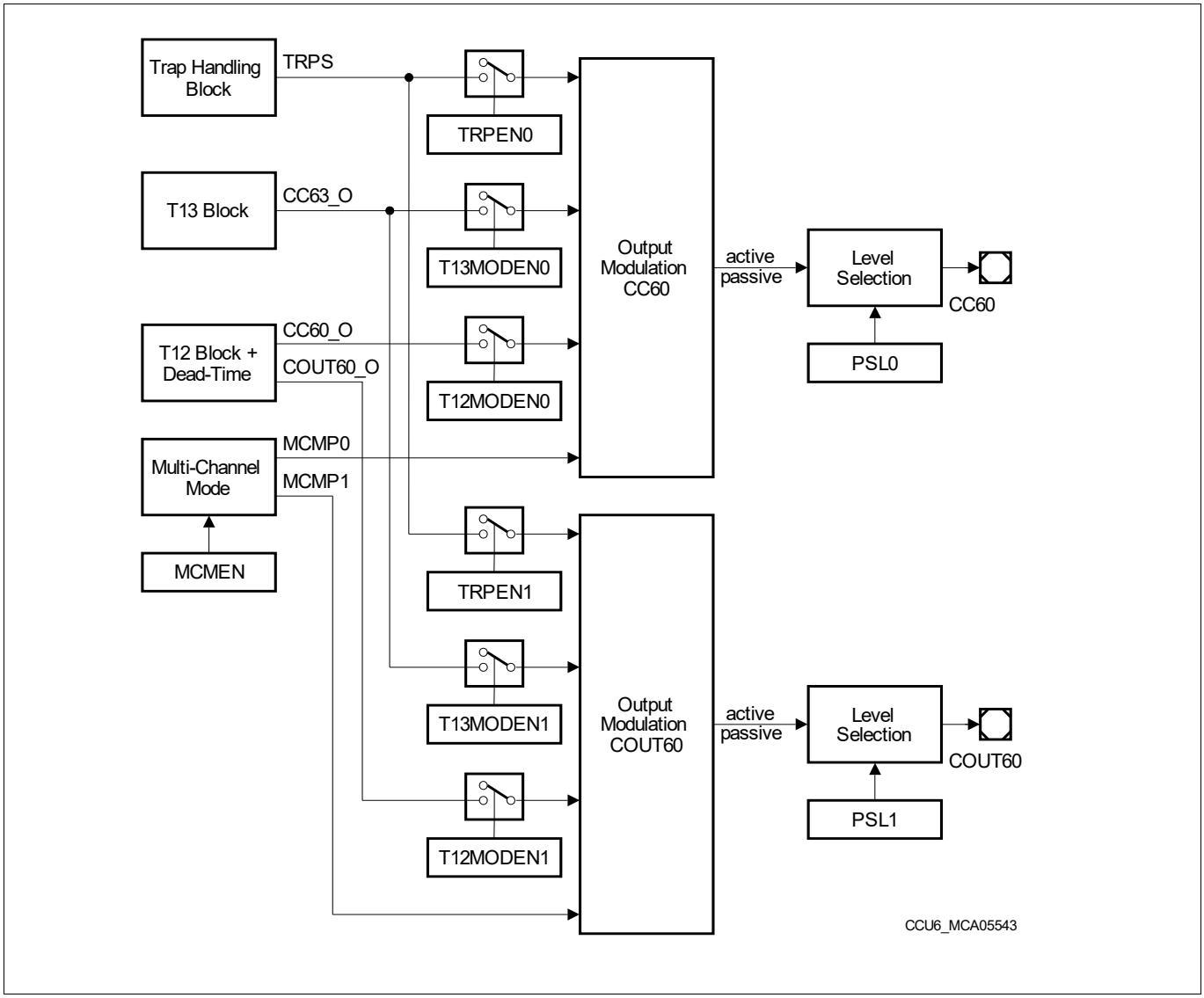


Figure 172 Output Modulation for Compare Channel CC60

29.3.5 T12 Capture Modes

Each of the three channels of the T12 Block can also be used to capture T12 time information in response to an external signal CC6xIN.
In capture mode, the interrupt event CC6x_R is detected when a rising edge is detected at the input CC6xIN, whereas the interrupt event CC6x_F is detected when a falling edge is detected.

There are a number of different modes for capture operation. In all modes, both of the registers of a channel are used. The selection of the capture modes is done via the **T12MSEL**.MSEL6x bit fields and can be selected individually for each of the channels.

Table 192 Capture Modes Overview

MSEL6x	Mode	Signal	Active Edge	CC6xSR Stored in	T12 Stored in
0100_B	1	CC6xIN	Rising	–	CC6xR
		CC6xIN	Falling	–	CC6xSR
0101_B	2	CC6xIN	Rising	CC6xR	CC6xSR
0110_B	3	CC6xIN	Falling	CC6xR	CC6xSR
0111_B	4	CC6xIN	Any	CC6xR	CC6xSR

Figure 173 illustrates **Capture Mode 1**. When a rising edge (0-to-1 transition) is detected at the corresponding input signal CC6xIN, the current contents of Timer T12 are captured into register CC6xR. When a falling edge (1-to-0 transition) is detected at the input signal CC6xIN, the contents of Timer T12 are captured into register CC6xSR.

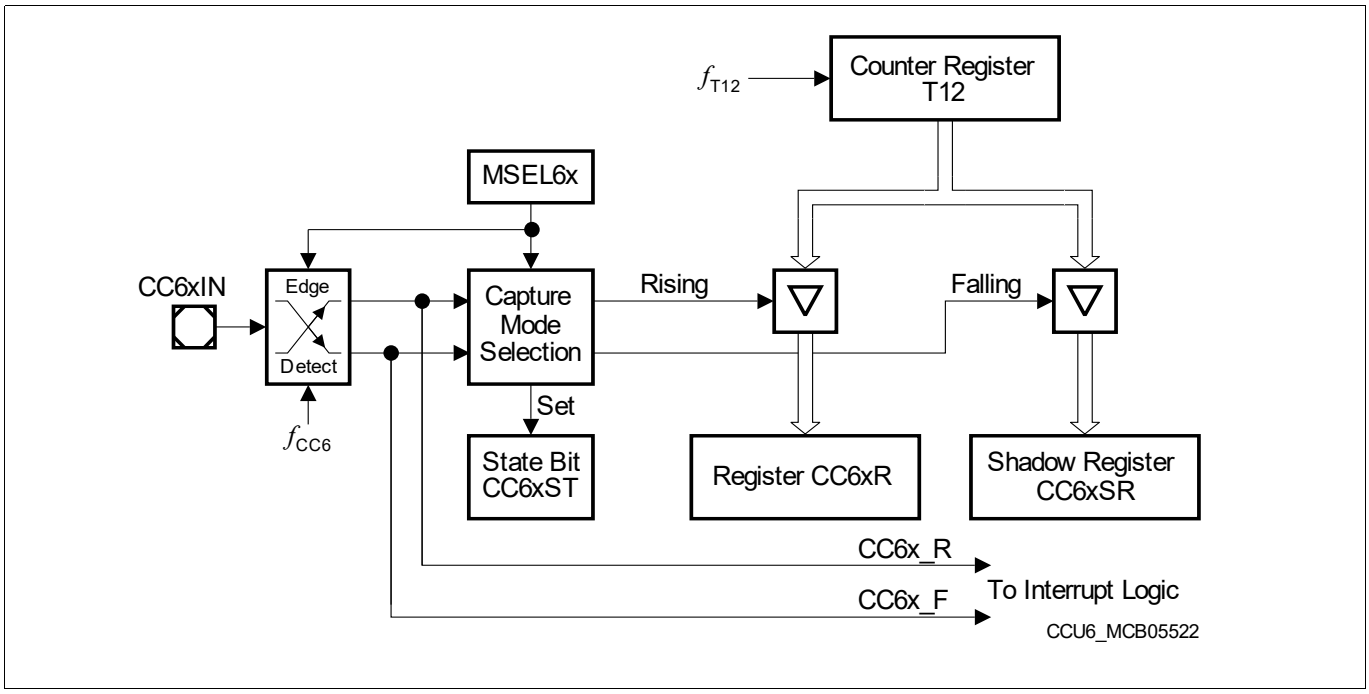


Figure 173 Capture Mode 1 Block Diagram

Capture Modes 2, 3 and 4 are shown in **Figure 174**. They differ only in the active edge causing the capture operation. In each of the three modes, when the selected edge is detected at the corresponding input signal CC6xIN, the current contents of the shadow register CC6xSR are transferred into register CC6xR, and the current Timer T12 contents are captured in register CC6xSR (simultaneous transfer). The active edge is a rising edge of CC6xIN for Capture Mode 2, a falling edge for Mode 3, and both, a rising or a falling edge for Capture Mode 4, as shown in **Table 192**. These capture modes are very useful in cases where there is little time between two consecutive edges of the input signal.

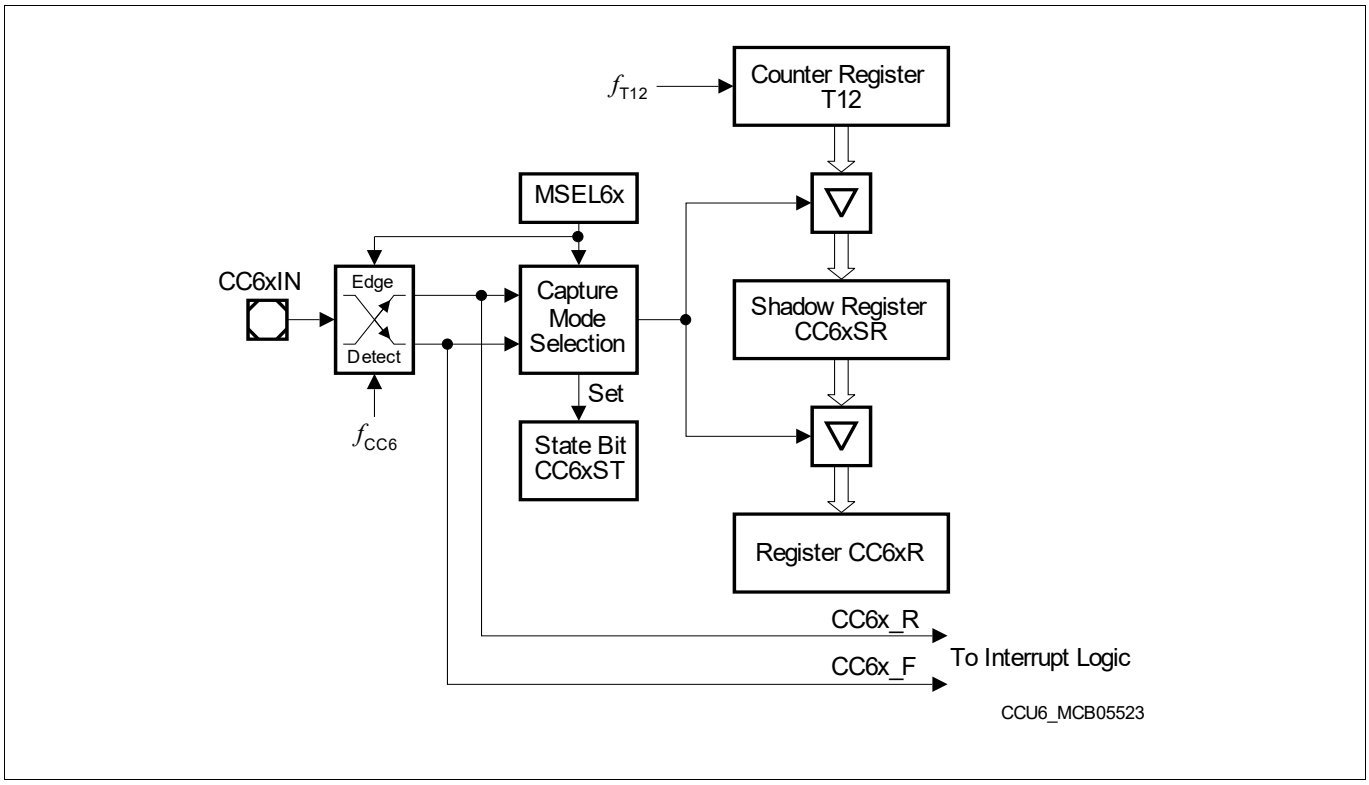


Figure 174 Capture Modes 2, 3 and 4 Block Diagram

Five further capture modes are called **Multi-Input Capture Modes**, as they use two different external inputs, signal CC6xIN and signal CCPOSx.

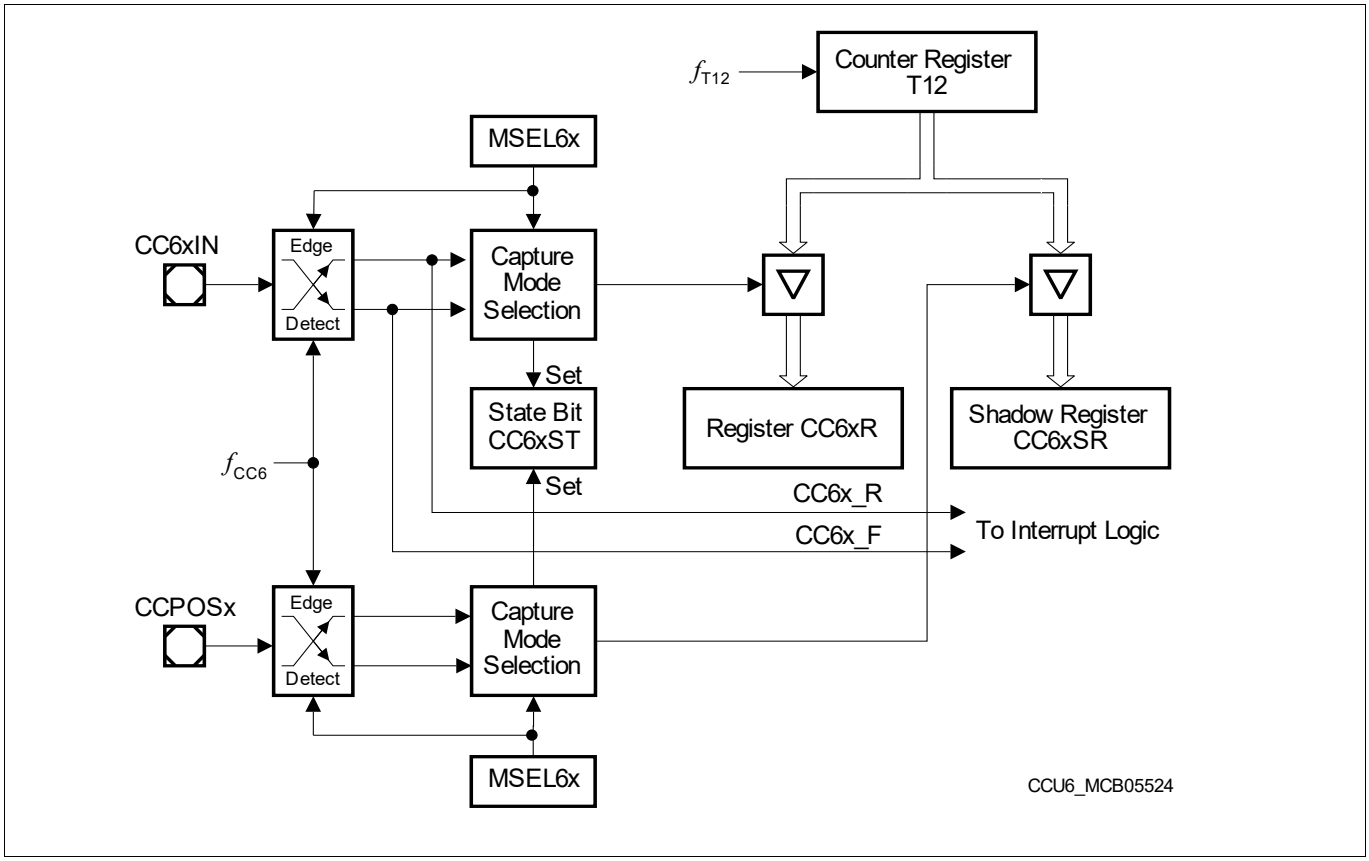


Figure 175 Multi-Input Capture Modes Block Diagram

In each of these modes, the current T12 contents are captured in register CC6xR in response to a selected event at signal CC6xIN, and in register CC6xSR in response to a selected event at signal CCPOSx. The possible events can be opposite input transitions, or the same transitions, or any transition at the two inputs. The different options are detailed in **Table 193**.

In each of the various capture modes, the Channel State Bit, CC6xST, is set to 1 when the selected capture trigger event at signal CC6xIN or CCPOSx has occurred. The State Bit is not cleared by hardware, but can be cleared by software.

In addition, appropriate signal lines to the interrupt logic are activated, that can generate an interrupt request to the CPU. Regardless of the selected active edge, all edges detected at signal CC6xIN can lead to the activation of the appropriate interrupt request line (see also **Section 29.10**).

Table 193 Multi-Input Capture Modes Overview

MSEL6x	Mode	Signal	Active Edge	T12 Stored in
1010_B	5	CC6xIN	Rising	CC6xR
		CCPOSx	Falling	CC6xSR
1011_B	6	CC6xIN	Falling	CC6xR
		CCPOSx	Rising	CC6xSR
1100_B	7	CC6xIN	Rising	CC6xR
		CCPOSx	Rising	CC6xSR

Table 193 Multi-Input Capture Modes Overview (cont'd)

MSEL6x	Mode	Signal	Active Edge	T12 Stored in
1101_B	8	CC6xIN	Falling	CC6xR
		CCPOSx	Falling	CC6xSR
1110_B	9	CC6xIN	Any	CC6xR
		CCPOSx	Any	CC6xSR
1111_B	–	Reserved (no capture or compare action)		

29.3.6 T12 Shadow Register Transfer

A special shadow transfer signal (T12_ST) can be generated to facilitate updating the period and compare values of the compare channels CC60, CC61, and CC62 synchronously to the operation of T12. Providing a shadow register for values defining one PWM period facilitates a concurrent update by software for all relevant parameters. The next PWM period can run with a new set of parameters. The generation of this signal is requested by software via bit **TCTR0**.STE12 (set by writing 1 to the write-only bit **TCTR4**.T12STR, cleared by writing 1 to the write-only bit **TCTR4**.T12STD).

Figure 176 shows the shadow register structure and the shadow transfer signals, as well as on the read/write accessibility of the various registers.

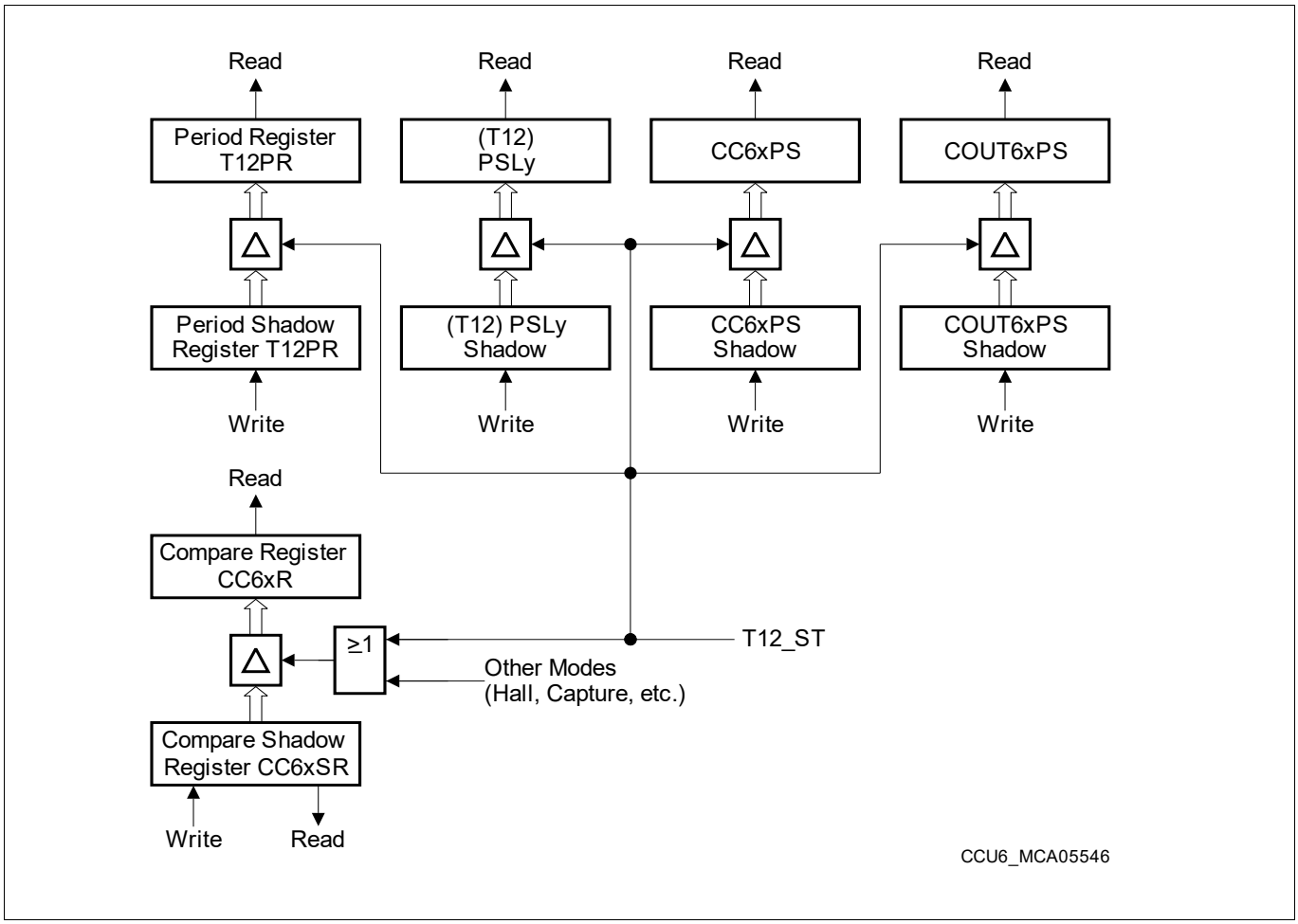


Figure 176 T12 Shadow Register Overview

A T12 shadow register transfer takes place (T12_ST active):

- While timer T12 is not running (T12R = 0), or
- STE12 = 1 and a Period-Match is detected while counting up, or
- STE12 = 1 and a One-Match is detected while counting down

When signal T12_ST is active, a shadow register transfer is triggered with the next cycle of the T12 clock. Bit STE12 is automatically cleared with the shadow register transfer.

29.3.7 Timer T12 Operating Mode Selection

The operating mode for the T12 channels are defined by the bit fields **T12MSEL**.MSEL6x.

Table 194 T12 Capture/Compare Modes Overview

MSEL6x	Selected Operating Mode
0000_B, 1111_B	Capture/Compare modes switched off
0001_B, 0010_B, 0011_B	Compare mode, see **Section 29.3.3** same behavior for all three codings
$01XX_B$	Double-Register Capture modes, see **Section 29.3.5**
1000_B	Hall Sensor Mode, see **Section 29.8** In order to properly enable this mode, all three MSEL6x fields have to be programmed to Hall Sensor mode.
1001_B	Hysteresis-like compare mode, see **Section 29.3.3.3**
1010_B 1011_B, 1100_B, 1101_B 1110_B	Multi-Input Capture modes, see **Section 29.3.5**

The clocking and counting scheme of the timers are controlled by the timer control registers **TCTR0** and **TCTR2**. Specific actions are triggered by write operations to register **TCTR4**.

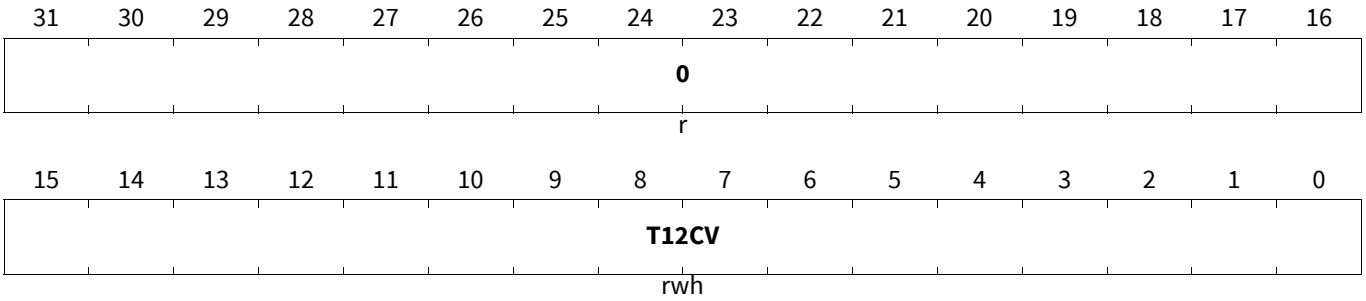
29.3.8 T12 related Registers

Timer T12 Counter Register

Register T12 represents the counting value of timer T12. It can only be written while the timer T12 is stopped. Write actions while T12 is running are not taken into account. Register T12 can always be read by SW. In edge-aligned mode, T12 only counts up, whereas in center-aligned mode, T12 can count up and down.

Note: While timer T12 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

T12
Timer T12 Counter Register **(0020_H)** **Application Reset Value: $0000\ 0000_H$**

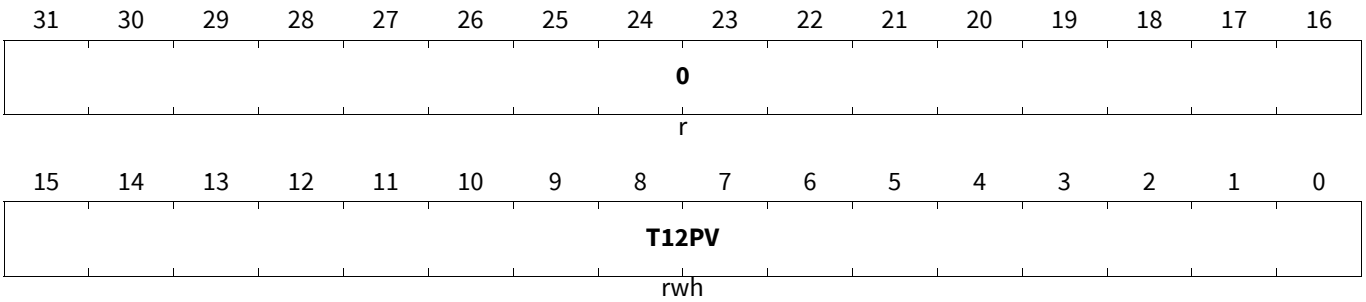


Field	Bits	Type	Description
T12CV	15:0	rwh	**Timer 12 Counter Value** This register represents the 16-bit counter value of Timer12.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Timer 12 Period Register

Register T12PR contains the period value for timer T12. The period value is compared to the actual counter value of T12 and the resulting counter actions depend on the defined counting rules. This register has a shadow register and the shadow transfer is controlled by bit STE12. A read action by SW delivers the value that is currently used for the compare action, whereas the write action targets a shadow register. The shadow register structure allows a concurrent update of all T12-related values.

T12PR
Timer 12 Period Register **(0024_H)** **Application Reset Value: $0000\ 0000_H$**



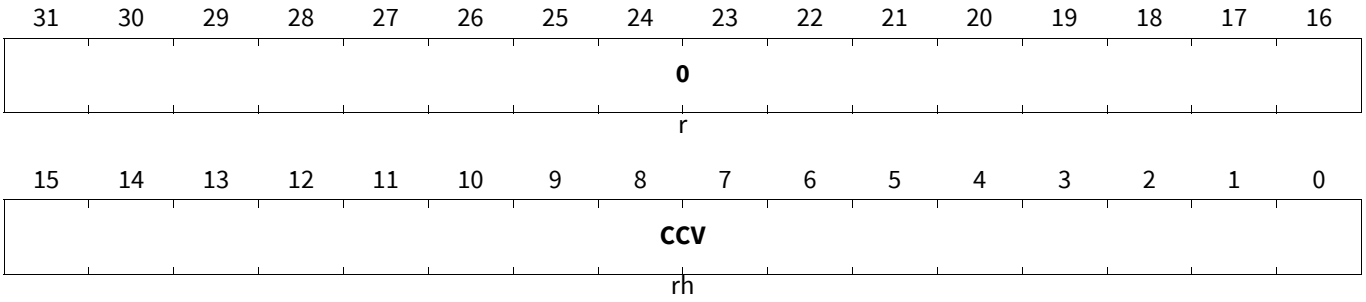
Field	Bits	Type	Description
T12PV	15:0	rwh	**T12 Period Value** The value T12PV defines the counter value for T12 leading to a period-match. When reaching this value, the timer T12 is set to zero (edge-aligned mode) or changes its count direction to down counting (center-aligned mode).
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Capture/Compare Register for Channel CC6x

In compare mode, the registers CC6xR (x = 0 - 2) are the actual compare registers for T12. The values stored in CC6xR are compared (all three channels in parallel) to the counter value of T12. In capture mode, the current value of the T12 counter register is captured by registers CC6xR if the corresponding capture event is detected.

CC6xR (x=0-2)

Capture/Compare Register for Channel CC6x (0030$_H$+x*4) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CCV	15:0	rh	**Capture/Compare Value** In compare mode, the bit fields CCV contain the values, that are compared to the T12 counter value. In capture mode, the captured value of T12 can be read from these registers.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

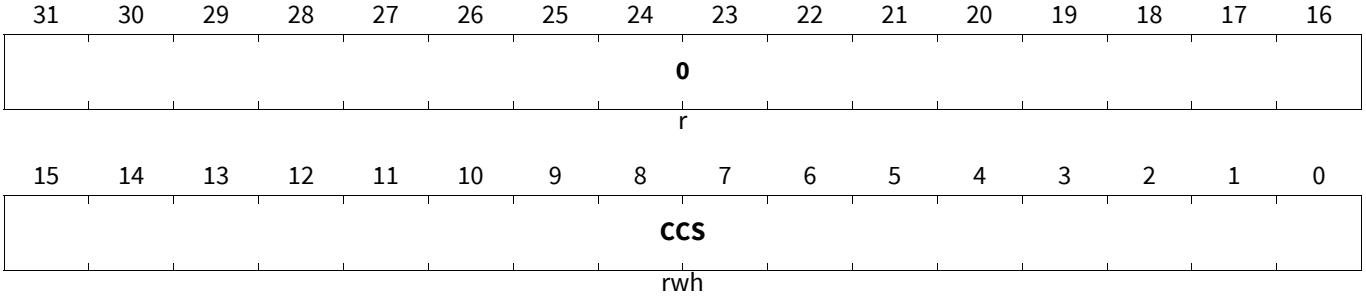
Capture/Compare Shadow Reg. for Channel CC6x

The registers CC6xR can only be read by SW, the modification of the value is done by a shadow register transfer from register CC6xSR. The corresponding shadow registers CC6xSR can be read and written by SW. In capture mode, the value of the T12 counter register can also be captured by registers CC6xSR if the selected capture event is detected (depending on the selected capture mode).

Note: The shadow registers can also be written by SW in capture mode. In this case, the HW capture event wins over the SW write if both happen in the same cycle (the SW write is discarded).

CC6xSR (x=0-2)

Capture/Compare Shadow Reg. for Channel CC6x(0040_H+x*4) **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CCS	15:0	rwh	**Shadow Register for Channel x Capture/Compare Value** In compare mode, the bit fields contents of CCS are transferred to the bit fields CCV for the corresponding channel during a shadow transfer. In capture mode, the captured value of T12 can be read from these registers.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

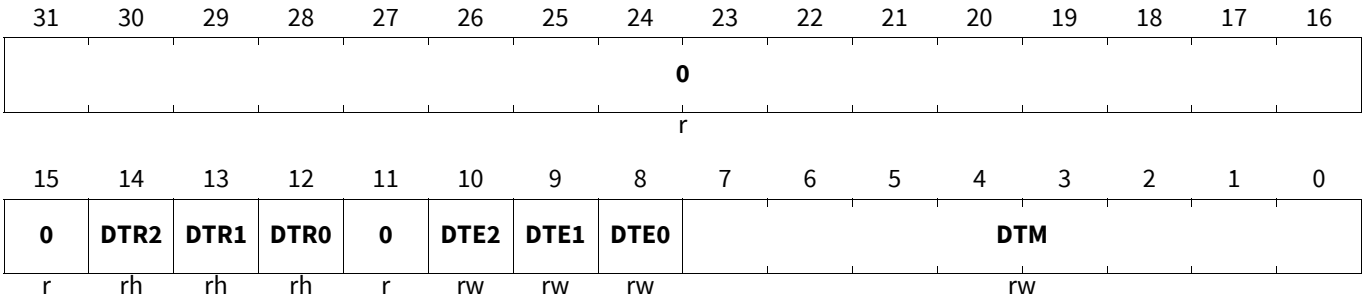
Dead-Time Control Register for Timer12

Register T12DTC controls the dead-time generation for the timer T12 compare channels. Each channel can be independently enabled/disabled for dead-time generation. If enabled, the transition from passive state to active state is delayed by the value defined by bit field DTM. The dead time counters are clocked with the same frequency as T12. This structure allows symmetrical dead-time generation in center-aligned and in edge-aligned PWM mode. A duty cycle of 50% leads to CC6x, COUT6x switched on for: 0.5 * period - dead time.

Note: The dead-time counters are not reset by bit T12RES, but by bit DTRES.

T12DTC

Dead-Time Control Register for Timer12 (0028_H) **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
DTM	7:0	rw	**Dead-Time** Bit field DTM determines the programmable delay between switching from the passive state to the active state of the selected outputs. The switching from the active state to the passive state is not delayed.

Field	Bits	Type	Description
DTEx (x=0-2)	x+8	rw	**Dead Time Enable Bits DTEx (x=0,1,2)** Bits DTE0…DTE2 enable and disable the dead time generation for each compare channel (0, 1, 2) of timer T12. 0_B Dead-Time Counter x is disabled. The corresponding outputs switch from the passive state to the active state (according to the actual compare status) without any delay. 1_B Dead-Time Counter x is enabled. The corresponding outputs switch from the passive state to the active state (according to the compare status) with the delay programmed in bit field DTM.
DTRx (x=0-2)	x+12	rh	**Dead Time Run Indication Bits DTRx (x=1,2,3)** Bits DTR0…DTR2 indicate the status of the dead time generation for each compare channel (0, 1, 2) of timer T12. 0_B Dead-Time Counter x is currently in the passive state. 1_B Dead-Time Counter x is currently in the active state.
0	11, 31:15	r	**Reserved;** Returns 0 if read; should be written with 0.

29.3.9 Capture/Compare Control Registers

Compare State Register

The Compare State Register CMPSTAT contains status bits monitoring the current capture and compare state and control bits defining the active/passive state of the compare channels.

Note: Bits CC6xPS, COUT6xPS (x = 0-2) and COUT63PS have shadow bits and are updated in parallel to the capture/compare registers of T12, T13 respectively. A read action targets the actually used values, whereas a write action targets the shadow bits.

Note: Bit T13IM has a shadow bit and is updated in parallel to the compare and period registers of T13. A read action targets the actually used values, whereas a write action targets the shadow bit.

CMPSTAT
Compare State Register **(0060_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13IM	COUT63PS	COUT62PS	CC62PS	COUT61PS	CC61PS	COUT60PS	CC60PS	0	CC63ST	CCPOS62	CCPOS61	CCPOS60	CC62ST	CC61ST	CC60ST
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CC6xST (x=0-2)	x	rh	**Capture/Compare State Bits for CC6x (x = 0, 1, 2)** Bits CC6xST monitor the state of the capture/compare channels. Bits CC6xST (x = 0, 1, 2) are related to T12 and are set and cleared according to the T12 switching rules. 0_B In compare mode, the timer count is less than the compare value. In capture mode, the selected edge has not yet been detected since the bit has been cleared by SW the last time. 1_B In compare mode, the counter value is greater than or equal to the compare value. In capture mode, the selected edge has been detected.
CCPOS6x (x=0-2)	x+3	rh	**Sampled Hall Pattern Bits** Bits CCPOS6x (x = 0, 1, 2) are indicating the value of the input Hall pattern that has been compared to the current and expected value. The value is sampled when the event HCRDY (Hall Compare Ready) occurs. 0_B The input CCPOSx has been sampled as 0. 1_B The input CCPOSx has been sampled as 1.

Field	Bits	Type	Description
CC63ST	6	rh	**Capture/Compare State Bit for CC63** Bit CC63ST monitors the state of the compare channel. Bit CC63ST is related to T13 and is set and cleared according to the T13 switching rules. 0_B In compare mode, the timer count is less than the compare value. 1_B In compare mode, the counter value is greater than or equal to the compare value.
CC6xPS (x=0-2)	2*x+8	rwh	**Passive State Select for Compare Outputs CC6x (x = 0, 1, 2)** Bits CC6xPS select the state of the corresponding compare channel, that is considered to be the passive state. During the passive state, the passive level (defined in register PSLR) is driven by the output pin. Bits CC6xPS (x = 0, 1, 2) are related to T12. In capture mode, these bits are not used. 0_B The corresponding compare signal is in passive state while CC6xST is 0. 1_B The corresponding compare signal is in passive state while CC6xST is 1.
COUT6xPS (x=0-2)	2*x+9	rwh	**Passive State Select for Compare Outputs COUT6x (x = 0, 1, 2)** Bits COUT6xPS select the state of the corresponding compare channel, that is considered to be the passive state. During the passive state, the passive level (defined in register PSLR) is driven by the output pin. Bits COUT6xPS (x = 0, 1, 2) are related to T12. In capture mode, these bits are not used. 0_B The corresponding compare signal is in passive state while CC6xST is 0. 1_B The corresponding compare signal is in passive state while CC6xST is 1.
COUT63PS	14	rwh	**Passive State Select for Compare Output COUT63** Bit COUT63PS selects the state of the corresponding compare channel, that is considered to be the passive state. During the passive state, the passive level (defined in register PSLR) is driven by the output pin. Bit COUT63PS is related to T13. 0_B The corresponding compare signal is in passive state while CC63ST is 0. 1_B The corresponding compare signal is in passive state while CC63ST is 1.
T13IM	15	rwh	**T13 Inverted Modulation** Bit T13IM inverts the T13 signal for the modulation of the CC6x and COUT6x (x = 0, 1, 2) signals. 0_B T13 output CC63_O is equal to CC63ST. 1_B T13 output CC63_O is equal to $\overline{\text{CC63ST}}$.
0	7, 31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Compare State Modification Register

The Compare Status Modification Register CMPMODIF provides software-control (independent set and clear conditions) for the channel state bits CC6xST. This feature enables the user to individually change the status of the output lines by software, for example when the corresponding compare timer is stopped, by a single data

write action. The functionality of a write access to bits concerning the same capture/compare state bit is shown in the following **Table 195**.

Table 195 Write Access to Bit Pair MCC6xR, MCC6xS in Register CMPMODIF

[MCC6xR, MCC6xS]	Function
00_B	Bit CC6xST is not changed
01_B	Bit CC6xST is set
10_B	Bit CC6xST is cleared
11_B	Reserved

CMPMODIF
Compare State Modification Register (0064_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	MCC63R	0			MCC62R	MCC61R	MCC60R	0	MCC63S	0			MCC62S	MCC61S	MCC60S
r	w	r			w	w	w	r	w	r			w	w	w

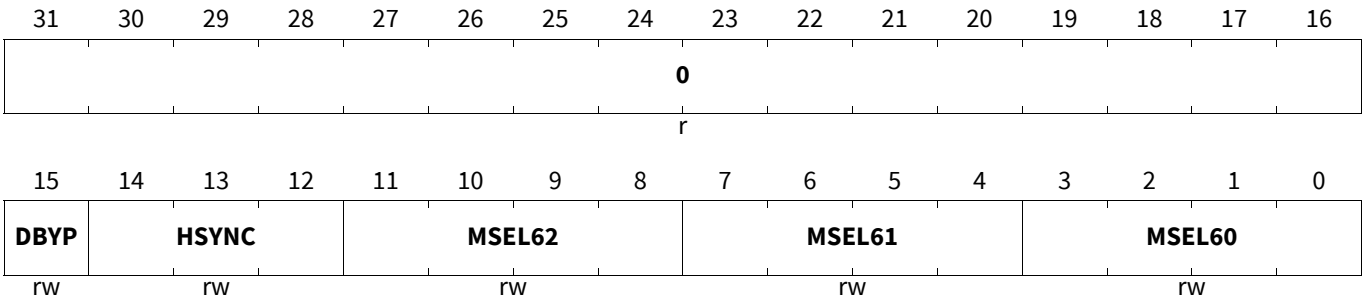
Field	Bits	Type	Description
MCC6xS (x=0-2)	x	w	**Capture/Compare Status Modification Bits MCC6xS (x = 0, 1, 2)** These bits are used to set the corresponding bits CC6xST by SW. The functionality of a write access to bits concerning the same capture/compare state bit is shown in **Table 195**.
MCC63S	6	w	**Capture/Compare Status Modification Bit MCC63S** This bit is used to set the corresponding bit CC63ST by SW. The functionality of a write access to bits concerning the same capture/compare state bit is shown in **Table 195**.
MCC6xR (x=0-2)	x+8	w	**Capture/Compare Status Modification Bits MCC6xR (x = 0, 1, 2)** These bits are used to clear the corresponding bits CC6xST by SW. The functionality of a write access to bits concerning the same capture/compare state bit is shown in **Table 195**.
MCC63R	14	w	**Capture/Compare Status Modification Bits MCC63R** This bit is used to clear the corresponding bit CC63ST by SW. The functionality of a write access to bits concerning the same capture/compare state bit is shown in **Table 195**.
0	5:3, 7, 13:11, 31:15	r	**Reserved;** Returns 0 if read; should be written with 0.

T12 Mode Select Register

Register T12MSEL contains control bits to select the capture/compare functionality of the three channels of Timer T12.

T12MSEL

T12 Mode Select Register (0068$_H$) Application Reset Value: 0000 0000$_H$



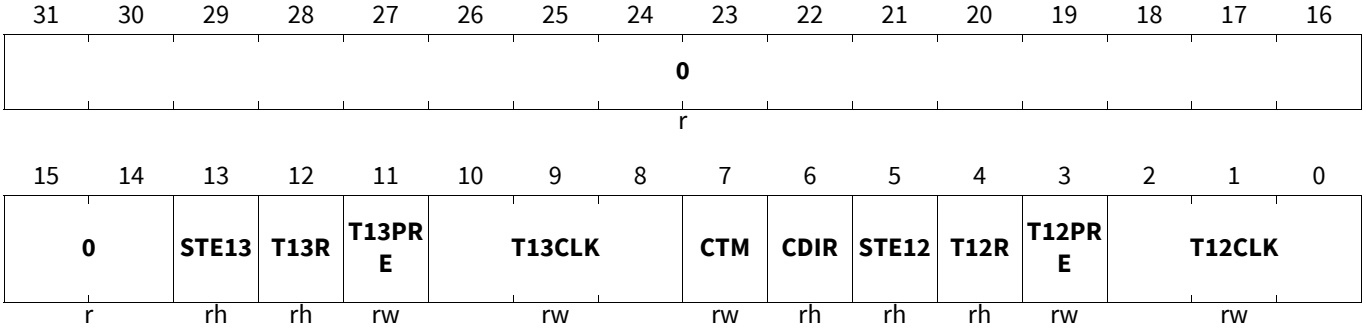
Field	Bits	Type	Description
MSEL6x (x=0-2)	4*x+3:4*x	rw	**Capture/Compare Mode Selection MSEL6x (x=0,1,2)** These bit fields select the operating mode of the three T12 capture/compare channels. Each channel (x = 0, 1, 2) can be programmed individually for one of these modes (except for Hall Sensor Mode). Coding see **Table 194**.
HSYNC	14:12	rw	**Hall Synchronization** Bit field HSYNC defines the source for the sampling of the Hall input pattern and the comparison to the current and the expected Hall pattern bit fields. Coding see **Table 201**.
DBYP	15	rw	**Delay Bypass** DBYP controls whether the source signal for the sampling of the Hall input pattern (selected by HSYNC) is delayed by the Dead-Time Counter 0. 0_B The bypass is not active. Dead-Time Counter 0 is generating a delay after the source signal becomes active. 1_B The bypass is active. Dead-Time Counter 0 is not used for a delay.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Timer Control Register 0

Register TCTR0 controls the basic functionality of both timers, T12 and T13.

Note: A write action to the bit fields T12CLK or T12PRE is only taken into account while the timer T12 is not running (T12R=0). A write action to the bit fields T13CLK or T13PRE is only taken into account while the timer T13 is not running (T13R=0).

TCTR0
Timer Control Register 0 **(0070_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T12CLK	2:0	rw	**Timer T12 Input Clock Select** Selects the input clock for timer T12 that is derived from the peripheral clock according to the equation $f_{T12} = f_{CC6} / 2^{<T12CLK>}$. 000_B $f_{T12} = f_{CC6}$ 001_B $f_{T12} = f_{CC6} / 2$ 010_B $f_{T12} = f_{CC6} / 4$ 011_B $f_{T12} = f_{CC6} / 8$ 100_B $f_{T12} = f_{CC6} / 16$ 101_B $f_{T12} = f_{CC6} / 32$ 110_B $f_{T12} = f_{CC6} / 64$ 111_B $f_{T12} = f_{CC6} / 128$
T12PRE	3	rw	**Timer T12 Prescaler Bit** In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T12. 0_B The additional prescaler for T12 is disabled. 1_B The additional prescaler for T12 is enabled.
T12R	4	rh	**Timer T12 Run Bit** T12R starts and stops timer T12. It is set/cleared by SW by setting bits T12RR or T12RS or it is cleared by HW according to the function defined by bit field T12SSC. 0_B Timer T12 is stopped. 1_B Timer T12 is running.
STE12	5	rh	**Timer T12 Shadow Transfer Enable** Bit STE12 enables or disables the shadow transfer of the T12 period value, the compare values and passive state select bits and levels from their shadow registers to the actual registers if a T12 shadow transfer event is detected. Bit STE12 is cleared by hardware after the shadow transfer. A T12 shadow transfer event is a period-match while counting up or a one-match while counting down. 0_B The shadow register transfer is disabled. 1_B The shadow register transfer is enabled.

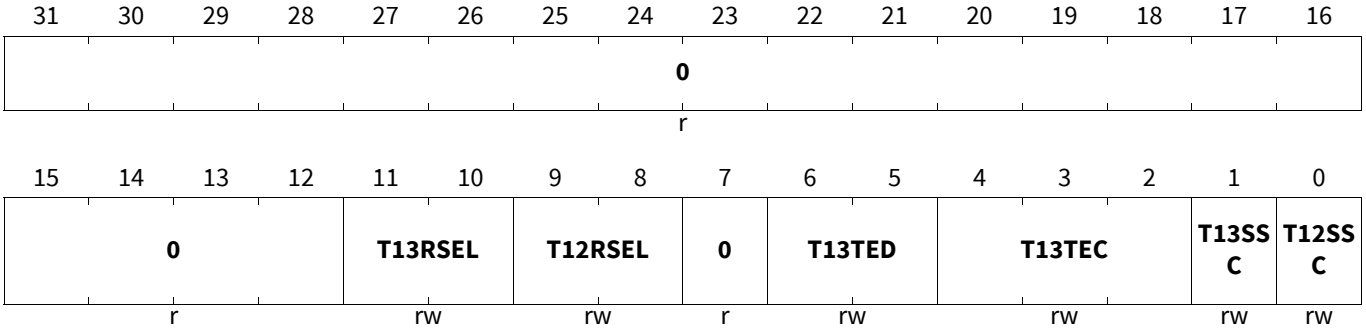
Field	Bits	Type	Description
CDIR	6	rh	**Count Direction of Timer T12** This bit is set/cleared according to the counting rules of T12. 0_B T12 counts up. 1_B T12 counts down.
CTM	7	rw	**T12 Operating Mode** 0_B Edge-aligned Mode: T12 always counts up and continues counting from zero after reaching the period value. 1_B Center-aligned Mode: T12 counts down after detecting a period-match and counts up after detecting a one-match.
T13CLK	10:8	rw	**Timer T13 Input Clock Select** Selects the input clock for timer T13 that is derived from the peripheral clock according to the equation $f_{T13} = f_{CC6} / 2^{<T13CLK>}$. 000_B $f_{T13} = f_{CC6}$ 001_B $f_{T13} = f_{CC6} / 2$ 010_B $f_{T13} = f_{CC6} / 4$ 011_B $f_{T13} = f_{CC6} / 8$ 100_B $f_{T13} = f_{CC6} / 16$ 101_B $f_{T13} = f_{CC6} / 32$ 110_B $f_{T13} = f_{CC6} / 64$ 111_B $f_{T13} = f_{CC6} / 128$
T13PRE	11	rw	**Timer T13 Prescaler Bit** In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T13. 0_B The additional prescaler for T13 is disabled. 1_B The additional prescaler for T13 is enabled.
T13R	12	rh	**Timer T13 Run Bit** T13R starts and stops timer T13. It is set/cleared by SW by setting bits T13RR orT13RS or it is set/cleared by HW according to the function defined by bit fields T13SSC, T13TEC and T13TED. 0_B Timer T13 is stopped. 1_B Timer T13 is running.
STE13	13	rh	**Timer T13 Shadow Transfer Enable** Bit STE13 enables or disables the shadow transfer of the T13 period value, the compare value and passive state select bit and level from their shadow registers to the actual registers if a T13 shadow transfer event is detected. Bit STE13 is cleared by hardware after the shadow transfer. A T13 shadow transfer event is a period-match. 0_B The shadow register transfer is disabled. 1_B The shadow register transfer is enabled.
0	31:14	r	**Reserved;** Returns 0 if read; should be written with 0.

Timer Control Register 2

Register TCTR2 controls the single-shot and the synchronization functionality of both timers T12 and T13. Both timers can run in single-shot mode. In this mode they stop their counting sequence automatically after one

counting period with a count value of zero. The single-shot mode and the synchronization feature of T13 to T12 allow the generation of events with a programmable delay after well-defined PWM actions of T12.

TCTR2
Timer Control Register 2 **(0074_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T12SSC	0	rw	**Timer T12 Single Shot Control** This bit controls the single shot-mode of T12. 0_B The single-shot mode is disabled, no HW action on T12R. 1_B The single shot mode is enabled, the bit T12R is cleared by HW if - T12 reaches its period value in edge-aligned mode - T12 reaches the value 1 while down counting in center-aligned mode. In parallel to the clear action of bit T12R, the bits CC6xST (x=0, 1, 2) are cleared.
T13SSC	1	rw	**Timer T13 Single Shot Control** This bit controls the single shot-mode of T13. 0_B No HW action on T13R 1_B The single-shot mode is enabled, the bit T13R is cleared by HW if T13 reaches its period value. In parallel to the clear action of bit T13R, the bit CC63ST is cleared.
T13TEC	4:2	rw	**T13 Trigger Event Control** Bit field T13TEC selects the trigger event to start T13 (automatic set of T13R for synchronization to T12 compare signals) according to following combinations: 000_B No action 001_B Set T13R on a T12 compare event on channel 0 010_B Set T13R on a T12 compare event on channel 1 011_B Set T13R on a T12 compare event on channel 2 100_B Set T13R on any T12 compare event (ch. 0, 1, 2) 101_B Set T13R upon a period-match of T12 110_B Set T13R upon a zero-match of T12 (while counting up) 111_B Set T13R on any edge of inputs CCPOSx

Field	Bits	Type	Description
T13TED	6:5	rw	**Timer T13 Trigger Event Direction** Bit field T13TED delivers additional information to control the automatic set of bit T13R in the case that the trigger action defined by T13TEC is detected. 00_B Reserved, no action 01_B While T12 is counting up 10_B While T12 is counting down 11_B Independent on the count direction of T12
T12RSEL	9:8	rw	**Timer T12 External Run Selection** Bit field T12RSEL defines the event of signal T12HR that can set the run bit T12R by HW. 00_B The external setting of T12R is disabled. 01_B Bit T12R is set if a rising edge of signal T12HR is detected. 10_B Bit T12R is set if a falling edge of signal T12HR is detected. 11_B Bit T12R is set if an edge of signal T12HR is detected.
T13RSEL	11:10	rw	**Timer T13 External Run Selection** Bit field T13RSEL defines the event of signal T13HR that can set the run bit T13R by HW. 00_B The external setting of T13R is disabled. 01_B Bit T13R is set if a rising edge of signal T13HR is detected. 10_B Bit T13R is set if a falling edge of signal T13HR is detected. 11_B Bit T13R is set if an edge of signal T13HR is detected.
0	7, 31:12	r	**Reserved;** Returns 0 if read; should be written with 0;

Timer Control Register 4

Register TCTR4 provides software-control (independent set and clear conditions) for the run bits T12R and T13R. Furthermore, the timers can be reset (while running) and bits STE12 and STE13 can be controlled by software. Reading these bits always returns 0.

Note: A simultaneous write of a 1 to bits that set and clear the same bit will trigger no action. The corresponding bit will remain unchanged.

TCTR4
Timer Control Register 4 **(0078_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13STD	T13STR	T13CNT	0		T13RES	T13RS	T13RR	T12STD	T12STR	T12CNT	0	DTRES	T12RES	T12RS	T12RR
w	w	w	r		w	w	w	w	w	w	r	w	w	w	w

Field	Bits	Type	Description
T12RR	0	w	**Timer T12 Run Reset** Setting this bit clears the T12R bit. 0_B T12R is not influenced. 1_B T12R is cleared, T12 stops counting.
T12RS	1	w	**Timer T12 Run Set** Setting this bit sets the T12R bit. 0_B T12R is not influenced. 1_B T12R is set, T12 starts counting.
T12RES	2	w	**Timer T12 Reset** 0_B No effect on T12. 1_B The T12 counter register is cleared to zero. The switching of the output signals is according to the switching rules. Setting of T12RES has no impact on bit T12R.
DTRES	3	w	**Dead-Time Counter Reset** 0_B No effect on the dead-time counters. 1_B The three dead-time counter channels are cleared to zero.
T12CNT	5	w	**Timer T12 Count Event** 0_B No action 1_B If enabled (PISEL2), timer T12 counts one step.
T12STR	6	w	**Timer T12 Shadow Transfer Request** 0_B No action 1_B STE12 is set, enabling the shadow transfer.
T12STD	7	w	**Timer T12 Shadow Transfer Disable** 0_B No action 1_B STE12 is cleared without triggering the shadow transfer.
T13RR	8	w	**Timer T13 Run Reset** Setting this bit clears the T13R bit. 0_B T13R is not influenced. 1_B T13R is cleared, T13 stops counting.
T13RS	9	w	**Timer T13 Run Set** Setting this bit sets the T13R bit. 0_B T13R is not influenced. 1_B T13R is set, T13 starts counting.
T13RES	10	w	**Timer T13 Reset** 0_B No effect on T13. 1_B The T13 counter register is cleared to zero. The switching of the output signals is according to the switching rules. Setting of T13RES has no impact on bit T13R.
T13CNT	13	w	**Timer T13 Count Event** 0_B No action 1_B If enabled (PISEL2), timer T13 counts one step.
T13STR	14	w	**Timer T13 Shadow Transfer Request** 0_B No action 1_B STE13 is set, enabling the shadow transfer.

Field	Bits	Type	Description
T13STD	15	w	**Timer T13 Shadow Transfer Disable** 0_B No action 1_B STE13 is cleared without triggering the shadow transfer.
0	4, 12:11, 31:16	r	**reserved;** returns 0 if read; should be written with 0;

29.4 Operating Timer T13

Timer T13 is implemented similarly to Timer T12, but only with one channel in compare mode. A 16-bit up-counter is connected to a channel register via a comparator, that generates a signal when the counter contents match the contents of the channel register. A variety of control functions facilitate the adaptation of the T13 structure to different application needs. In addition, T13 can be started synchronously to timer T12 events.

This section provides information about:

- T13 overview (see **Section 29.4.1**)
- Counting scheme (see **Section 29.4.2**)
- Compare mode (see **Section 29.4.3**)
- Compare output path (see **Section 29.4.4**)
- Shadow register transfer (see **Section 29.4.5**)
- T13 counter register description (see **Section 29.4.6**)

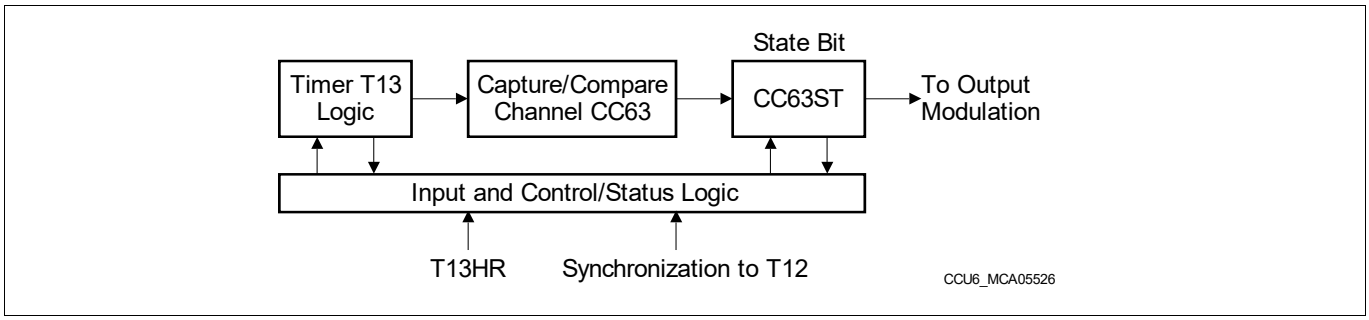


Figure 177 Overview Diagram of the Timer T13 Block

29.4.1 T13 Overview

Figure 178 shows a detailed block diagram of Timer T13. The functions of the timer T12 block are controlled by bits in registers **TCTR0**, **TCTR2**, and **PISEL2**.

Timer T13 receives its input clock, f_{T13}, from the module clock f_{CC6} via a programmable prescaler and an optional 1/256 divider or from an input signal T13HR. T13 can only count up (similar to the Edge-Aligned mode of T12).

Via a comparator, the timer T13 Counter Register **T13** is connected to the Period Register **T13PR**. This register determines the maximum count value for T13. When T13 reaches the period value, signal T13_PM (T13 Period Match) is generated and T13 is cleared to 0000_H with the next T13 clock edge. The Period Register receives a new period value from its Shadow Period Register, T13PS, that is loaded via software. The transfer of a new period value from the shadow register into T13PR is controlled via the 'T13 Shadow Transfer' control signal, T13_ST. The generation of this signal depends on the associated control bit STE13. Providing a shadow register for the period value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters (refer to **Table 29.4.5**).

Another signal indicates whether the counter contents are equal to 0000_H (T13_ZM).

A Single-Shot control bit, T13SSC, enables an automatic stop of the timer when the current counting period is finished (see **Figure 180**).

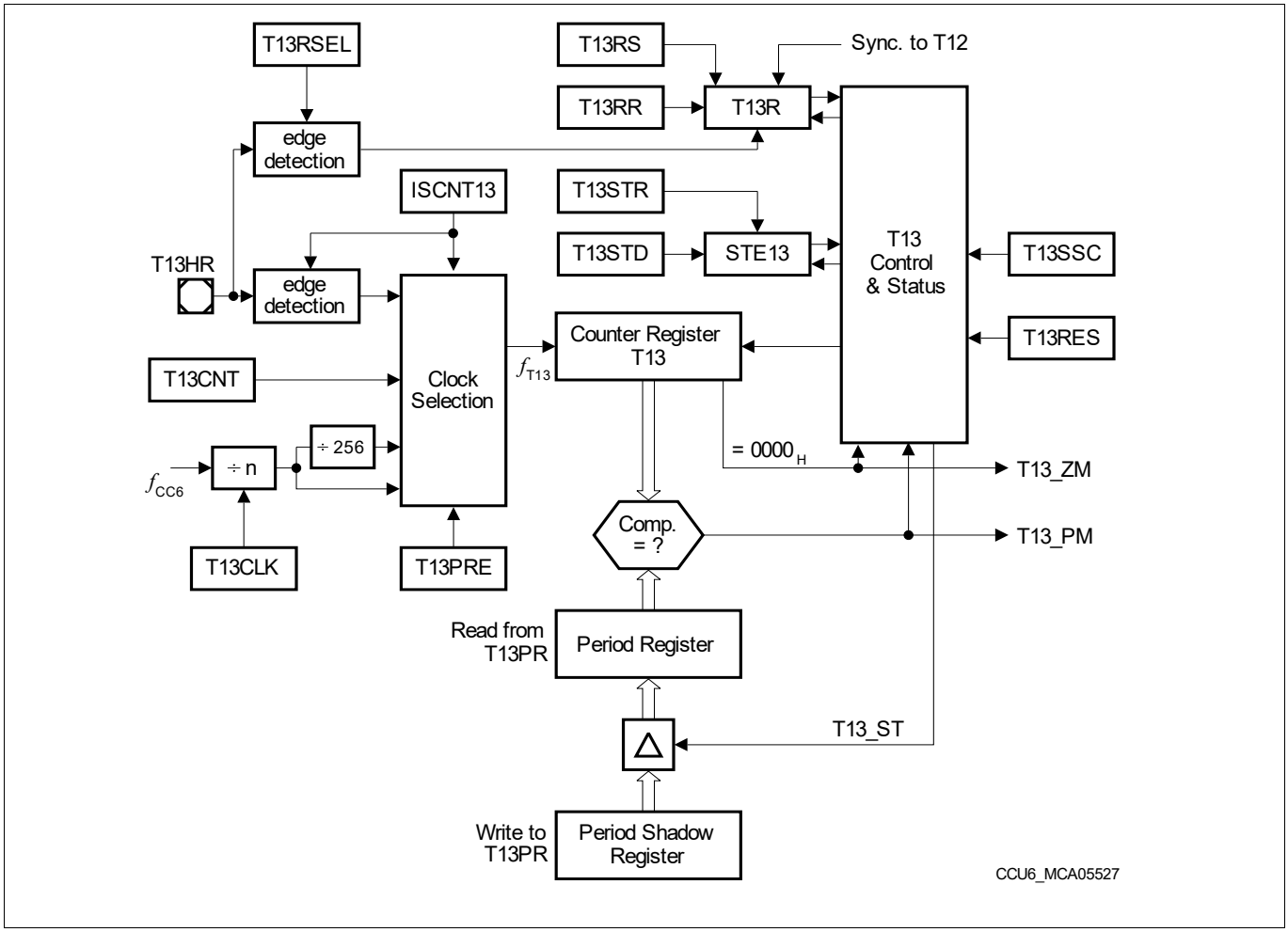


Figure 178 T13 Counter Logic and Period Comparators

The start or stop of T13 is controlled by the Run bit, T13R. This control bit can be set by software via the associated set/clear bits T13RS or T13RR in register **TCTR4**, or it is cleared by hardware according to preselected conditions (single-shot mode).
The timer T13 run bit T13R must not be set while the applied T13 period value is zero.
Bit T13R can be set automatically if an event of T12 is detected to synchronize T13 timings to T12 events, e.g. to generate a programmable delay via T13 after an edge of a T12 compare channel before triggering an AD conversion (T13 can trigger ADC conversions).
Timer T13 can be cleared to 0000_H via control bit T13RES. Setting this write-only bit only clears the timer contents, but has no further effects, e.g., it does not stop the timer.

The generation of the T13 shadow transfer control signal, T13_ST, is enabled via bit STE13. This bit can be set or cleared by software indirectly through its associated set/reset control bits T13STR and T13STD.

Two bit fields, T13TEC and T13TED, control the synchronization of T13 to Timer T12 events. T13TEC selects the trigger event, while T13TED determines for which T12 count direction the trigger should be active.
While Timer T13 is running, write accesses to the count register T13 are not taken into account. If T13 is stopped, write actions to register T13 are immediately taken into account.

Note: The T13 Period Register and its associated shadow register are located at the same physical address. A write access to this address targets the Shadow Register, while a read access reads from the actual period register.

29.4.2 T13 Counting Scheme

This section describes the clocking and the counting capabilities of T13.

29.4.2.1 Clock Selection

In **Timer Mode** (**PISEL2**. ISCNT13 = 00_B), the input clock f_{T13} of Timer T13 is derived from the internal module clock f_{CC6} through a programmable prescaler and an optional 1/256 divider. The resulting prescaler factors are listed in **Table 196**. The prescaler of T13 is cleared while T13 is not running (**TCTR0**.T13R = 0) to ensure reproducible timings and delays.

Table 196 Timer T13 Input Clock Options

T13CLK	Resulting Input Clock *f*T13 Prescaler Off (T13PRE = 0)	Resulting Input Clock *f*T13 Prescaler On (T13PRE = 1)
000_B	f_{CC6}	f_{CC6} / 256
001_B	f_{CC6} / 2	f_{CC6} / 512
010_B	f_{CC6} / 4	f_{CC6} / 1024
011_B	f_{CC6} / 8	f_{CC6} / 2048
100_B	f_{CC6} / 16	f_{CC6} / 4096
101_B	f_{CC6} / 32	f_{CC6} / 8192
110_B	f_{CC6} / 64	f_{CC6} / 16384
111_B	f_{CC6} / 128	f_{CC6} / 32768

In **Counter Mode**, timer T13 counts one step:

- If a 1 is written to **TCTR4**.T13CNT and **PISEL2**.ISCNT13 = 01_B
- If a rising edge of input signal T13HR is detected and **PISEL2**.ISCNT13 = 10_B
- If a falling edge of input signal T13HR is detected and **PISEL2**.ISCNT13 = 11_B

29.4.2.2 T13 Counting

The period of the timer is determined by the value in the period Register T13PR according to the following formula:

$T13_{PER}$ = <Period-Value> + 1; in T13 clocks (f_{T13}) (29.3)

Timer T13 can only count up, comparable to the Edge-Aligned mode of T12. This leads to very simple 'counting rule' for the T13 counter:

- The counter is cleared with the next T13 clock edge if a Period-Match is detected. The counting direction is always upwards.

The behavior of T13 is illustrated in **Figure 179**.

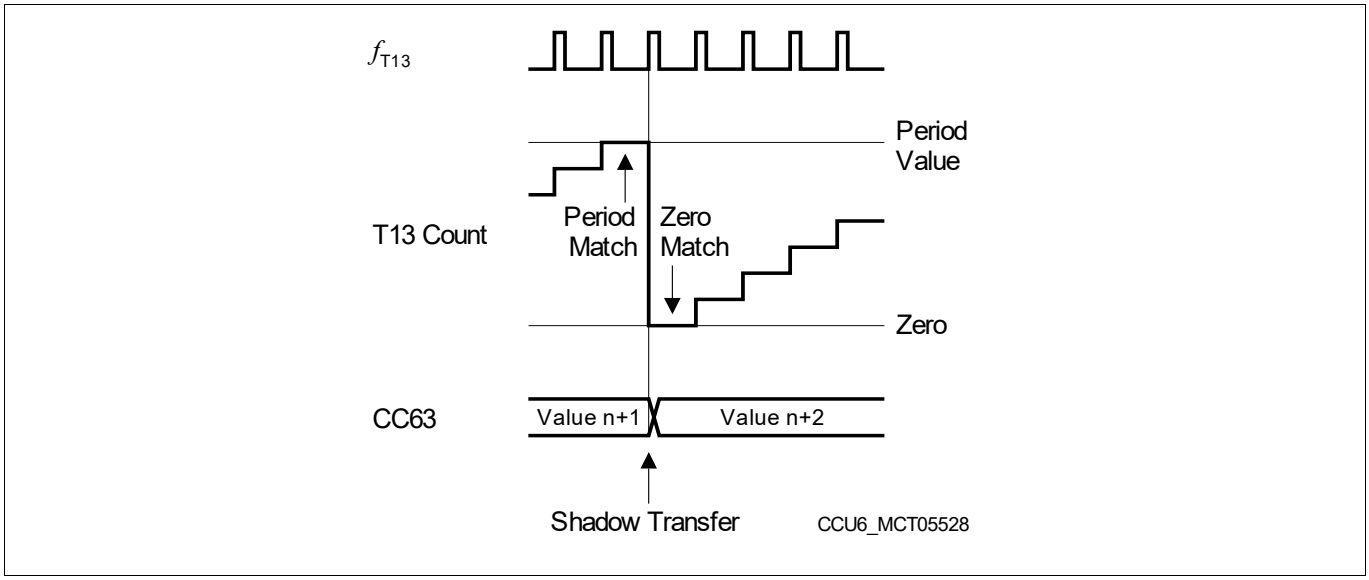


Figure 179 T13 Counting Sequence

29.4.2.3 Single-Shot Mode

In Single-Shot Mode, the timer run bit T13R is cleared by hardware. If bit T13SSC = 1, the timer T13 will stop when the current timer period is finished.

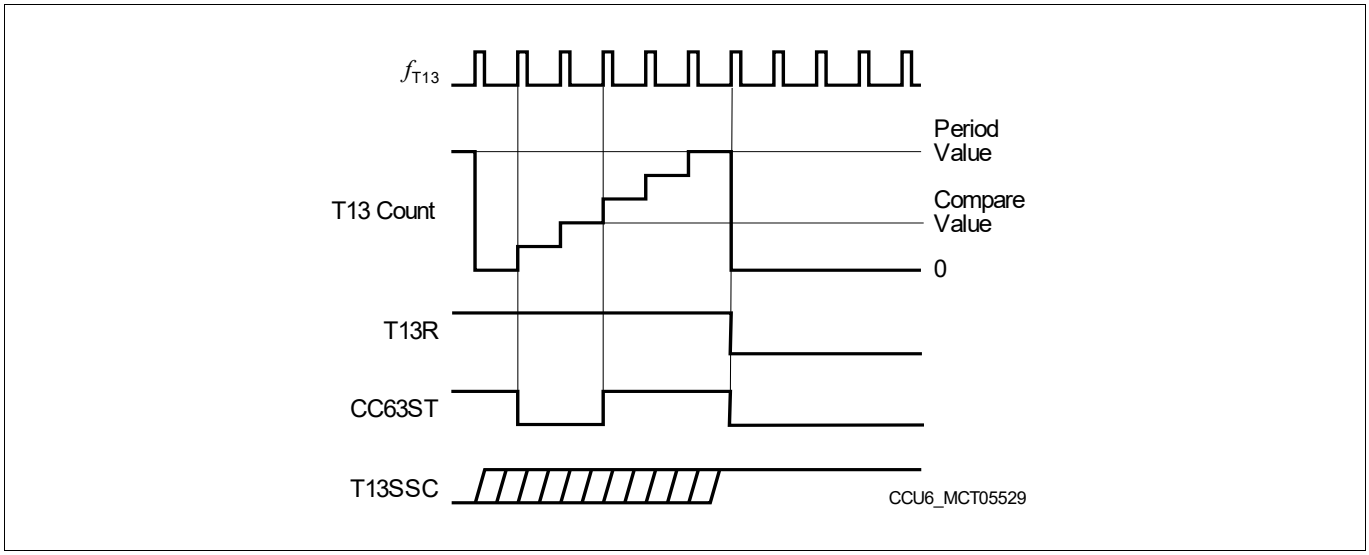


Figure 180 Single-Shot Operation of Timer T13

29.4.2.4 Synchronization to T12

Timer T13 can be synchronized to a T12 event. Bit fields T13TEC and T13TED select the event that is used to start Timer T13. The selected event sets bit T13R via HW, and T13 starts counting. Combined with the Single-Shot mode, this feature can be used to generate a programmable delay after a T12 event.

Figure 181 shows an example for the synchronization of T13 to a T12 event. Here, the selected event is a compare-match (compare value = 2) while counting up. The clocks of T12 and T13 can be different (other prescaler factor); the figure shows an example in which T13 is clocked with half the frequency of T12.

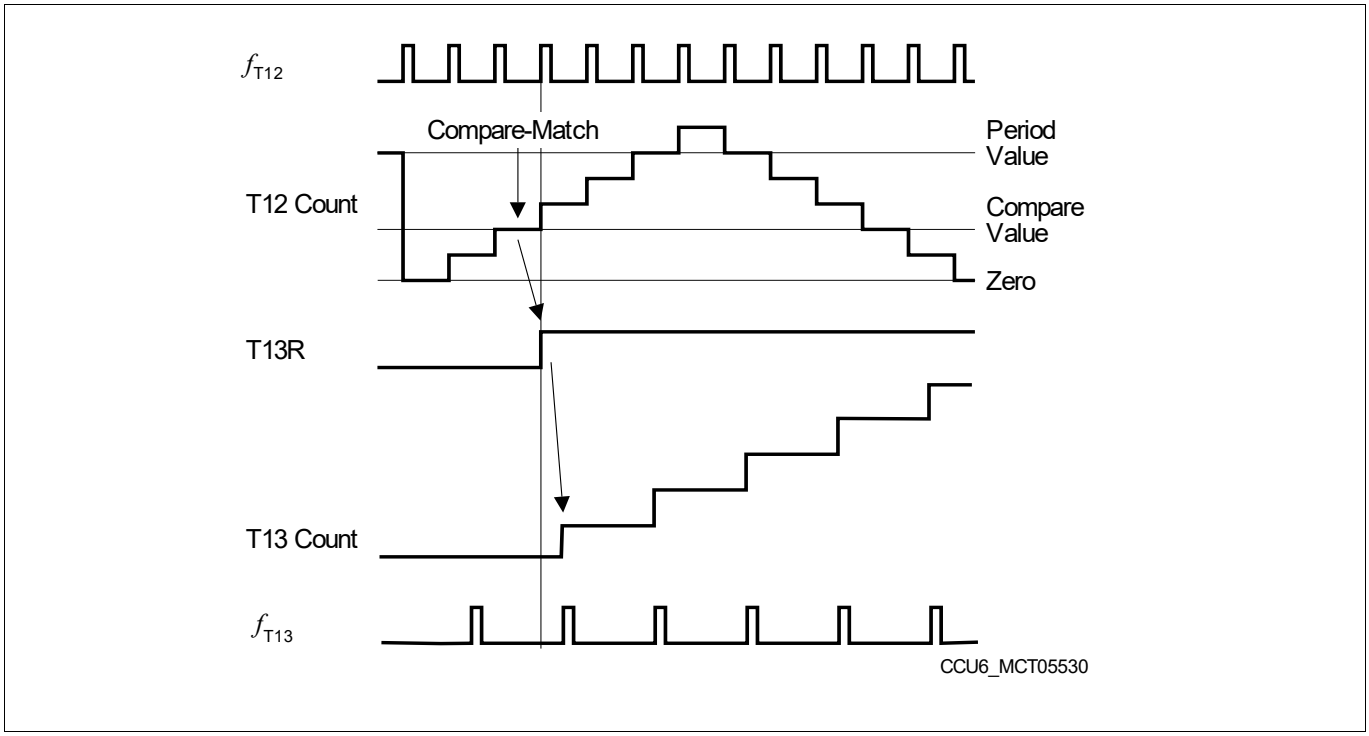


Figure 181 Synchronization of T13 to T12 Compare Match

Bit field T13TEC selects the trigger event to start T13 (automatic set of T13R for synchronization to T12 compare signals) according to the combinations shown in **Table 197**. Bit field T13TED additionally specifies for which count direction of T12 the selected trigger event should be regarded (see **Table 198**).

Table 197 T12 Trigger Event Selection

T13TEC	Selected Event
000_B	None
001_B	T12 Compare Event on Channel 0 (CM_CC60)
010_B	T12 Compare Event on Channel 1 (CM_CC61)
011_B	T12 Compare Event on Channel 2 (CM_CC62)
100_B	T12 Compare Event on any Channel (0, 1, 2)
101_B	T12 Period-Match (T12_PM)
110_B	T12 Zero-Match while counting up (T12_ZM and CDIR = 0)
111_B	Any Hall State Change

Table 198 T12 Trigger Event Additional Specifier

T13TED	Selected Event Specifier
00_B	Reserved, no action
01_B	Selected event is active while T12 is counting up (CDIR = 0)
10_B	Selected event is active while T12 is counting down (CDIR = 1)
11_B	Selected event is active independently of the count direction of T12

29.4.3 T13 Compare Mode

Associated with Timer T13 is one compare channel, that can perform compare operations with regard to the contents of the T13 counter.

Figure 177 gives an overview on the T13 channel in Compare Mode. The channel is connected to the T13 counter register via an equal-to comparator, generating a compare match signal when the contents of the counter matches the contents of the compare register.

The channel consists of the comparator and a double register structure - the actual compare register, **CC63R**, feeding the comparator, and an associated shadow register, **CC63SR**, that is preloaded by software and transferred into the compare register when signal T13 shadow transfer, T13_ST, gets active. Providing a shadow register for the compare value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

Associated with the channel is a State Bit, **CMPSTAT**.CC63ST, holding the status of the compare operation. **Figure 182** gives an overview on the logic for the State Bit.

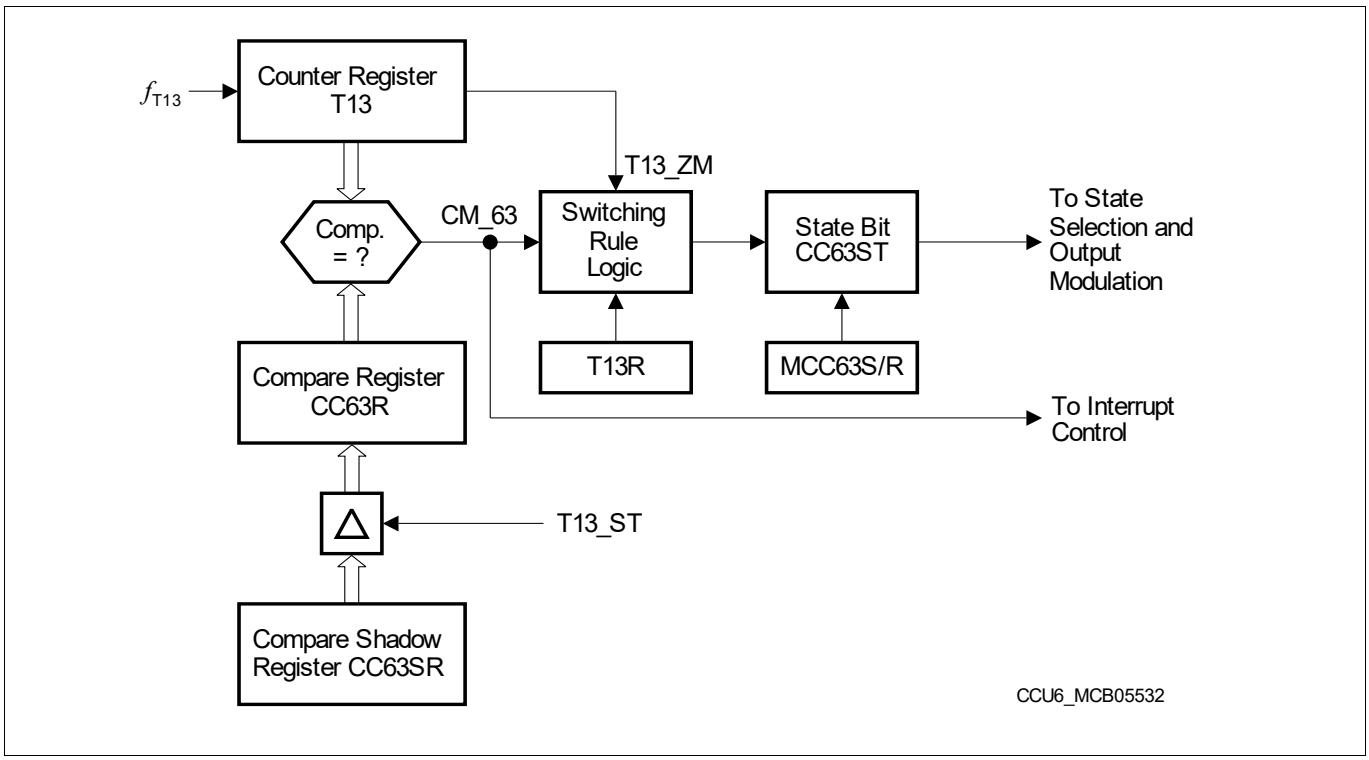


Figure 182 T13 State Bit Block Diagram

A compare interrupt event CM_63 is signaled when a compare match is detected. The actual setting of a State Bit has no influence on the interrupt generation.

The inputs to the switching rule logic for the CC63ST bit are the timer run bit (T13R), the timer zero-match signal (T13_ZM), and the actual individual compare-match signal CM_63. In addition, the state bit can be set or cleared by software via bits MCC63S and MCC63R in register **CMPMODIF**.
A modification of the State Bit CC63ST by hardware is only possible while Timer T13 is running (T13R = 1). If this is the case, the following switching rules apply for setting and resetting the State Bit in Compare Mode:

State Bit **CC63ST is set** to 1

- with the next T13 clock (f_{T13}) after a compare-match (T13 is always counting up) (i.e., when the counter is incremented above the compare value);
- with the next T13 clock (f_{T13}) after a zero-match AND a parallel compare-match.

State Bit **CC63ST is cleared** to 0

- with the next T13 clock (f_{T13}) after a zero-match AND NO parallel compare-match.

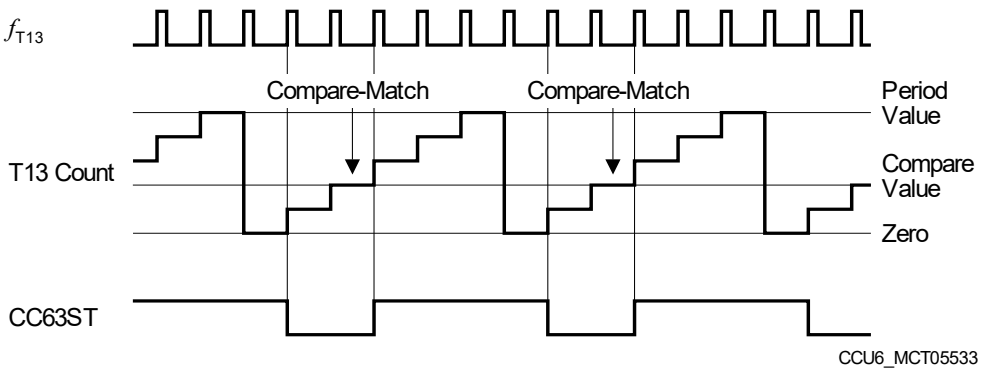


Figure 183 T13 Compare Operation

29.4.4 Compare Mode Output Path

Figure 184 gives an overview on the signal path from the channel State Bit CC63ST to its output pin COUT63. As illustrated, a user can determine the desired output behavior in relation to the current state of CC63ST. Please refer to **Section 29.3.4.3** for detailed information on the output modulation for T12 signals.

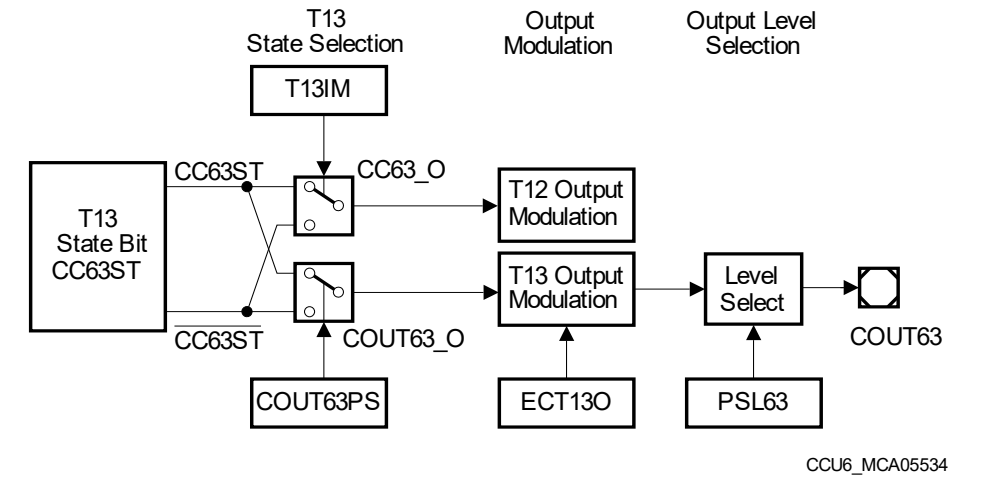


Figure 184 Channel 63 Output Path

The output line COUT63_O can generate a T13 PWM at the output pin COUT63. The signal CC63_O can be used to modulate the T12-related output signals with a T13 PWM. In order to decouple COUT63 from the internal modulation, the compare state leading to an active signal can be selected independently by bits T13IM and COUT63PS.

The last block of the data path is the Output Modulation block. Here, the modulation source T13 and the trap functionality are combined and control the actual level of the output pin COUT63 (see **Figure 185**):

- The **T13 related compare signal** COUT63_O delivered by the T13 state selection with the enable bit **MODCTR**.ECT13O
- The **trap state** TRPS with an individual enable bit **TRPCTR**.TRPEN13

If the modulation input signal COUT63_O is enabled (ECT13O = 1) and is at passive state, the modulated is also in passive state. If the modulation input is not enabled, the output is in passive state.
If the Trap State is active (TRPS = 1), then the output enabled for the trap signal (by TRPEN13 = 1) is set to the passive state.

The output of the modulation control block is connected to a level select block. It offers the option to determine the actual output level of a pin, depending on the state of the output line (decoupling of active/passive state and output polarity) as specified by the Passive State Select bit **PSLR**.PSL63. If the modulated output signal is in the passive state, the level specified directly by PSL63 is output. If it is in the active state, the inverted level of PSL63 is output. This allows the user to adapt the polarity of an active output signal to the connected circuitry.

The PSL63 bit has a shadow register to allow for updates with the T13 shadow transfer signal (T13_ST) without undesired pulses on the output lines. A read action returns the actually used value, whereas a write action targets the shadow bit. Providing a shadow register for the PSL value as well as for other values related to the generation of the PWM signal facilitates a concurrent update by software for all relevant parameters.

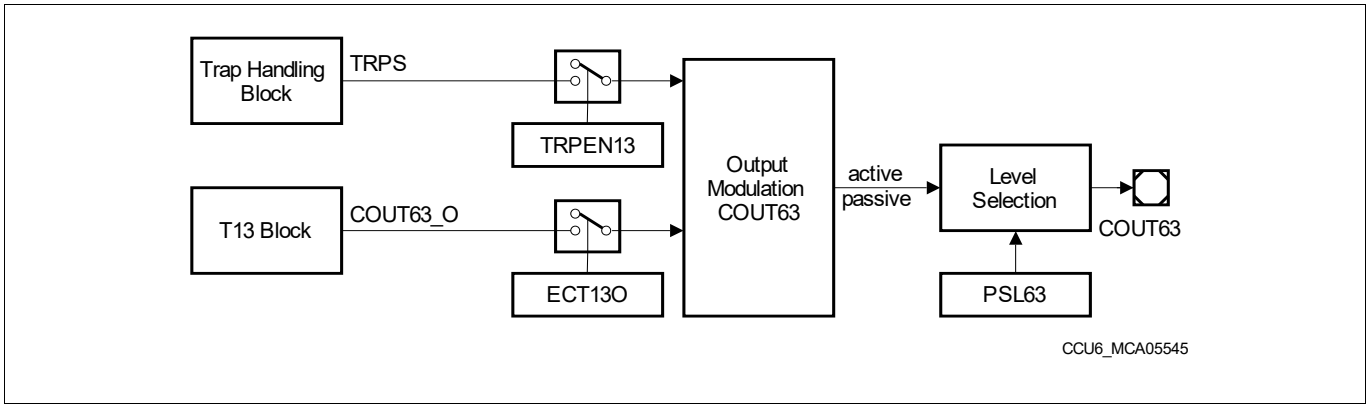


Figure 185 T13 Output Modulation

29.4.5 T13 Shadow Register Transfer

A special shadow transfer signal (T13_ST) can be generated to facilitate updating the period and compare values of the compare channel CC63 synchronously to the operation of T13. Providing a shadow register for values defining one PWM period facilitates a concurrent update by software for all relevant parameters. The next PWM period can run with a new set of parameters. The generation of this signal is requested by software via bit **TCTR0**.STE13 (set by writing 1 to the write-only bit **TCTR4**.T13STR, cleared by writing 1 to the write-only bit **TCTR4**.T13STD).
When signal T13_ST is active, a shadow register transfer is triggered with the next cycle of the T13 clock. Bit STE13 is automatically cleared with the shadow register transfer.
A T13 shadow register transfer takes place (T13_ST active):

- while timer T13 is not running (T13R = 0), or
- STE13 = 1 and a Period-Match is detected while T13R = 1

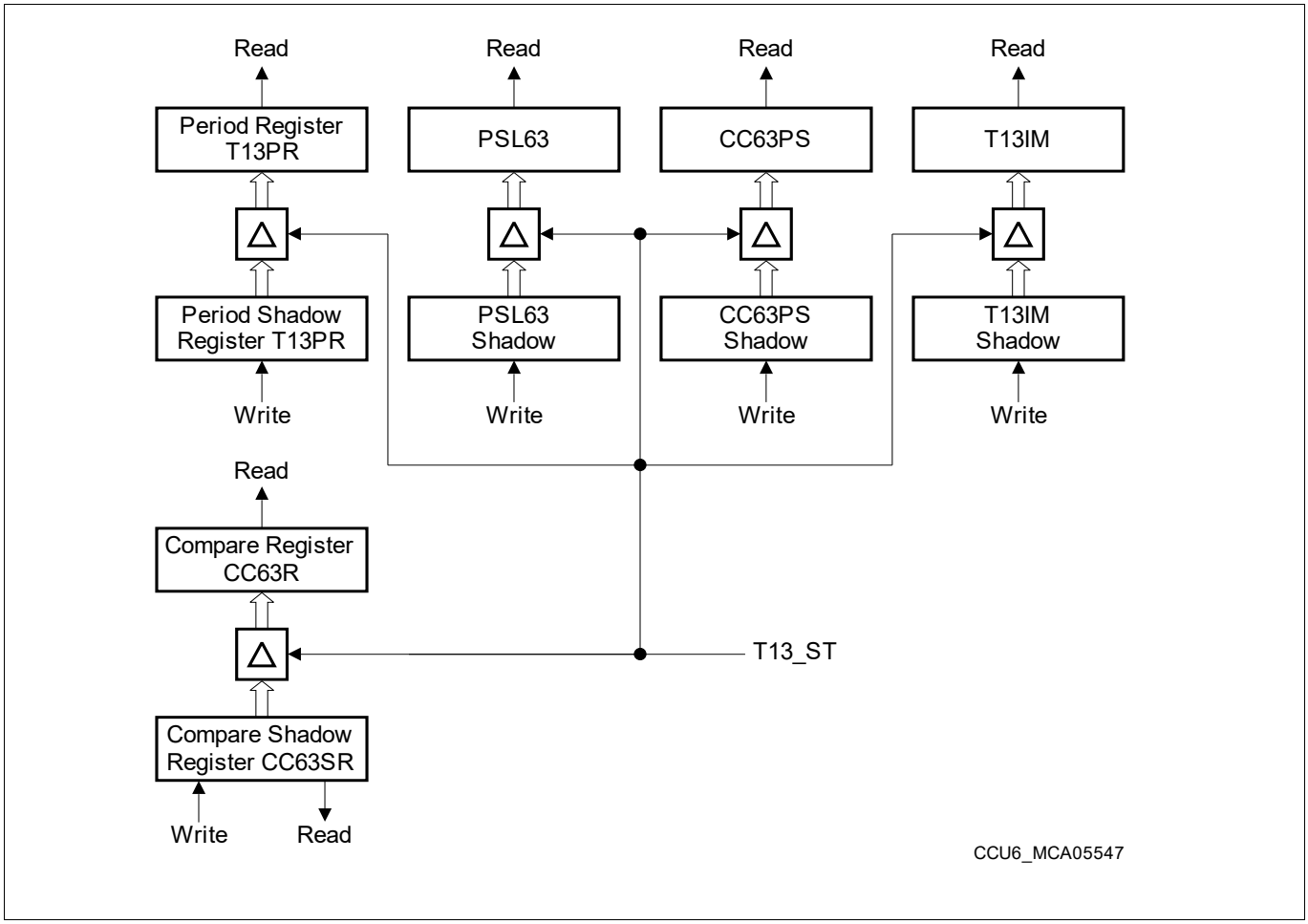


Figure 186 T13 Shadow Register Overview

29.4.6 T13 related Registers

Timer T13 Counter Register

The generation of the patterns for a single channel pulse width modulation (PWM) is based on timer T13. The registers related to timer T13 can be concurrently updated (with well-defined conditions) in order to ensure consistency of the PWM signal. T13 can be synchronized to several timer T12 events. Timer T13 only supports compare mode on its compare channel CC63. Register T13 represents the counting value of timer T13. It can only be written while the timer T13 is stopped. Write actions while T13 is running are not taken into account. Register T13 can always be read by SW. Timer T13 only supports edge-aligned mode (counting up).

Note: While timer T13 is stopped, the internal clock divider is reset in order to ensure reproducible timings and delays.

T13
Timer T13 Counter Register **(0050_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13CV															
rwh															

Field	Bits	Type	Description
T13CV	15:0	rwh	**Timer 13 Counter Value** This register represents the 16-bit counter value of Timer13.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Timer 13 Period Register

Register T13PR contains the period value for timer T13. The period value is compared to the actual counter value of T13 and the resulting counter actions depend on the defined counting rules. This register has a shadow register and the shadow transfer is controlled by bit STE13. A read action by SW delivers the value currently used for the compare action, whereas the write action targets a shadow register. The shadow register structure allows a concurrent update of all T13-related values.

T13PR
Timer 13 Period Register **(0054_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13PV															
rwh															

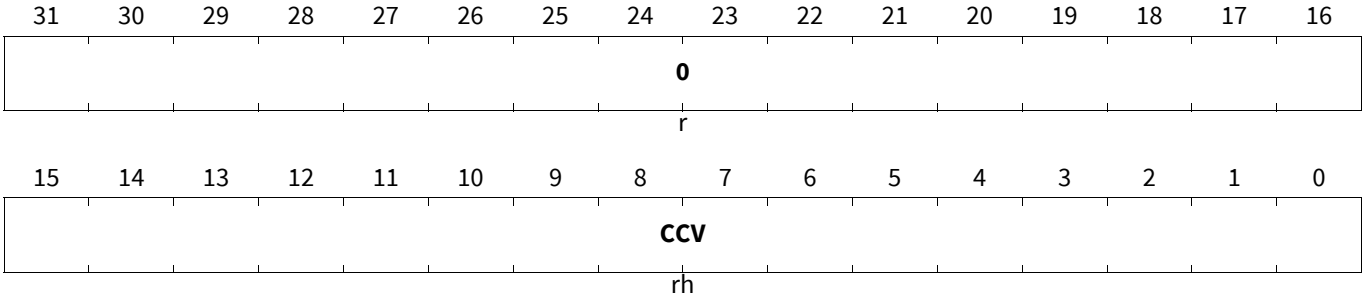
Field	Bits	Type	Description
T13PV	15:0	rwh	**T13 Period Value** The value T13PV defines the counter value for T13 leading to a period-match. When reaching this value, the timer T13 is set to zero.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Compare Register for T13

Registers CC63R is the actual compare register for T13. The values stored in CC63R is compared to the counter value of T13. The State Bit CC63ST is located in register **CMPSTAT**.

CC63R

Compare Register for T13 **(0058_H)** **Application Reset Value: 0000 0000$_H$**



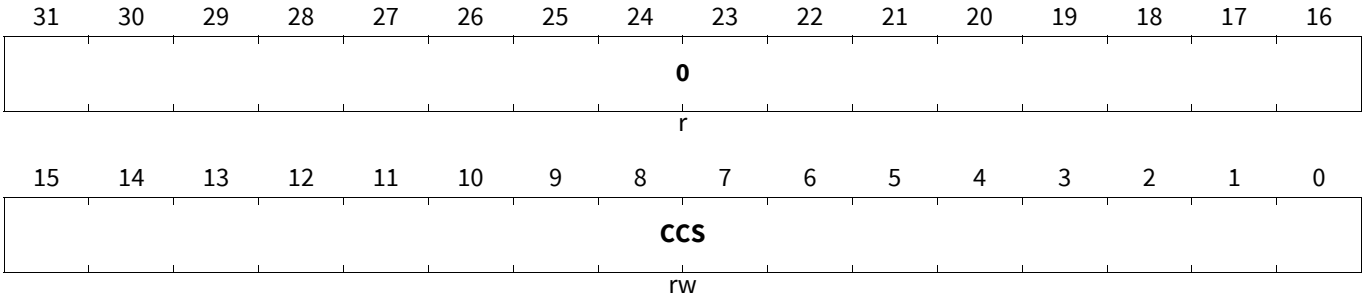
Field	Bits	Type	Description
CCV	15:0	rh	**Channel CC63 Compare Value** The bit field CCV contains the value, that is compared to the T13 counter value.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Compare Shadow Register for T13

The register CC63R can only be read by SW, the modification of the value is done by a shadow register transfer from register CC63SR. The corresponding shadow register CC63SR can be read and written by SW.

CC63SR

Compare Shadow Register for T13 **($005C_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CCS	15:0	rw	**Shadow Register for Channel CC63 Compare Value** The bit field contents of CCS is transferred to the bit field CCV during a shadow transfer.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

29.5 Synchronous Start Feature

Synchronous start of the capture/compare timers is supported by control bit SYSCON.CCTRIG0 in the SCU module. Bit SYSCON.CCTRIG0 is connected to the T12HR and T13HR inputs of the CCU60 and CCU61 kernels, so all timers T12 and T13 can be started synchronously.

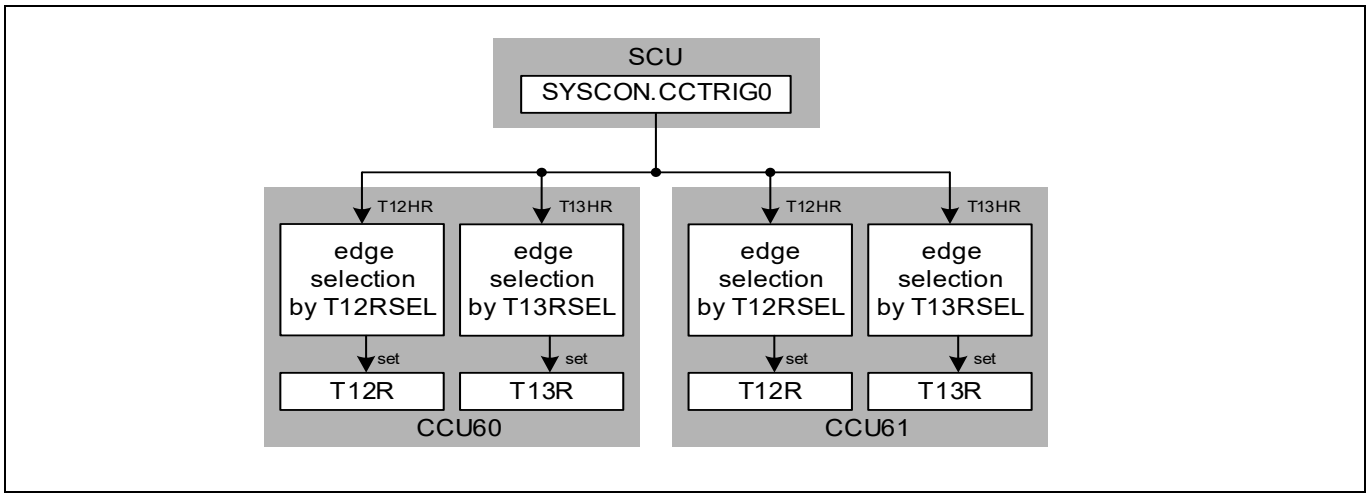


Figure 187 Synchronization Concept

29.6 Trap Handling

The trap functionality permits the PWM outputs to react on the state of the input signal $\overline{\text{CTRAP}}$. This functionality can be used to switch off the power devices if the trap input becomes active (e.g. to perform an emergency stop). The trap handling and the effect on the output modulation are controlled by the bits in the trap control register **TRPCTR**. The trap flags TRPF and TRPS are located in register **IS** and can be set/cleared by SW by writing to registers **ISS** and **ISR**.
Figure 188 gives an overview on the trap function.

The Trap Flag TRPF monitors the trap input and initiates the entry into the Trap State. The Trap State Bit TRPS determines the effect on the outputs and controls the exit of the Trap State.
When a trap condition is detected ($\overline{\text{CTRAP}}$ = 0) and the input is enabled (TRPPEN = 1), both, the Trap Flag TRPF and the Trap State Bit TRPS, are set to 1 (trap state active).
The output of the Trap State Bit TRPS leads to the Output Modulation Blocks (for T12 and for T13) and can there deactivate the outputs (set them to the passive state). Individual enable control bits for each of the six T12-related outputs and the T13-related output facilitate a flexible adaptation to the application needs.

There are a number of different ways to exit the Trap State. This offers SW the option to select the best operation for the application. Exiting the Trap State can be done either immediately when the trap condition is removed ($\overline{\text{CTRAP}}$ = 1 or TRPPEN = 0), or under software control, or synchronously to the PWM generated by either Timer T12 or Timer T13.

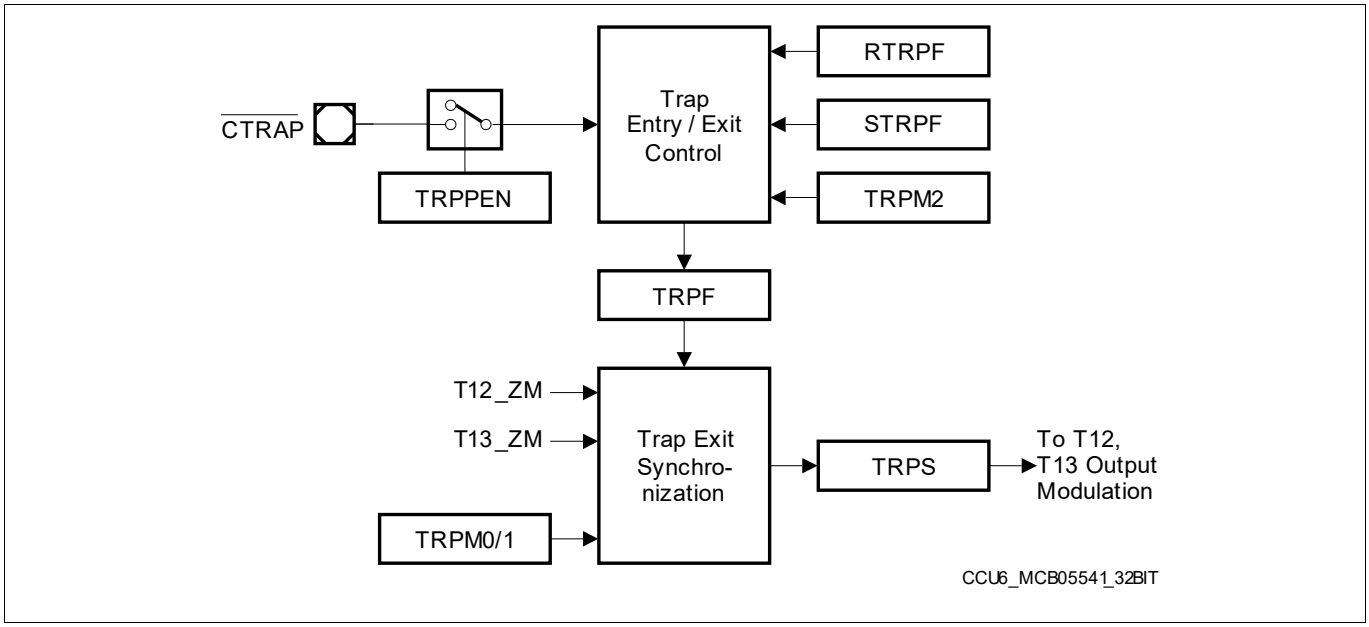


Figure 188 Trap Logic Block Diagram

Clearing of TRPF is controlled by the mode control bit TRPM2. If TRPM2 = 0, TRPF is automatically cleared by HW when $\overline{\text{CTRAP}}$ returns to the inactive level ($\overline{\text{CTRAP}}$ = 1) or if the trap input is disabled (TRPPEN = 0). When TRPM2 = 1, TRPF must be reset by SW after $\overline{\text{CTRAP}}$ has become inactive.

Clearing of TRPS is controlled by the mode control bits TRPM1 and TRPM0 (located in the Trap Control Register TRPCTR). A reset of TRPS terminates the Trap State and returns to normal operation. There are three options selected by TRPM1 and TRPM0. One is that the Trap State is left immediately when the Trap Flag TRPF is cleared, without any synchronization to timers T12 or T13. The other two options facilitate the synchronization of the termination of the Trap State to the count periods of either Timer T12 or Timer T13. **Figure 189** gives an overview on the associated operation.

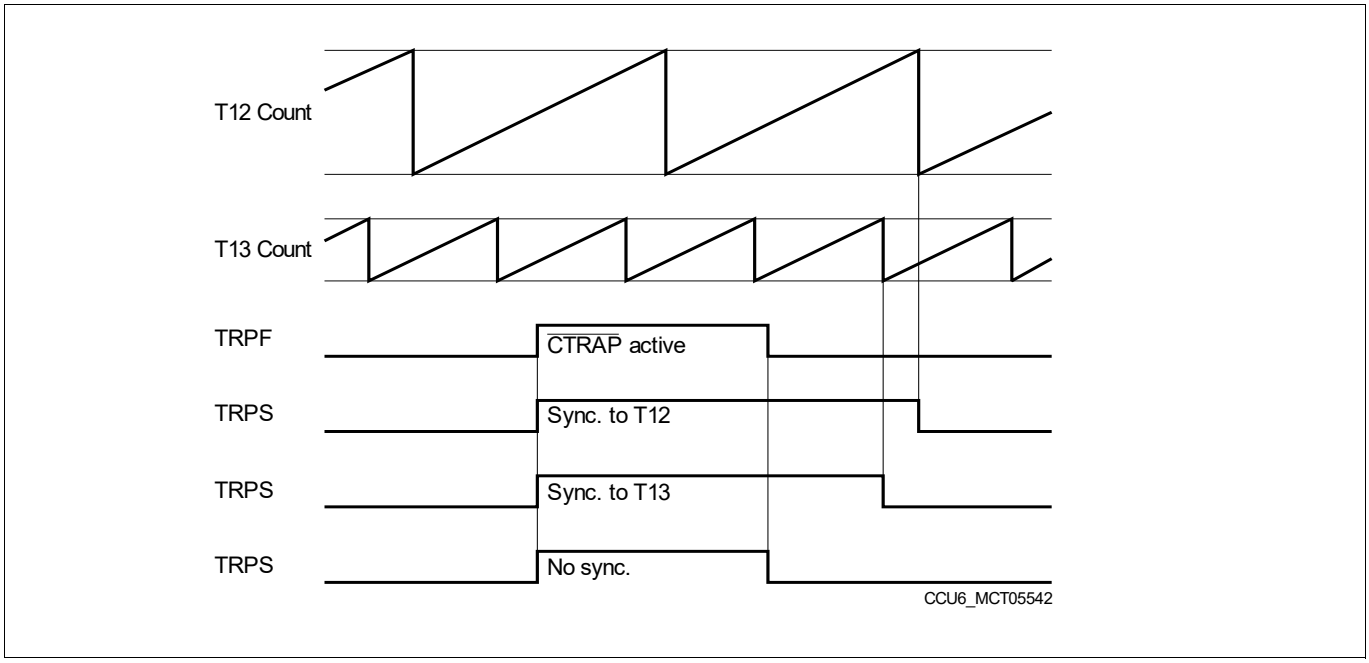


Figure 189 Trap State Synchronization (with TRPM2 = 0)

29.7 Multi-Channel Mode

The Multi-Channel mode offers the possibility to modulate all six T12-related output signals with one instruction. The bits in bit field **MCMOUT**.MCMP are used to specify the outputs that may become active. If Multi-Channel mode is enabled (bit **MODCTR**.MCMEN = 1), only those outputs may become active, that have a 1 at the corresponding bit position in bit field MCMP.
This bit field has its own shadow bit field **MCMOUTS**.MCMPS, that can be written by software. The transfer of the new value in MCMPS to the bit field MCMP can be triggered by, and synchronized to, T12 or T13 events. This structure permits the software to write the new value, that is then taken into account by the hardware at a well-defined moment and synchronized to a PWM signal. This avoids unintended pulses due to unsynchronized modulation sources.

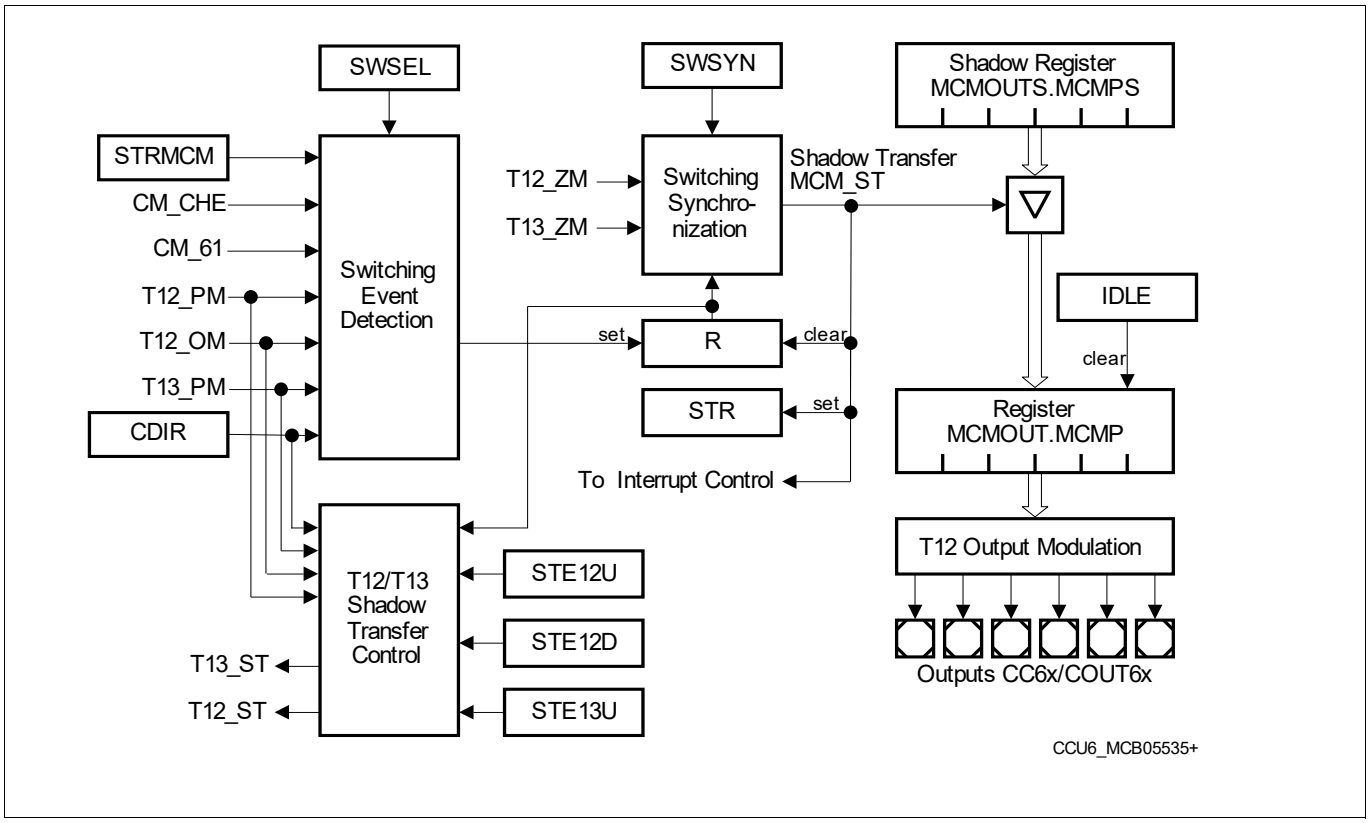


Figure 190 Multi-Channel Mode Block Diagram

Figure 190 shows the functional blocks for the Multi-Channel operation, controlled by bit fields in register **MCMCTR**. The event that triggers the update of bit field MCMP is chosen by SWSEL. In order to synchronize the update of MCMP to a PWM generated by T12 or T13, bit field SWSYN allows the selection of the synchronization event leading to the transfer from MCMPS to MCMP. Due to this structure, an update takes place with a new PWM period. A reminder flag R is set when the selected switching event occurs (the event is not necessarily synchronous to the modulating PWM), and is cleared when the transfer takes place. This flag can be monitored by software to check for the status of this logic block. If the shadow transfer from MCMPS to MCMP takes place, bit **IS**.STR becomes set and an interrupt can be generated.
In addition to the Multi-Channel shadow transfer event MCM_ST, the shadow transfers for T12 (T12_ST) and T13 (T13_ST) can be generated to allow concurrent updates of applied duty cycles for T12 and/or T13 modulation and Multi-Channel patterns.

If it is explicitly desired, the update takes place immediately with the occurrence of the selected event when the direct synchronization mode is selected. The update can also be requested by software by writing to bit field MCMPS with the shadow transfer request bit STRMCM = 1. The option to trigger an update by SW is possible for

all settings of SWSEL.
By using the direct mode and bit STRMCM = 1, the update takes place completely under software control.

Table 199 Multi-Channel Mode Switching Event Selection

SWSEL	Selected Event (see register MCMCTR)
000_B	No automatic event detection
001_B	Correct Hall Event (CM_CHE) detected at input signals CCPOSx without additional delay
010_B	T13 Period-Match (T13_PM)
011_B	T12 One-Match while counting down (T12_OM and CDIR = 1)
100_B	T12 Compare Channel 1 Event while counting up (CM_61 and CDIR = 0) to support the phase delay function by CC61 for block commutation mode.
101_B	T12 Period-Match while counting up (T12_PM and CDIR = 0)
110_B, 111_B	Reserved, no action

Table 200 Multi-Channel Mode Switching Synchronization

SWSYN	Synchronization Event (see register MCMCTR)
00_B	Direct Mode: the trigger event directly causes the shadow transfer
01_B	T13 Zero-Match (T13_ZM), the MCM shadow transfer is synchronized to a T13 PWM
10_B	T12 Zero-Match (T12_ZM), the MCM shadow transfer is synchronized to a T12 PWM
11_B	Reserved, no action

29.8 Hall Sensor Mode

For Brushless DC-Motors in block commutation mode, the Multi-Channel Mode has been introduced to provide efficient means for switching pattern generation. These patterns need to be output in relation to the angular position of the motor. For this, usually Hall sensors or Back-EMF sensing are used to determine the angular rotor position. The CCU6 provides three inputs, CCPOS0, CCPOS1, and CCPOS2, that can be used as inputs for the Hall sensors or the Back-EMF detection signals.

There is a strong correlation between the motor position and the output modulation pattern. When a certain position of the motor has been reached, indicated by the sampled Hall sensor inputs (the Hall pattern), the next, pre-determined Multi-Channel Modulation pattern has to be output. Because of different machine types, the modulation pattern for driving the motor can vary. Therefore, it is wishful to have a wide flexibility in defining the correlation between the Hall pattern and the corresponding Modulation pattern. Furthermore, a hardware mechanism significantly reduces the CPU load for block-commutation.

The CCU6 offers the flexibility by having a register containing the currently assumed Hall pattern (CURH), the next expected Hall pattern (EXPH) and the corresponding output pattern (MCMP). A new Modulation pattern is output when the sampled Hall inputs match the expected ones (EXPH). To detect the next rotation phase (segment for block commutation), the CCU6 monitors the Hall inputs for changes. When the next expected Hall pattern is detected, the next corresponding Modulation pattern is output.

To increase for noise immunity (to a certain extend), the CCU6 offers the possibility to introduce a sampling delay for the Hall inputs. Some changes of the Hall inputs are not leading to the expected Hall pattern, because they are only short spikes due to noise. The Hall pattern compare logic compares the Hall inputs to the next expected pattern and also to the currently assumed pattern to filter out spikes.

For the Hall and Modulation output patterns, a double-register structure is implemented. While register **MCMOUT** holds the actually used values, its shadow register **MCMOUTS** can be loaded by software from a pre-defined table, holding the appropriate Hall and Modulation patterns for the given motor control.
A transfer from the shadow register into register MCMOUT can take place when a correct Hall pattern change is detected. Software can then load the next values into register MCMOUTS. It is also possible by software to force a transfer from MCMOUTS into MCMOUT.

Note: The Hall input signals CCPOSx and the CURH and EXPH bit fields are arranged in the following order:
CCPOS0 corresponds to CURH.0 (LSB) and EXPH.0 (LSB)
CCPOS1 corresponds to CURH.1 and EXPH.1
CCPOS2 corresponds to CURH.2 (MSB) and EXPH.2 (MSB)

29.8.1 Hall Pattern Evaluation

The Hall sensor inputs CCPOSx can be permanently monitored via an edge detection block (with the module clock f_{CC6}). In order to suppress spikes on the Hall inputs due to noise in rugged inverter environment, two optional noise filtering methods are supported by the Hall logic (both methods can be combined).

- Noise filtering with delay:
  For this function, the mode control bit fields MSEL6x for all T12 compare channels must be programmed to 1000_B and DBYP = 0. The selected event triggers Dead-Time Counter 0 to generate a programmable delay (defined by bit field DTM). When the delay has elapsed, the evaluation signal HCRDY becomes activated. Output modulation with T12 PWM signals is not possible in this mode.
- Noise filtering by synchronization to PWM:
  The Hall inputs are not permanently monitored by the edge detection block, but samples are taken only at defined points in time during a PWM period. This can be used to sample the Hall inputs when the switching noise (due to PWM) does not disturb the Hall input signals.

If neither the delay function of Dead-Time Counter 0 is not used for the Hall pattern evaluation nor the Hall mode for Brushless DC-Drive control is enabled, the timer T12 block is available for PWM generation and output modulation.

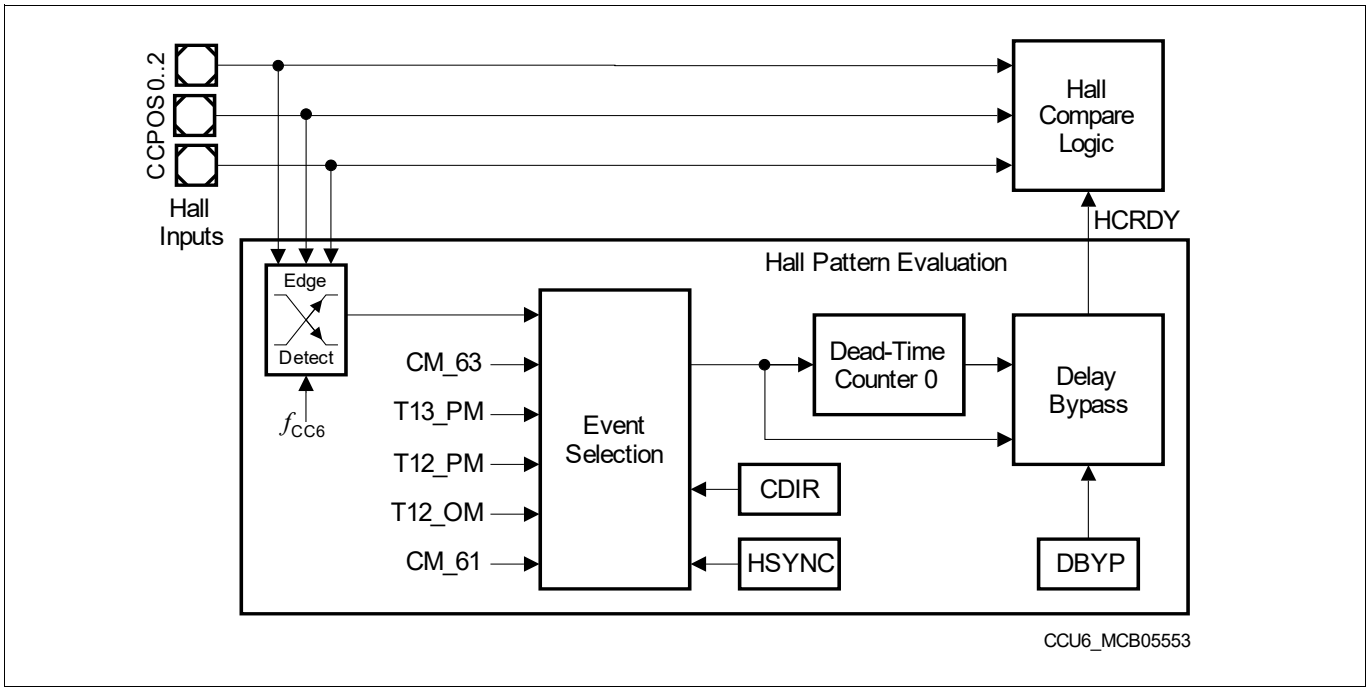


Figure 191 Hall Pattern Evaluation

If the evaluation signal HCRDY (Hall Compare Ready, see **Figure 192**) becomes activated, the Hall inputs are sampled and the Hall compare logic starts the evaluation of the Hall inputs.

Figure 191 illustrates the events for Hall pattern evaluation and the noise filter logic, **Table 201** summarizes the selectable trigger input signals.

Table 201 Hall Sensor Mode Trigger Event Selection

HSYNC	Selected Event (see register T12MSEL)
000_B	Any edge at any of the inputs CCPOSx, independent from any PWM signal (permanent check).
001_B	A T13 Compare-Match (CM_63).
010_B	A T13 Period-Match (T13_PM).
011_B	Hall sampling triggered by HW sources is switched off.
100_B	A T12 Period-Match while counting up (T12_PM and CDIR = 0).
101_B	A T12 One-Match while counting down (T12_OM and CDIR = 1).
110_B	A T12 Compare-Match of compare channel CC61 while counting up (CM_61 and CDIR = 0).
111_B	A T12 Compare-Match of compare channel CC61 while counting down (CM_61 and CDIR = 1).

29.8.2 Hall Pattern Compare Logic

Figure 192 gives an overview on the double-register structure and the pattern compare logic. Software writes the next modulation pattern (MCMPS) and the corresponding current (CURHS) and expected (EXPHS) Hall patterns into the shadow register MCMOUTS. Register MCMOUT holds the actually used values CURH and EXPH. The modulation pattern MCMP is provided to the T12 Output Modulation block. The current (CURH) and expected (EXPH) Hall patterns are compared to the sampled Hall sensor inputs (visible in register **CMPSTAT**). Sampling of the inputs and the evaluation of the comparator outputs is triggered by the evaluation signal HCRDY (Hall Compare Ready), that is detailed in the next section.

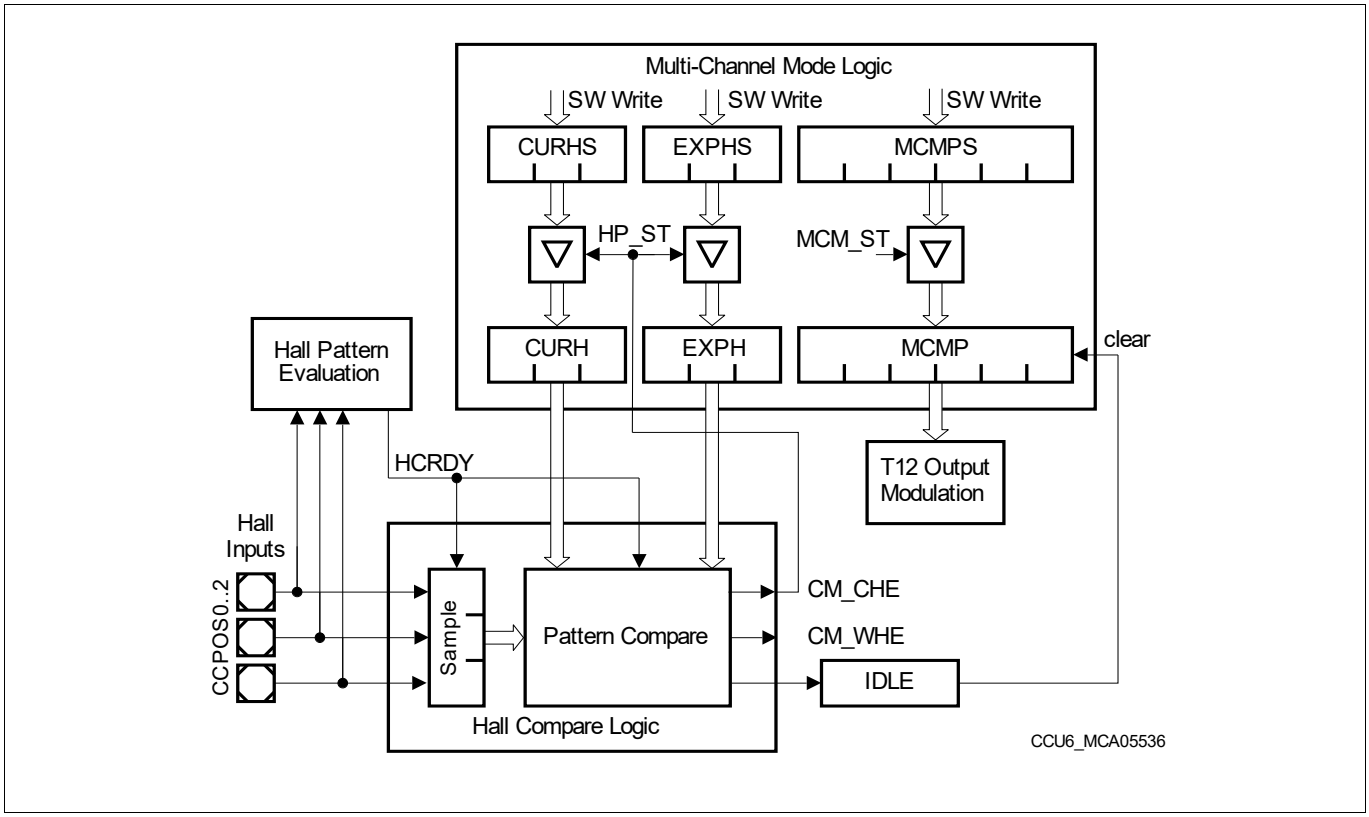


Figure 192 Hall Pattern Compare Logic

- If the sampled Hall pattern matches the value programmed in CURH, the detected transition was a spike (no Hall event) and no further actions are necessary.
- If the sampled Hall pattern matches the value programmed in EXPH, the detected transition was the expected event (correct Hall event CM_CHE) and the MCMP value has to change.
- If the sampled Hall pattern matches neither CURH nor EXPH, the transition was due to a major error (wrong Hall event CM_CWE) and can lead to an emergency shut down (IDLE).

At every correct Hall event (CM_CHE), the next Hall patterns are transferred from the shadow register MCMOUTS into MCMOUT (Hall pattern shadow transfer HP_ST), and a new Hall pattern with its corresponding output pattern can be loaded (e.g. from a predefined table in memory) by software into MCMOUTS. For the Modulation patterns, signal MCM_ST is used to trigger the transfer.
Loading this register can also be done by writing MCMOUTS.STRHP = 1 (for EXPH and CURH) or MCMOUTS.STRMCMP = 1 (for MCMP).

Note: If in a corner case a hardware event occurs simultaneously with a software write where MCMOUTS.STRHP = 1 or MCMOUTS.STRMCMP = 1, the current contents of MCMOUTS is copied to the corresponding bit fields of MCMOUT. The new value written to MCMOUTS will be loaded upon the next event.

29.8.3 Hall Mode Flags

Depending on the Hall pattern compare operation, a number of flags are set in order to indicate the status of the module and to trigger further actions and interrupt requests.

Flag **IS**.CHE (Correct Hall Event) is set by signal CM_CHE when the sampled Hall pattern matches the expected one (EXPH). This flag can also be set by SW by setting bit **ISS**.SCHE = 1. If enabled by bit **IEN**.ENCHE = 1, the set signal for CHE can also generate an interrupt request to the CPU. Bit field **INP**.INPCHE defines which service

request output becomes activated in case of an interrupt request.To clear flag CHE, SW needs to write **ISR**.RCHE = 1.
Flag IS.WHE indicates a Wrong Hall Event. Its handling for flag setting and resetting as well as interrupt request generation are similar to the mechanism for flag CHE.
The implementation of flag STR is done in the same way as for CHE and WHE. This flag is set by HW by the shadow transfer signal MCM_ST (see also **Figure 190**).

Please note that for flags CHE, WHE, and STR, the interrupt request generation is triggered by the set signal for the flag. That means, a request can be generated even if the flag is already set. There is no need to clear the flag in order to enable further interrupt requests.

The implementation for the IDLE flag is different. It is set by HW through signal CM_WHE if enabled by bit ENIDLE. Software can also set the flag via bit SIDLE. As long as bit IDLE is set, the modulation pattern field MCMP is cleared to force the outputs to the passive state. Flag IDLE must be cleared by software by writing RIDLE = 1 in order to return to normal operation. To fully restart from IDLE mode, the transfer requests for the bit fields in register MCMOUTS to register MCMOUT have to be initiated by software via bits STRMCM and STRHP in register MCMOUTS. In this way, the release from IDLE mode is under software control, but can be performed synchronously to the PWM signal.

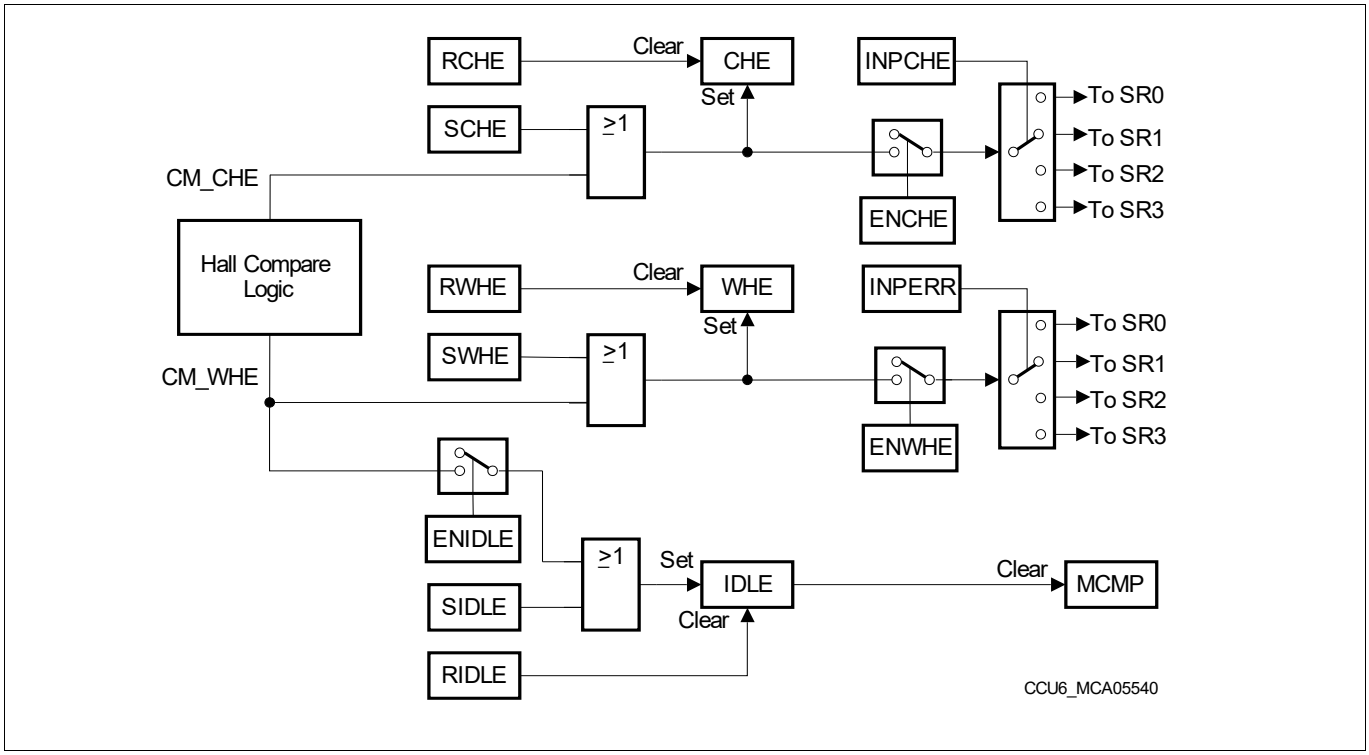


Figure 193 Hall Mode Flags

29.8.4 Hall Mode for Brushless DC-Motor Control

The CCU6 provides a mode for the Timer T12 Block especially targeted for convenient control of block commutation patterns for Brushless DC-Motors. This mode is selected by setting all **T12MSEL**.MSEL6x bit fields of the three T12 Channels to 1000_B.
In this mode, illustrated in **Figure 194**, channel CC60 is placed in capture mode to measure the time elapsed between the last two correct Hall events, channel CC61 in compare mode to provide a programmable phase delay between the Hall event and the application of a new PWM output pattern, and channel CC62 also in compare mode as first time-out criterion. A second time-out criterion can be built by the T12 period match event.

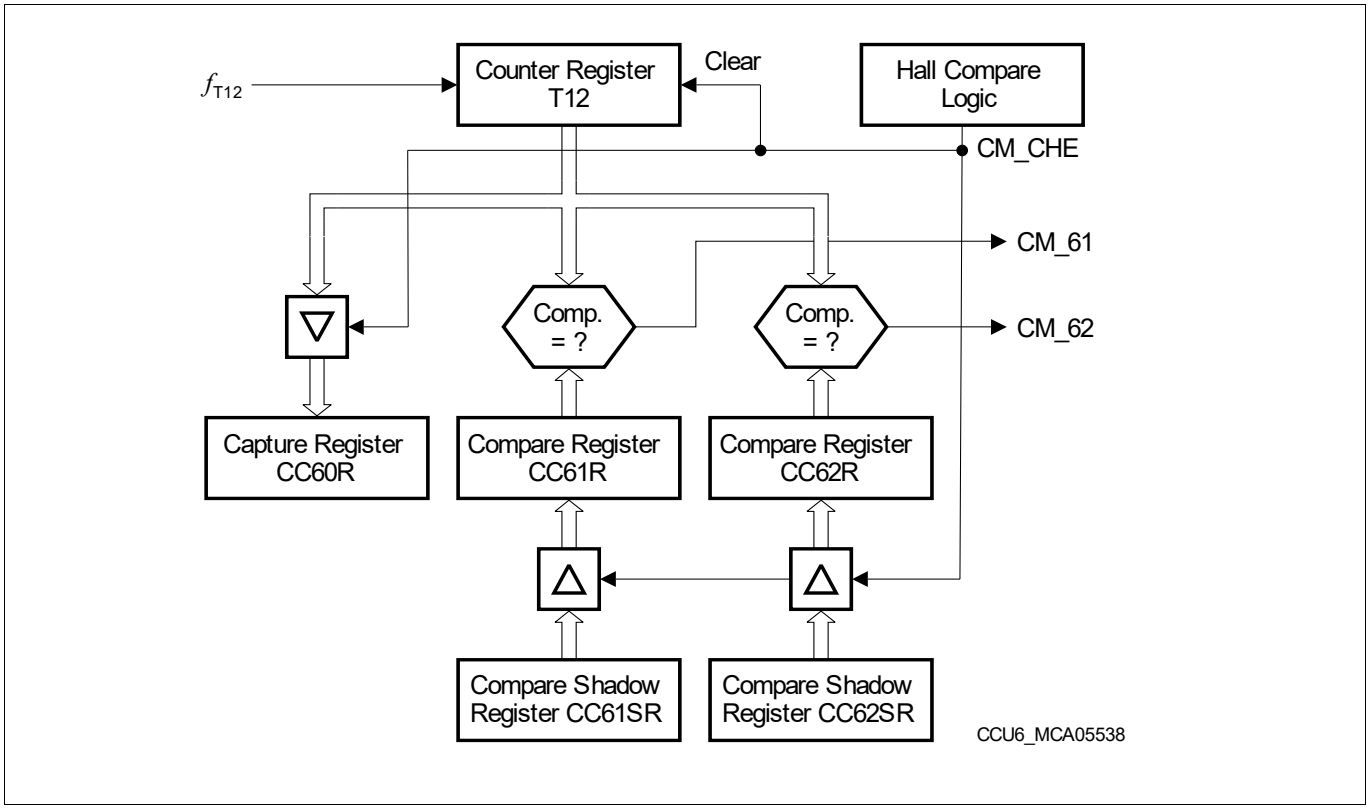


Figure 194 T12 Block in Hall Sensor Mode

The signal CM_CHE from the Hall compare logic is used to transfer the new compare values from the shadow registers CC6xSR into the actual compare registers CC6xR, performs the shadow transfer for the T12 period register, to capture the current T12 contents into register CC60R, and to clear T12.

Note: In this mode, the shadow transfer signal T12_ST is not generated. Not all shadow bits, such as the PSLy bits, will be transferred to their main registers. To program the main registers, SW needs to write to these registers while Timer T12 is stopped. In this case, a SW write actualizes both registers.

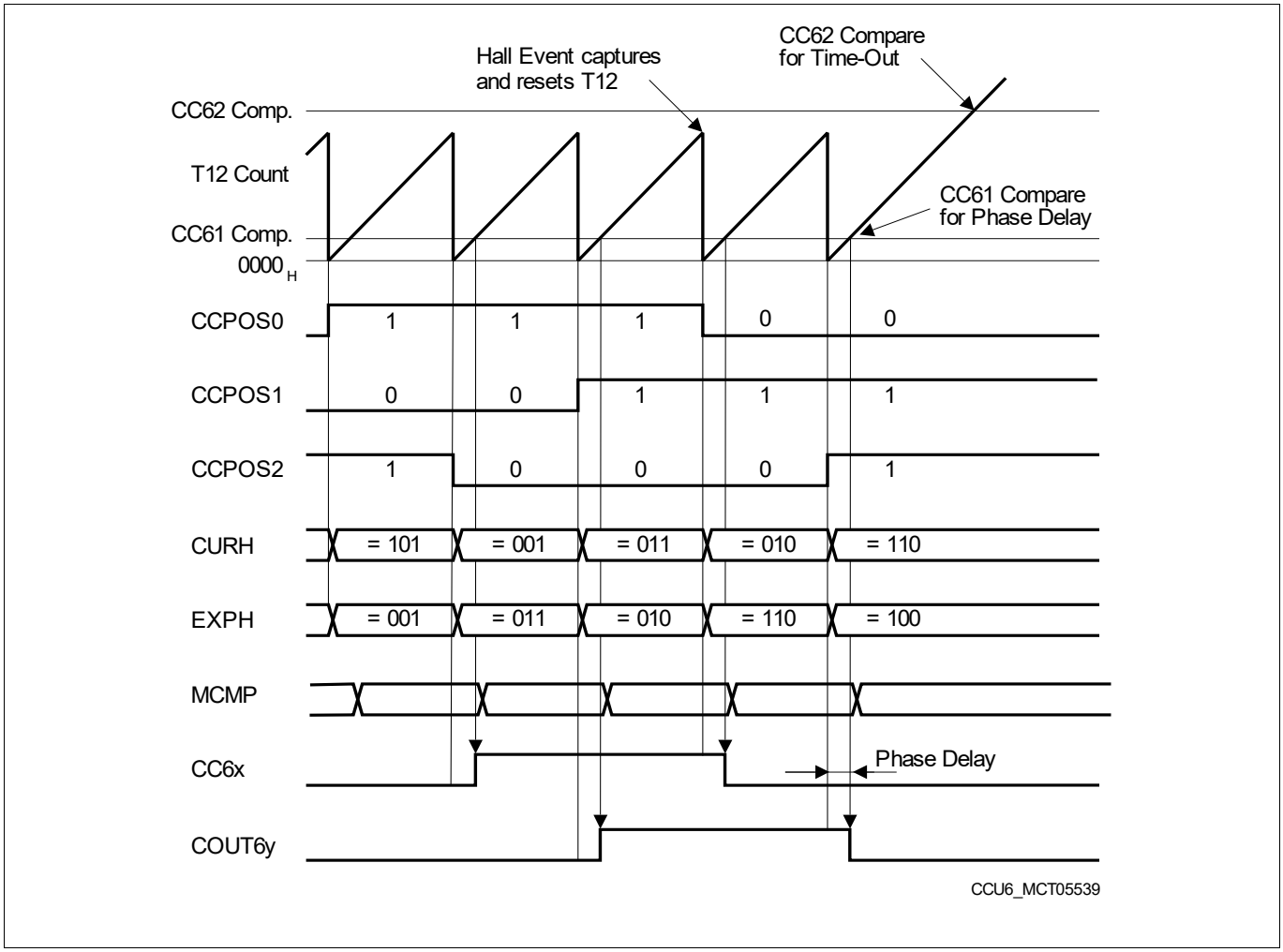


Figure 195 Brushless DC-Motor Control Example (all MSEL6x = 1000_B)

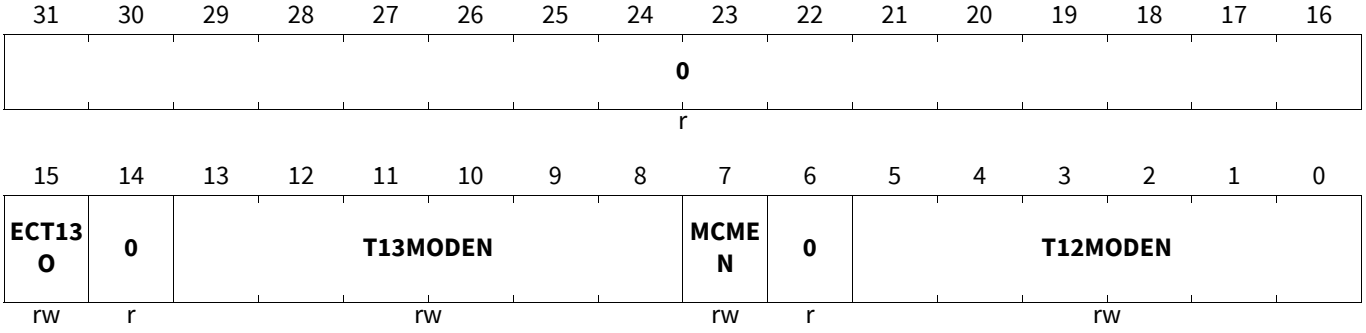
After the detection of an expected Hall pattern (CM_CHE active), the T12 count value is captured into channel CC60 (representing the actual rotor speed by measuring the elapsed time between the last two correct Hall events), and T12 is reset. When the timer reaches the compare value in channel CC61, the next multi-channel state is switched by triggering the shadow transfer of bit field MCMP (if enabled in bit field **SWEN**). This trigger event can be combined with the synchronization of the next multi-channel state to the PWM source (to avoid spikes on the output lines, see **Section 29.7**). This compare function of channel CC61 can be used as a phase delay from the position sensor input signals to the switching of the output signals, that is necessary if a sensorless back-EMF technique or Hall sensors are used. The compare value in channel CC62 can be used as a time-out trigger (interrupt), indicating that the actual motor speed is far below the desired destination value. An abnormal load change can be detected with this feature and PWM generation can be disabled.

29.9 Modulation Control Registers

Modulation Control Register

This register contains bits enabling the modulation of the corresponding output signal by PWM pattern generated by the timers T12 and T13. Furthermore, the multi-channel mode can be enabled as additional modulation source for the output signals.

MODCTR
Modulation Control Register **(0080_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T12MODEN	5:0	rw	**T12 Modulation Enable** These bits enable the modulation of the corresponding output signal by a PWM pattern generated by timer T12. T12MODEN0 = MODCTR.0 for output CC60 T12MODEN1 = MODCTR.1 for output COUT60 T12MODEN2 = MODCTR.2 for output CC61 T12MODEN3 = MODCTR.3 for output COUT61 T12MODEN4 = MODCTR.4 for output CC62 T12MODEN5 = MODCTR.5 for output COUT62 00_H The modulation of the corresponding output signal by a T12 PWM pattern is disabled. 01_H The modulation of the corresponding output signal by a T12 PWM pattern is enabled.
MCMEN	7	rw	**Multi-Channel Mode Enable** 0_B The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMOUT is disabled. 1_B The modulation of the corresponding output signal by a multi-channel pattern according to bit field MCMOUT is enabled.

Field	Bits	Type	Description
T13MODEN	13:8	rw	**T13 Modulation Enable** These bits enable the modulation of the corresponding output signal by the PWM pattern CC63_O generated by timer T13. T13MODEN0 = MODCTR.8 for output CC60 T13MODEN1 = MODCTR.9 for output COUT60 T13MODEN2 = MODCTR.10 for output CC61 T13MODEN3 = MODCTR.11 for output COUT61 T13MODEN4 = MODCTR.12 for output CC62 T13MODEN5 = MODCTR.13 for output COUT62 00_H The modulation of the corresponding output signal by a T13 PWM pattern is disabled. 01_H The modulation of the corresponding output signal by a T13 PWM pattern is enabled.
ECT13O	15	rw	**Enable Compare Timer T13 Output** 0_B The output COUT63 is in the passive state. 1_B The output COUT63 is enabled for the PWM signal generated by T13.
0	6, 14, 31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Trap Control Register

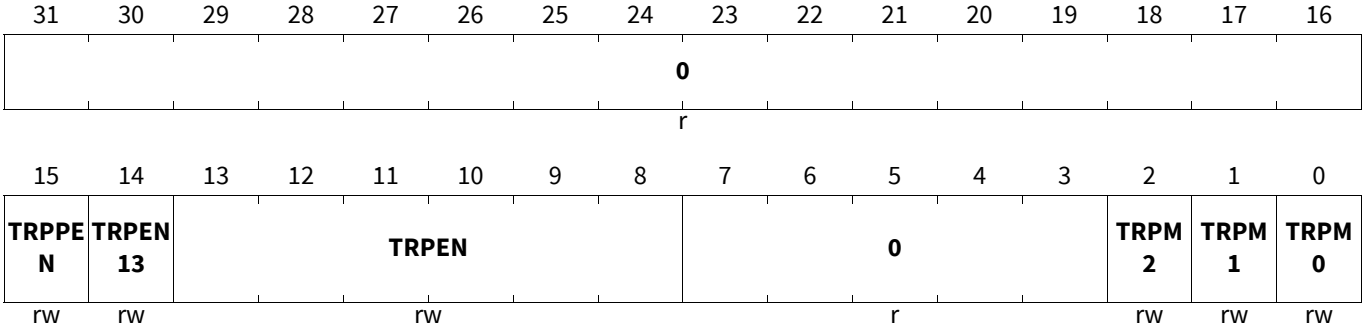
The register TRPCTR controls the trap functionality. It contains independent enable bits for each output signal and control bits to select the behavior in case of a trap condition. The trap condition is a low level on the $\overline{\text{CTRAP}}$ input pin, that is monitored (inverted level) by bit IS.TRPF. While TRPF=1 (trap input active), the trap state bit IS.TRPS is set to 1.

The behavior resulting from the combination [TRPM1, TRPM0] of the Trap Mode Control bits TRPM1, TRPM0 is described in **Table 202**.

Table 202 Leaving the Trap State

[TRPM1, TRPM0]	Function
00_B	The trap state is left (return to normal operation) after TRPF has become 0 again when a zero-match of T12 (while counting up) is detected (synchronization to T12).
01_B	The trap state is left (return to normal operation) after TRPF has become 0 again when a zero-match of T13 is detected (synchronization to T13).
10_B	Reserved
11_B	The trap state is left (return to normal operation) immediately after TRPF has become 0 again without any synchronization to T12 or T13.

TRPCTR
Trap Control Register **(0084_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TRPM0	0	rw	**Trap Mode Control Bit 0** Together with bit TRPM1, these two bits define the behavior of the selected outputs when leaving the trap state after the trap condition has become inactive again. A synchronization to the timer driving the PWM pattern avoids unintended pulses when leaving the trap state. The behavior resulting from the combination [TRPM1, TRPM0] is described in **Table 202**.
TRPM1	1	rw	**Trap Mode Control Bit 1** Together with bit TRPM0, these two bits define the behavior of the selected outputs when leaving the trap state after the trap condition has become inactive again. A synchronization to the timer driving the PWM pattern avoids unintended pulses when leaving the trap state. The behavior resulting from the combination [TRPM1, TRPM0] is described in **Table 202**.
TRPM2	2	rw	**Trap Mode Control Bit 2** This bit defines how the trap flag TRPF can be cleared after the trap input condition ($\overline{CTRAP}$ = 0 and TRPPEN = 1) is no longer valid (either by $\overline{CTRAP}$ = 1 or by TRPPEN = 0). 0_B Automatic Mode: Bit TRPF is cleared by HW if the trap input condition is no longer valid. 1_B Manual Mode: Bit TRPF stays 1 after the trap input condition is no longer valid. It has to be cleared by SW by writing ISR.RTRPF = 1.

Field	Bits	Type	Description
TRPEN	13:8	rw	**Trap Enable Control** Setting a bit enables the trap functionality for the following corresponding output signals: TRPEN0 = TRPCTR.8 for output CC60 TRPEN1 = TRPCTR.9 for output COUT60 TRPEN2 = TRPCTR.10 for output CC61 TRPEN3 = TRPCTR.11 for output COUT61 TRPEN4 = TRPCTR.12 for output CC62 TRPEN5 = TRPCTR.13 for output COUT62 00_H The trap functionality of the corresponding output signal is disabled. The output state is independent from bit IS.TRPS. 01_H The trap functionality of the corresponding output signal is enabled. The output state is set to the passive while IS.TRPS=1.
TRPEN13	14	rw	**Trap Enable Control for Timer T13** 0_B The trap functionality for output COUT63 is disabled. The output state is independent from bit IS.TRPS. 1_B The trap functionality for output COUT63 is enabled. The output state is set to the passive while IS.TRPS=1.
TRPPEN	15	rw	**Trap Pin Enable** This bit enables the input (pin) function for the trap generation. An interrupt can only be generated if a falling edge is detected at pin CTRAP while TRPPEN = 1. 0_B The CCU6 trap functionality based on the input $\overline{\text{CTRAP}}$ is disabled. A CCU6 trap can only be generated by SW by setting bit TRPF. 1_B The CCU6 trap functionality based on the input $\overline{\text{CTRAP}}$ is enabled. A CCU6 trap can be generated by SW by setting bit TRPF or by $\overline{\text{CTRAP}}$=0.
0	7:3, 31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Passive State Level Register

Register PSLR defines the passive state level of the PWM outputs of the module. The passive state level is the value that is driven during the passive state of the output. During the active state, the corresponding output pin drives the active state level, that is the inverted passive state level. The passive state level permits to adapt the driven output levels to the driver polarity (inverted, not inverted) of the connected power stage. The bits in this register have shadow bit fields to permit a concurrent update of all PWM-related parameters (bit field PSL is updated with T12_ST, whereas PSL63 is updated with T13_ST). The actually used values can be read (attribute “rh”), whereas the shadow bits can only be written (attribute “w”).

PSLR

Passive State Level Register **(0088$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								PSL63	0	PSL					
r								rwh	r	rwh					

Field	Bits	Type	Description
PSL	5:0	rwh	**Compare Outputs Passive State Level** These bits define the passive level driven by the module outputs during the passive state. PSL0 = PSLR.0 for output CC60 PSL1 = PSLR.1 for output COUT60 PSL2 = PSLR.2 for output CC61 PSL3 = PSLR.3 for output COUT61 PSL4 = PSLR.4 for output CC62 PSL5 = PSLR.5 for output COUT62 00_H The passive level is 0. 01_H The passive level is 1.
PSL63	7	rwh	**Passive State Level of Output COUT63** This bit defines the passive level driven by the module output COUT63 during the passive state. 0_B The passive level is 0. 1_B The passive level is 1.
0	6, 31:8	r	**Reserved;** Returns 0 if read; should be written with 0.

Multi-Channel Mode Control Register

Register MCMCTR contains control bits for the multi-channel functionality.

MCMCTR

Multi-Channel Mode Control Register **(0094$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

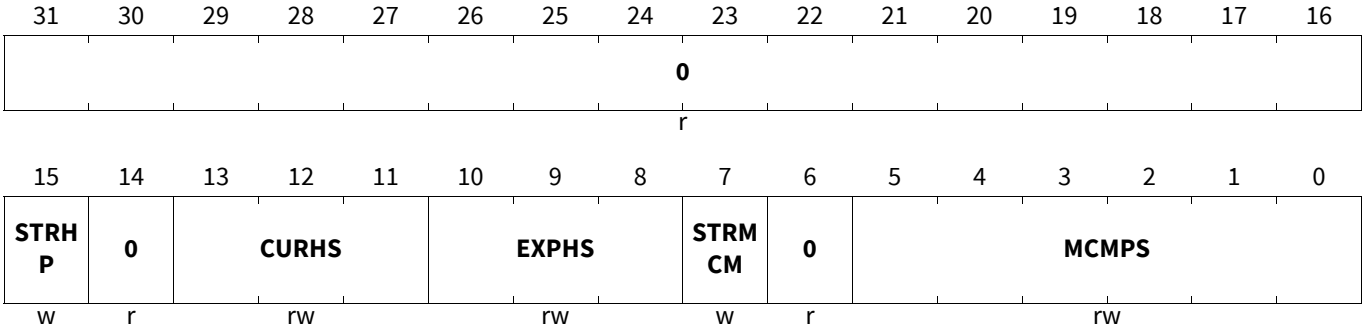
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					STE13U	STE12D	STE12U	0		SWSYN		0	SWSEL		
r					rw	rw	rw	r		rw		r	rw		

Field	Bits	Type	Description
SWSEL	2:0	rw	**Switching Selection** Bit field SWSEL selects one of the following trigger request sources (next multi-channel event) for the shadow transfer MCM_ST from MCMPS to MCMP. The trigger request is stored in the reminder flag R until the shadow transfer is done and flag R is cleared automatically with the shadow transfer. The shadow transfer takes place synchronously with an event selected in bit field SWSYN. 000_B No trigger request will be generated 001_B Correct Hall pattern detected (CM_CHE) 010_B T13 period-match detected (while counting up) 011_B T12 one-match (while counting down) 100_B T12 channel 1 compare-match detected (phase delay function) 101_B T12 period match detected (while counting up) 110_B reserved, no trigger request will be generated 111_B reserved, no trigger request will be generated
SWSYN	5:4	rw	**Switching Synchronization** Bit field SWSYN defines the synchronization mechanism of the shadow transfer event MCM_ST if it has been requested before (flag R set by an event selected by SWSEL) and if MCMEN = 1. This feature permits the synchronization of the outputs to the PWM source, that is used for modulation (T12 or T13). 00_B Direct; the trigger event immediately leads to the shadow transfer 01_B A T13 zero-match triggers the shadow transfer 10_B A T12 zero-match (while counting up) triggers the shadow transfer 11_B reserved; no action
STE12U	8	rw	**Shadow Transfer Enable for T12 Upcounting** This bit enables the shadow transfer T12_ST if flag MCMOUT.R is set or becomes set while a T12 period match is detected while counting up. 0_B No action 1_B The T12_ST shadow transfer mechanism is enabled if MCMEN = 1.
STE12D	9	rw	**Shadow Transfer Enable for T12 Downcounting** This bit enables the shadow transfer T12_ST if flag MCMOUT.R is set or becomes set while a T12 one match is detected while counting down. 0_B No action 1_B The T12_ST shadow transfer mechanism is enabled if MCMEN = 1.
STE13U	10	rw	**Shadow Transfer Enable for T13 Upcounting** This bit enables the shadow transfer T13_ST if flag MCMOUT.R is set or becomes set while a T13 period match is detected. 0_B No action 1_B The T13_ST shadow transfer mechanism is enabled if MCMEN = 1.
0	3, 7:6, 31:11	r	**Reserved;** Returns 0 if read; should be written with 0.

Multi-Channel Mode Output Shadow Register

Register MCMOUTS contains bits used as pattern input for the multi-channel mode and the Hall mode. This register is a shadow register (that can be read and written) for register MCMOUT, indicating the currently active signals.

MCMOUTS
Multi-Channel Mode Output Shadow Register (008C$_H$) **Application Reset Value: 0000 0000$_H$**

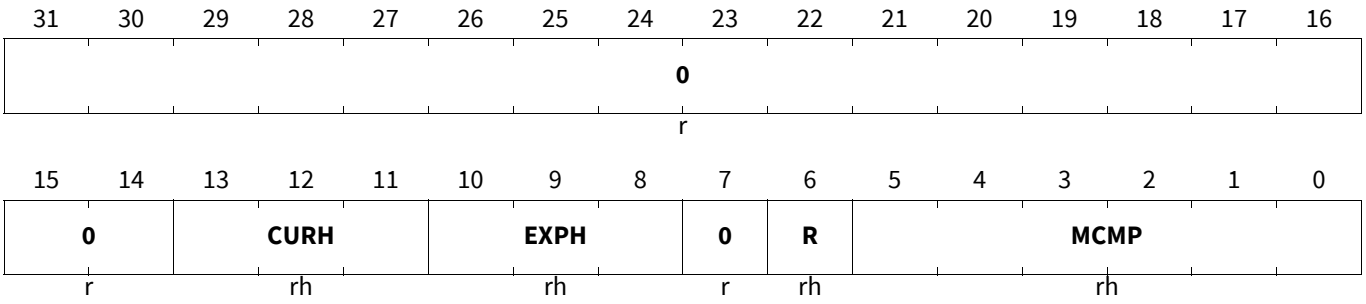


Field	Bits	Type	Description
MCMPS	5:0	rw	**Multi-Channel PWM Pattern Shadow** Bit field MCMPS is the shadow bit field for bit field MCMP. The multi-channel shadow transfer is triggered by MCM_ST according to the transfer conditions defined by register MCMCTR.
STRMCM	7	w	**Shadow Transfer Request for MCMPS** Writing STRMCM = 1 leads to an immediate activation of MCM_ST to update bit field MCMP by the value of MCMPS. When read, this bit always delivers 0. 0_B No action. 1_B Bit field MCMP is updated.
EXPHS	10:8	rw	**Expected Hall Pattern Shadow** Bit field EXPHS is the shadow bit field for bit field EXPH. The shadow transfer takes place when a correct Hall event is detected (CM_CHE).
CURHS	13:11	rw	**Current Hall Pattern Shadow** Bit field CURHS is the shadow bit field for bit field CURH. The shadow transfer takes place when a correct Hall event is detected (CM_CHE).
STRHP	15	w	**Shadow Transfer Request for the Hall Pattern** Writing STRHP = 1 leads to an immediate activation of HP_ST to update bit fields EXPH and CURH by EXPHS and CURHS. When read, this bit always delivers 0. 0_B No action. 1_B Bit fields EXPH and CURH are updated.
0	6, 14, 31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Multi-Channel Mode Output Register

MCMOUT

Multi-Channel Mode Output Register (0090$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
MCMP	5:0	rh	**Multi-Channel PWM Pattern** Bit field MCMP defines the output pattern for the multi-channel mode. If this mode is enabled by MODCTR.MCMEN = 1, the output state of all T12 related PWM outputs can be modified. This bit field is 0 while IS.IDLE = 1. MCMP0 = MCMOUT.0 for output CC60 MCMP1 = MCMOUT.1 for output COUT60 MCMP2 = MCMOUT.2 for output CC61 MCMP3 = MCMOUT.3 for output COUT61 MCMP4 = MCMOUT.4 for output CC62 MCMP5 = MCMOUT.5 for output COUT62 00_H The output is set to the passive state. A PWM generated by T12 or T13 are not taken into account. 01_H The output can be in the active state, depending on the enabled PWM modulation signals generated by T12, T13 and the trap state.
R	6	rh	**Reminder Flag** This flag indicates that the shadow transfer from MCMPS to MCMP has been requested by the selected trigger source. It is cleared when the shadow transfer takes place or while MCMEN=0. 0_B A shadow transfer MCM_ST is not requested. 1_B A shadow transfer MCM_ST is requested, but has not yet been executed, because the selected synchronization condition has not yet occurred.
EXPH	10:8	rh	**Expected Hall Pattern** Bit field EXPH is updated by a shadow transfer HP_ST from bit field EXPHS. If HCRDY = 1, EXPH is compared to the sampled CCPOSx inputs in order to detect the occurrence of the next desired (=expected) hall pattern or a wrong pattern. If the sampled hall pattern at the hall input pins is equal to bit field EXPH, a correct Hall event has been detected (CM_CHE).

Field	Bits	Type	Description
CURH	13:11	rh	**Current Hall Pattern** Bit field CURH is updated by a shadow transfer HP_ST from bit field CURHS. If HCRDY = 1, CURH is compared to the sampled CCPOSx inputs in order to detect a spike. If the sampled Hall pattern at the Hall input pins is equal to bit field CURH, no Hall event has been detected. If the sampled Hall input pattern is neither equal to CURH nor equal to EXPH, the Hall event was not the desired one and may be due to a fatal error (e.g. blocked rotor, etc.). In this case, a wrong Hall event has been detected (CM_WHE).
0	7, 31:14	r	**Reserved;** Returns 0 if read; should be written with 0.

29.10 Interrupt Handling

This section describes the interrupt handling of the CCU6 module.

29.10.1 Interrupt Structure

The HW interrupt event or the SW setting of the corresponding interrupt set bit (in register ISS) sets the event indication flags (in register IS) and can trigger the interrupt generation. The interrupt pulse is generated independently from the interrupt status flag in register IS (it is not necessary to clear the related status bit to be able to generate another interrupt). The interrupt flag can be cleared by SW by writing to the corresponding bit in register ISR.

If enabled by the related interrupt enable bit in register IEN, an interrupt pulse can be generated on one of the four service request outputs (SR0 to SR3) of the module. If more than one interrupt source is connected to the same interrupt node pointer (in register INP), the requests are logically OR-combined to one common service request output (see **Figure 196**).

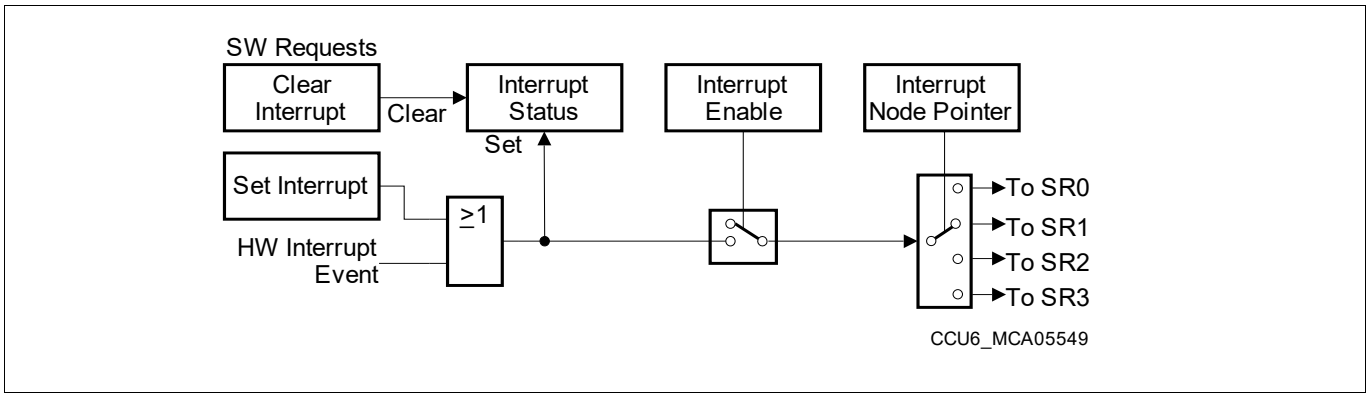


Figure 196 General Interrupt Structure

The available interrupt events in the CCU6 are shown in **Figure 197**.

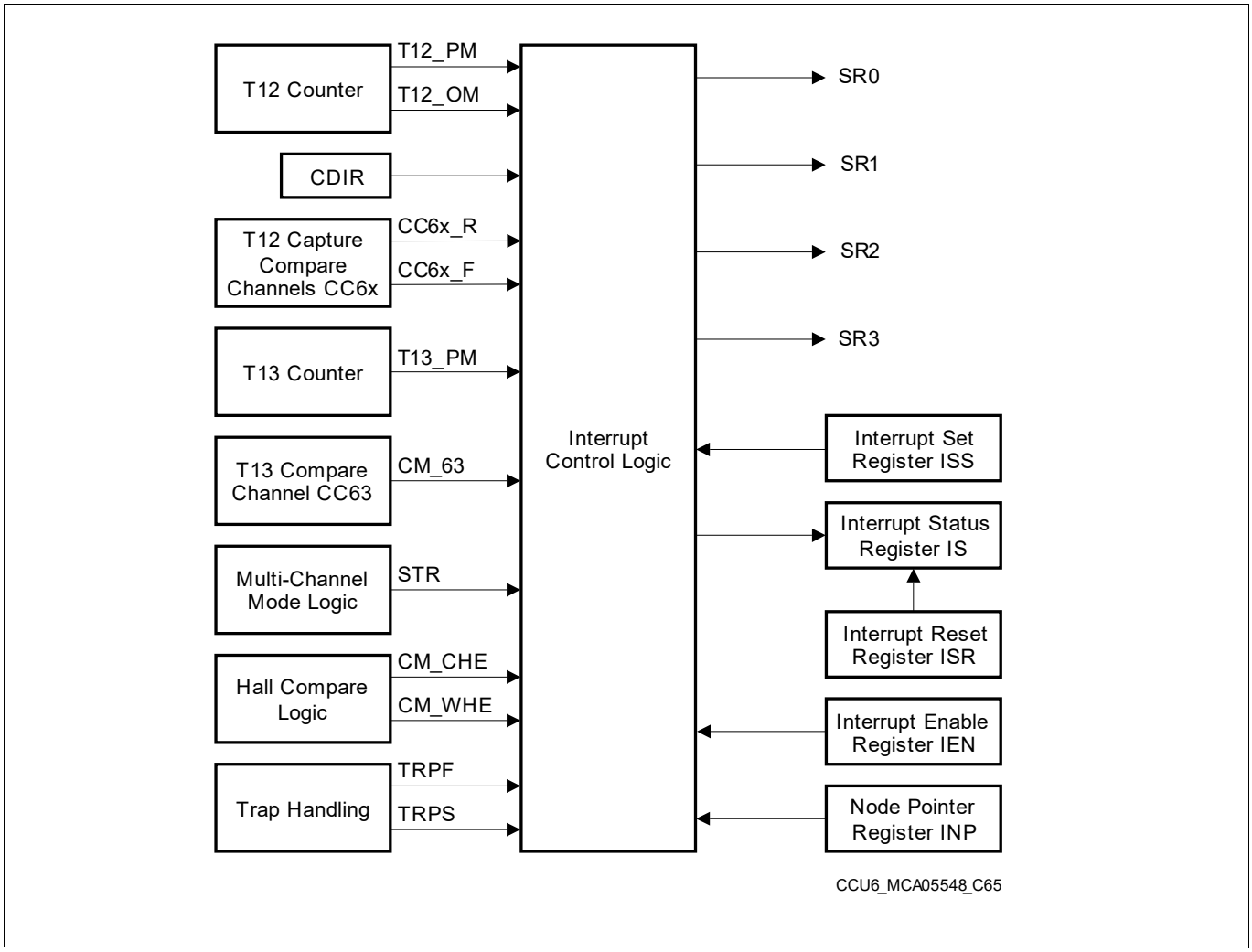


Figure 197 Interrupt Sources and Events

29.10.2 Interrupt Registers

Interrupt Status Register

Register IS contains the individual interrupt request bits. This register can only be read, write actions have no impact on the contents of this register. The SW can set or clear the bits individually by writing to the registers ISS (to set the bits) or to register ISR (to clear the bits).

The interrupt generation is independent from the value of the bits in register IS, e.g. the interrupt will be generated (if enabled) even if the corresponding bit is already set. The trigger for an interrupt generation is the detection of a set condition (by HW or SW) for the corresponding bit in register IS.

In compare mode (and hall mode), the timer-related interrupts are only generated while the timer is running (T1xR=1). In capture mode, the capture interrupts are also generated while the timer T12 is stopped.

Note: Not all bits in register IS can generate an interrupt. Other status bits have been added, that have a similar structure for their set and clear actions. It is recommended that SW checks the interrupt bits bit-wisely (instead of common OR over the bits).

IS

Interrupt Status Register **(00A0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STR	IDLE	WHE	CHE	TRPS	TRPF	T13PM	T13CM	T12PM	T12OM	ICC62F	ICC62R	ICC61F	ICC61R	ICC60F	ICC60R
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
ICC6xR (x=0-2)	2*x	rh	**Capture, Compare-Match Rising Edge Flag ICC6xR (x=0,1,2)** This bit indicates that event CC6x_R has been detected. This event occurs in compare mode when a compare-match is detected while T12 is counting up (CM_6x and CDIR = 0) and in capture mode when a rising edge is detected at the related input CC6xIN. 0_B The event has not yet been detected. 1_B The event has been detected.
ICC6xF (x=0-2)	2*x+1	rh	**Capture, Compare-Match Falling Edge Flag ICC6xF (x=0,1,2)** This bit indicates that event CC6x_F has been detected. This event occurs in compare mode when a compare-match is detected while T12 is counting down (CM_6x and CDIR = 1) and in capture mode when a falling edge is detected at the related input CC6xIN. 0_B The event has not yet been detected. 1_B The event has been detected.

Field	Bits	Type	Description
T12OM	6	rh	**Timer T12 One-Match Flag** This bit indicates that a timer T12 one-match while counting down (T12_OM and CDIR = 1) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
T12PM	7	rh	**Timer T12 Period-Match Flag** This bit indicates that a timer T12 period-match while counting up (T12_PM and CDIR = 0) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
T13CM	8	rh	**Timer T13 Compare-Match Flag** This bit indicates that a timer T13 compare-match (CM_63) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
T13PM	9	rh	**Timer T13 Period-Match Flag** This bit indicates that a timer T13 period-match (T13_PM) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
TRPF	10	rh	**Trap Flag** This bit indicates if a trap condition (input $\overline{\text{CTRAP}}$ = 0 or by SW) is / has been detected. If TRPM2= 0, it becomes cleared automatically if $\overline{\text{CTRAP}}$ = 1 or TRPPEN = 0, whereas if TRPM2 = 1, it has to be cleared by writing RTRPF = 1. 0_B The trap condition has not been detected. 1_B The trap condition is / has been detected.
TRPS	11	rh	**Trap State** This bit indicates the actual trap state. It is set if TRPF = 1 and becomes cleared according to the mode selected in register TRPCTR. 0_B The trap state is not active. 1_B The trap state is active.
CHE	12	rh	**Correct Hall Event** This bit indicates that a correct Hall event (CM_CHE) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
WHE	13	rh	**Wrong Hall Event** This bit indicates that a wrong Hall event (CM_WHE) has been detected. 0_B The event has not yet been detected. 1_B The event has been detected.
IDLE	14	rh	**IDLE State** If enabled by ENIDLE = 1, this bit is set together with bit WHE and it has to be cleared by SW. 0_B No action. 1_B Bit field MCMP is cleared, the selected outputs are set to passive state.

Field	Bits	Type	Description
STR	15	rh	**Multi-Channel Mode Shadow Transfer Request** This bit indicates that a shadow transfer from MCMPS to MCMP (MCM_ST) has taken place. 0_B The event has not yet been detected. 1_B The event has been detected.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Interrupt Status Set Register

Register ISS contains individual interrupt request set bits to generate a CCU6 interrupt request by software. Writing a 1 sets the bit(s) in register IS at the corresponding bit position(s) and can generate an interrupt event (if available and enabled). All bit positions read as 0.

ISS

Interrupt Status Set Register **($00A4_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSTR	**SIDLE**	**SWHE**	**SCHE**	**SWHC**	**STRPF**	**ST13PM**	**ST13CM**	**ST12PM**	**ST12OM**	**SCC62F**	**SCC62R**	**SCC61F**	**SCC61R**	**SCC60F**	**SCC60R**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
SCC6xR (x=0-2)	2*x	w	**Set Capture, Compare-Match Rising Edge Flag** 0_B No action 1_B Bit CC6xR will be set.
SCC6xF (x=0-2)	2*x+1	w	**Set Capture, Compare-Match Falling Edge Flag** 0_B No action 1_B Bit CC6xF will be set.
ST12OM	6	w	**Set Timer T12 One-Match Flag** 0_B No action 1_B Bit T12OM will be set.
ST12PM	7	w	**Set Timer T12 Period-Match Flag** 0_B No action 1_B Bit T12PM will be set.
ST13CM	8	w	**Set Timer T13 Compare-Match Flag** 0_B No action 1_B Bit T13CM will be set.
ST13PM	9	w	**Set Timer T13 Period-Match Flag** 0_B No action 1_B Bit T13PM will be set.

Field	Bits	Type	Description
STRPF	10	w	**Set Trap Flag** 0_B No action 1_B Bits TRPF and TRPS will be set.
SWHC	11	w	**Software Hall Compare** 0_B No action 1_B The Hall compare action is triggered.
SCHE	12	w	**Set Correct Hall Event Flag** 0_B No action 1_B Bit CHE will be set.
SWHE	13	w	**Set Wrong Hall Event Flag** 0_B No action 1_B Bit WHE will be set.
SIDLE	14	w	**Set IDLE Flag** 0_B No action 1_B Bit IDLE will be set.
SSTR	15	w	**Set STR Flag** 0_B No action 1_B Bit STR will be set.
0	31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Interrupt Status Reset Register

Register ISR contains bits to individually clear the interrupt event flags by software. Writing a 1 clears the bit(s) in register IS at the corresponding bit position(s). All bit positions read as 0.

ISR

Interrupt Status Reset Register **($00A8_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSTR	**RIDLE**	**RWHE**	**RCHE**	**0**	**RTRPF**	**RT13PM**	**RT13CM**	**RT12PM**	**RT12OM**	**RCC62F**	**RCC62R**	**RCC61F**	**RCC61R**	**RCC60F**	**RCC60R**
w	w	w	w	r	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RCC6xR (x=0-2)	2*x	w	**Reset Capture, Compare-Match Rising Edge Flag** 0_B No action 1_B Bit CC6xR will be cleared.
RCC6xF (x=0-2)	2*x+1	w	**Reset Capture, Compare-Match Falling Edge Flag** 0_B No action 1_B Bit CC6xF will be cleared.

Field	Bits	Type	Description
RT12OM	6	w	**Reset Timer T12 One-Match Flag** 0_B No action 1_B Bit T12OM will be cleared.
RT12PM	7	w	**Reset Timer T12 Period-Match Flag** 0_B No action 1_B Bit T12PM IS will be cleared.
RT13CM	8	w	**Reset Timer T13 Compare-Match Flag** 0_B No action 1_B Bit T13CM will be cleared.
RT13PM	9	w	**Reset Timer T13 Period-Match Flag** 0_B No action 1_B Bit T13PM will be cleared.
RTRPF	10	w	**Reset Trap Flag** 0_B No action 1_B Bit TRPF will be cleared (not taken into account while input $\overline{\text{CTRAP}}$=0 and TRPPEN=1.
RCHE	12	w	**Reset Correct Hall Event Flag** 0_B No action 1_B Bit CHE will be cleared.
RWHE	13	w	**Reset Wrong Hall Event Flag** 0_B No action 1_B Bit WHE will be cleared.
RIDLE	14	w	**Reset IDLE Flag** 0_B No action 1_B Bit IDLE will be cleared.
RSTR	15	w	**Reset STR Flag** 0_B No action 1_B Bit STR will be cleared.
0	11, 31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Interrupt Enable Register

Register IEN contains the interrupt enable bits and a control bit to enable the automatic idle function in the case of a wrong hall pattern.

IEN

Interrupt Enable Register **(00B0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENSTR	ENIDLE	ENWHE	ENCHE	0	ENTRPF	ENT13PM	ENT13CM	ENT12PM	ENT12OM	ENCC62F	ENCC62R	ENCC61F	ENCC61R	ENCC60F	ENCC60R
rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENCC6xR (x=0-2)	2*x	rw	**Capture, Compare-Match Rising Edge Interrupt Enable for Channel CC6x ENCC6xF (x=0,1,2)** 0_B No interrupt will be generated if the set condition for bit CC6xR in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CC6xR in register IS occurs. The service request output that will be activated is selected by bit field INPCC6x.
ENCC6xF (x=0-2)	2*x+1	rw	**Capture, Compare-Match Falling Edge Interrupt Enable for Channel CC6x ENCC6xF (x=0,1,2)** 0_B No interrupt will be generated if the set condition for bit CC6xF in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CC6xF in register IS occurs. The service request output that will be activated is selected by bit field INPCC6x.
ENT12OM	6	rw	**Enable Interrupt for T12 One-Match** 0_B No interrupt will be generated if the set condition for bit T12OM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T12OM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.
ENT12PM	7	rw	**Enable Interrupt for T12 Period-Match** 0_B No interrupt will be generated if the set condition for bit T12PM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T12PM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.
ENT13CM	8	rw	**Enable Interrupt for T13 Compare-Match** 0_B No interrupt will be generated if the set condition for bit T13CM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T13CM in register IS occurs. The service request output that will be activated is selected by bit field INPT13.

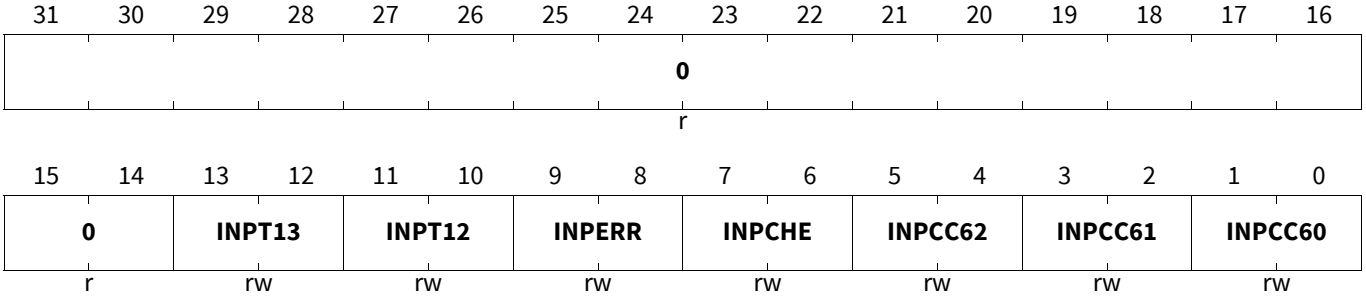
Field	Bits	Type	Description
ENT13PM	9	rw	**Enable Interrupt for T13 Period-Match** 0_B No interrupt will be generated if the set condition for bit T13PM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T13PM in register IS occurs. The service request output that will be activated is selected by bit field INPT13.
ENTRPF	10	rw	**Enable Interrupt for Trap Flag** 0_B No interrupt will be generated if the set condition for bit TRPF in register IS occurs. 1_B An interrupt will be generated if the set condition for bit TRPF in register IS occurs. The service request output that will be activated is selected by bit field INPERR.
ENCHE	12	rw	**Enable Interrupt for Correct Hall Event** 0_B No interrupt will be generated if the set condition for bit CHE in register IS occurs. 1_B An interrupt will be generated if the set condition for bit CHE in register IS occurs. The service request output that will be activated is selected by bit field INPCHE.
ENWHE	13	rw	**Enable Interrupt for Wrong Hall Event** 0_B No interrupt will be generated if the set condition for bit WHE in register IS occurs. 1_B An interrupt will be generated if the set condition for bit WHE in register IS occurs. The service request output that will be activated is selected by bit field INPERR.
ENIDLE	14	rw	**Enable Idle** This bit enables the automatic entering of the idle state (bit IDLE will be set) after a wrong hall event has been detected (bit WHE is set). During the idle state, the bit field MCMP is automatically cleared. 0_B The bit IDLE is not automatically set when a wrong hall event is detected. 1_B The bit IDLE is automatically set when a wrong hall event is detected.
ENSTR	15	rw	**Enable Multi-Channel Mode Shadow Transfer Interrupt** 0_B No interrupt will be generated if the set condition for bit STR in register IS occurs. 1_B An interrupt will be generated if the set condition for bit STR in register IS occurs. The service request output that will be activated is selected by bit field INPCHE.
0	11, 31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Interrupt Node Pointer Register

Register INP contains the interrupt node pointers allowing a flexible interrupt handling. These bit fields define which service request output will be activated if the corresponding interrupt event occurs and the interrupt generation for this event is enabled.

INP

Interrupt Node Pointer Register **($00AC_H$)** **Application Reset Value: $0000\ 3940_H$**



Field	Bits	Type	Description
INPCC6x (x=0-2)	2*x+1:2*x	rw	**Interrupt Node Pointer for Channel CC6x Interrupts INPCC6x (x=0,1,2)** This bit field defines the service request output activated due to a set condition for bit CC6xR (if enabled by bit ENCC6xR) or for bit CC6xF (if enabled by bit ENCC6xF). 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
INPCHE	7:6	rw	**Interrupt Node Pointer for the CHE Interrupt** This bit field defines the service request output activated due to a set condition for bit CHE (if enabled by bit ENCHE) of for bit STR (if enabled by bit ENSTR). Coding see INPCC6x.
INPERR	9:8	rw	**Interrupt Node Pointer for Error Interrupts** This bit field defines the service request output activated due to a set condition for bit TRPF (if enabled by bit ENTRPF) or for bit WHE (if enabled by bit ENWHE). Coding see INPCC6x.
INPT12	11:10	rw	**Interrupt Node Pointer for Timer12 Interrupts** This bit field defines the service request output activated due to a set condition for bit T12OM (if enabled by bit ENT12OM) or for bit T12PM (if enabled by bit ENT12PM). Coding see INPCC6x.
INPT13	13:12	rw	**Interrupt Node Pointer for Timer13 Interrupt** This bit field defines the service request output activated due to a set condition for bit T13CM (if enabled by bit ENT13CM) or for bit T13PM (if enabled by bit ENT13PM). Coding see INPCC6x.
0	31:14	r	**Reserved;** Returns 0 if read; should be written with 0.

29.11 General Module Operation

This section provides information about the:

- Input Selection (see **Section 29.11.1**)
- Input Monitoring (see **Section 29.11.2**)
- OCDS Suspend Functionality (see **Section 29.11.3**)
- OCDS Trigger Bus (OTGB) Interface (see **Section 29.11.4**)
- General Register Description (see **Section 29.11.5**)
- BPI Register Description (see **Section 29.11.6**)

29.11.1 Input Selection

Each CCU6 input signal can be selected from a vector of four or eight possible inputs by programming the port input select registers **PISEL0** and **PISEL2**. This permits to adapt the pin functionality of the device to the application requirements.
The output pins for the module output signals are chosen in the ports.

Naming convention:
The input vector CC60IN[D:A] for input signal CC60IN is composed of the signals CC60INA to CC60IND.

Note: All functional inputs of the CCU6 are synchronized to f_{CC6} before they affect the module internal logic. The resulting delay of $2/f_{CC6}$ and for asynchronous signals an additional uncertainty of $1/f_{CC6}$ have to be taken into account for precise timing calculation. An edge of an input signal can only be correctly detected if the high phase and the low phase of the input signal are both longer than $1/f_{CC6}$.

29.11.2 Input Monitoring

A selected event which occurs at each CCU6 input signal can be monitored through **IMON**.x. Every input signal can be included for the detection of a lost bit event (**IMON**.LBE) if enabled through its individual lost indicator enable bits (**LI**.yEN). The lost bit event occurs if a selected event occurs again with the previous event captured (IMON.x remains set) and its lost indicator is enabled for at least one of the monitored input signals. The lost bit event can be enabled (**LI**.LBEEN) for an interrupt to be generated at one of the SRx line, selected through **LI**.INPLBE. The LBE output signal of the kernel can be connected to a capture input to indicate when the lost bit event happens.

The lost bit event can be used as a kind of interrupt or event watchdog to monitor if an action related to an event has been processed before a second event of the same type occurs. Like this, if a certain event is treated by an interrupt that should be monitored, the related indication flag has to be cleared by SW. If the SW has not yet cleared the flag and the event occurs again, the event is considered as being lost and another interrupt can be generated to inform the system about the loss. This can be also used to indicate that input events occur too often and the main task has not enough time to treat them.

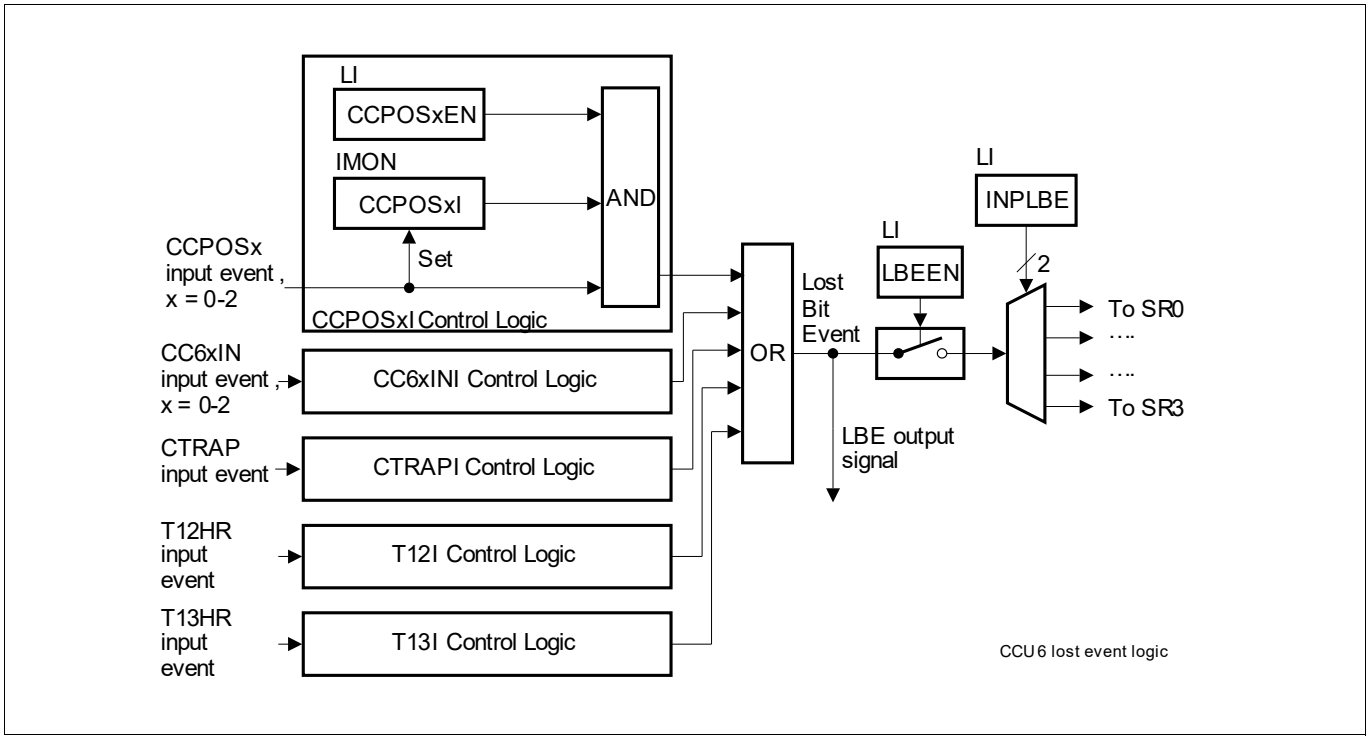


Figure 198 Lost Event Logic

29.11.3 OCDS Suspend

The behavior of CCU6 upon an OCDS suspend request is controlled by the **OCS** register. CCU6 supports both Hard Suspend Mode and Soft Suspend Mode.

Hard Suspend Mode

In Hard Suspend Mode the CCU6 kernel clock is switched off immediately. Reading and writing of registers is possible but will enable the kernel clock for a few cycles.

Attention: Register accesses with clocking in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so a CCU6 kernel reset might not be sufficient to bring the system into a defined state.

Soft Suspend Mode

In Soft Suspend Mode CCU6 may finalize specific actions before it enters the suspended state with **OCS**.SUSSTA set. This can be advantageous for applications (e.g. motor control) where critical system states could occur if the kernel clock would be switched off immediately upon an OCDS suspend request.

Soft Suspend mode does not influence the kernel clock. Reading and writing of registers is possible without the side effects that may occur when reading/writing registers in Hard Suspend Mode.

For CCU6 , two basic soft suspend options (**Stop Mode 0**, **Stop Mode 1**) are available, selected via bit field **OCS**.SUS.

In addition, the internal functional blocks (T12, T13, Hall logic, Trap logic) may individually be programmed via their Sensitivity Bits in register **KSCSR** to accept or ignore a suspend request. If the request sensitivity is disabled, the block continues normal operation. If the request sensitivity is enabled, the block operates as specified for the selected stop mode. The mapping of the CCU6 functional blocks to their Sensitivity Bits is shown in **Table 203**.

Stop Mode 0

In Stop Mode 0, if selected to be stopped, the Hall and Trap logic stops immediately, while Timer T12 and/or T13 continues normal operation (if running) until it reaches the end of the PWM period; then it stops (same stop condition as in single shot mode). When Timer T12 stops, the capture inputs CC6xIN are frozen.

Stop Mode 1

In Stop Mode 1, if selected to be stopped, the internal functional blocks (T12, T13, Hall logic, Trap logic) will stop immediately upon a suspend request.

If the sensitivity bits for Timer T12 or T13 are set, the corresponding output lines enabled for the trap condition are set to their passive values (similar to a trap condition).

Table 203 summarizes the reaction of the CCU6 functional blocks to OCDS suspend requests.

Table 203 CCU6 Functional Blocks

Block	Reaction to OCDS Suspend Request	Sensitivity Bit
0	**Timer T12:** if bit SB0 = 1_B, - in **Stop Mode 0** Timer T12 continues (if running) until the end of the PWM period. Then it stops and the CCxIN input stages are frozen. - in **Stop Mode 1** Timer T12 stops immediately and the CC6xIN input stages are frozen. Output lines CC6x, COUT6x with enabled trap functionality are set to their passive states.	KSCSR.SB0
1	**Timer T13:** if bit SB1 = 1_B, - in **Stop Mode 0** Timer T13 continues (if running) until the end of the PWM period. Then it stops. - in **Stop Mode 1** Timer T13 stops immediately. If trap functionality for COUT63 is enabled, it is set to the passive state.	KSCSR.SB1
2	**Hall Logic:** if bit SB2 = 1_B, the hall logic is stopped immediately and the CCPOSx input stages are frozen (same behavior for Stop Mode 0 and 1).	KSCSR.SB2
3	**Trap Logic:** if bit SB3 = 1_B, the trap logic is stopped immediately and the CTRAP input stage is frozen (same behavior for Stop Mode 0 and 1).	KSCSR.SB3

29.11.4 OCDS Trigger Bus (OTGB) Interface

The CCU6 kernel provides a set of 16 internal status signals that are combined to a Trigger Set. The CCU6 Trigger Set is shown in **Table 204**. It is output on OTGB0 or OTGB1 controlled by the **OCS** register.

Table 204 TS16_CCU6 Trigger Set CCU6

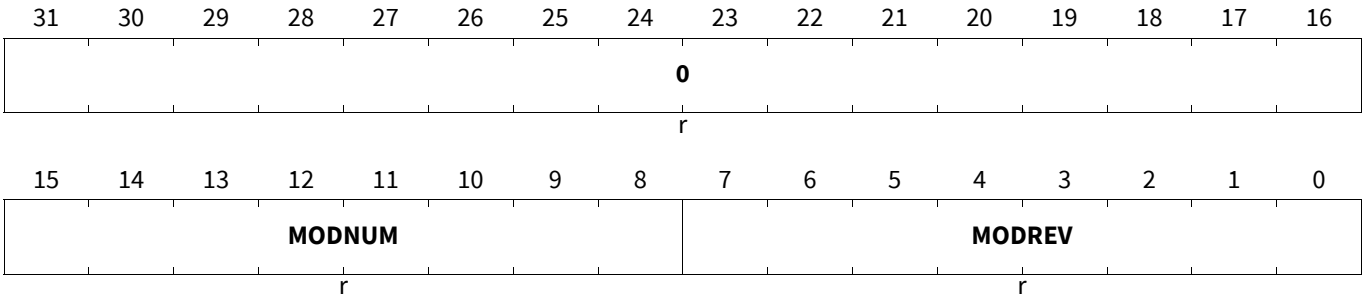
Bits	Name	Description
0	T12pre	T12 prescaler output
1	T13pre	T13 prescaler output
2	T12_cm0	T12 compare match channel 0
3	T12_cm1	T12 compare match channel 1
4	T12_cm2	T12 compare match channel 2
5	T12_pm	T12 period match
6	T12_om	T12 one match
7	T12_cdir	T12 count direction
8	T12_zm	T12 zero match
9	T13_cm	T13 compare match
10	T13_pm	T13 period match
11	T13_zm	T13 zero match
12	mcm_st	MCM shadow transfer
13	che	Correct Hall event
14	whe	Wrong Hall event
15	trpf	Trap flag

29.11.5 General Registers

Module Identification Register

The CCU6 Module Identification Register ID contains read-only information about the module identification number and its revision.

ID
Module Identification Register **(0008_H)** **Reset Value: Table 205**



Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** MODREV defines the module revision number. The value of a module revision starts with 01_H (first revision), 02_H, 03_H,...up to FF_H.
MODNUM	15:8	r	**Module Number Value** This bit field defines the module identification number for the CCU6: 54_H
0	31:16	r	**Reserved** Read as 0.

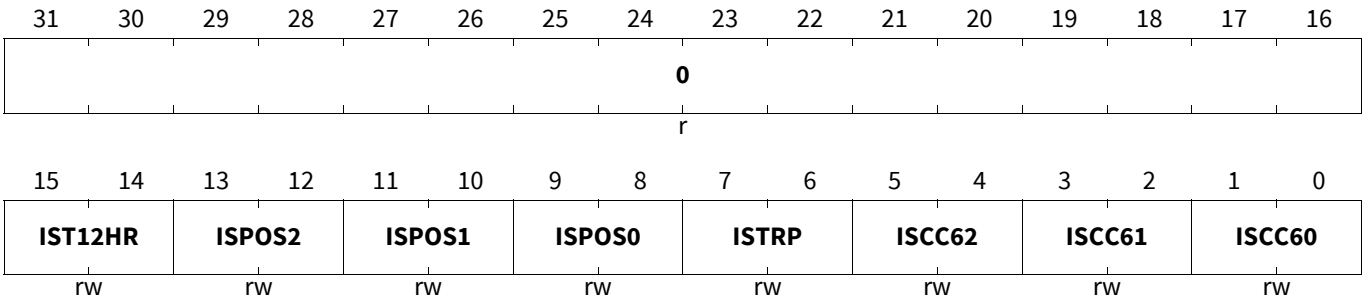
Table 205 Reset Values of ID

Reset Type	Reset Value	Note
Application Reset	0000 $54XX_H$	

Port Input Select Register 0

Registers PISEL0 and PISEL2 contain bit fields selecting the actual input signal for the module inputs.

PISEL0
Port Input Select Register 0 **(0010_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
ISCC6x (x=0-2)	2*x+1:2*x	rw	**Input Select for CC60 ISCC6x (x=0,1,2)** This bit field defines the input signal used as CC6x capture input. 00_B The signal CC6xINA is selected. 01_B The signal CC6xINB is selected. 10_B The signal CC6xINC is selected. 11_B The signal CC6xIND is selected.
ISTRP	7:6	rw	**Input Select for CTRAP** This bit field defines the input signal used as CTRAP input. 00_B The signal CTRAPA is selected. 01_B The signal CTRAPB is selected. 10_B The signal CTRAPC is selected. 11_B The signal CTRAPD is selected.
ISPOSx (x=0-2)	2*x+9:2*x+8	rw	**Input Select for CCPOS0 ISPOSx (x=0,1,2)** This bit field defines the input signal used as CCPOSx input. 00_B The signal CCPOSxA is selected. 01_B The signal CCPOSxB is selected. 10_B The signal CCPOSxC is selected. 11_B The signal CCPOSxD is selected.
IST12HR	15:14	rw	**Input Select for T12HR** This bit field defines the input signal used as T12HR input. 00_B Either signal T12HRA (if T12EXT = 0) or T12HRE (if T12EXT = 1) is selected. 01_B Either signal T12HRB (if T12EXT = 0) or T12HRF (if T12EXT = 1) is selected. 10_B Either signal T12HRC (if T12EXT = 0) or T12HRG (if T12EXT = 1) is selected. 11_B Either signal T12HRD (if T12EXT = 0) or T12HRH (if T12EXT = 1) is selected.
0	31:16	r	**Reserved** Returns 0 if read, should be written with 0.

Port Input Select Register 2

PISEL2

Port Input Select Register 2 **(0014_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

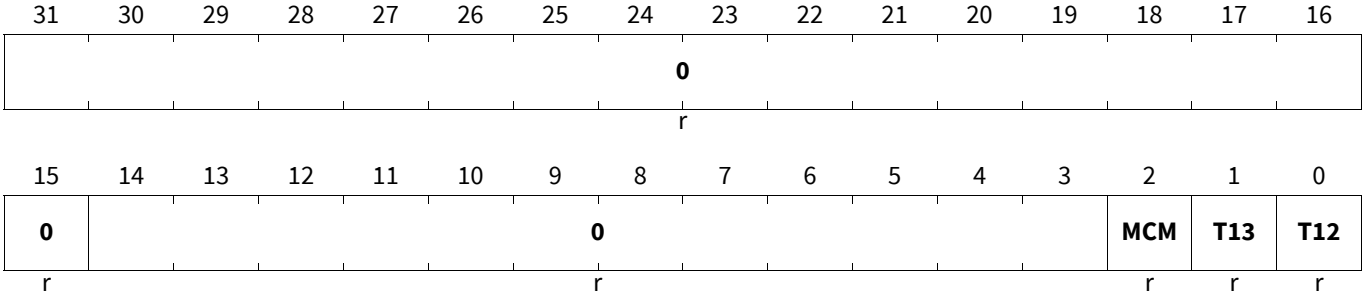
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								T13EXT	T12EXT	ISCNT13		ISCNT12		IST13HR	
r								rw	rw	rw		rw		rw	

Field	Bits	Type	Description
IST13HR	1:0	rw	**Input Select for T13HR** This bit field defines the input signal used as T13HR input. 00_B Either signal T13HRA (if T13EXT = 0) or T13HRE (if T13EXT = 1) is selected. 01_B Either signal T13HRB (if T13EXT = 0) or T13HRF (if T13EXT = 1) is selected. 10_B Either signal T13HRC (if T13EXT = 0) or T13HRG (if T13EXT = 1) is selected. 11_B Either signal T13HRD (if T13EXT = 0) or T13HRH (if T13EXT = 1) is selected.
ISCNT12	3:2	rw	**Input Select for T12 Counting Input** This bit field defines the input event leading to a counting action of T12. 00_B The T12 prescaler generates the counting events. Bit TCTR4.T12CNT is not taken into account. 01_B Bit TCTR4.T12CNT written with 1 is a counting event. The T12 prescaler is not taken into account. 10_B The timer T12 is counting each rising edge detected in the selected T12HR signal. 11_B The timer T12 is counting each falling edge detected in the selected T12HR signal.
ISCNT13	5:4	rw	**Input Select for T13 Counting Input** This bit field defines the input event leading to a counting action of T13. 00_B The T13 prescaler generates the counting events. Bit TCTR4.T13CNT is not taken into account. 01_B Bit TCTR4.T13CNT written with 1 is a counting event. The T13 prescaler is not taken into account. 10_B The timer T13 is counting each rising edge detected in the selected T13HR signal. 11_B The timer T13 is counting each falling edge detected in the selected T13HR signal.
T12EXT	6	rw	**Extension for T12HR Inputs** This bit extends the 2-bit field IST12HR. 0_B One of the signals T12HR[D:A] is selected. 1_B One of the signals T12HR[H:E] is selected.
T13EXT	7	rw	**Extension for T13HR Inputs** This bit extends the 2-bit field IST13HR. 0_B One of the signals T13HR[D:A] is selected. 1_B One of the signals T13HR[H:E] is selected.
0	31:8	r	**Reserved;** Returns 0 if read; should be written with 0.

Module Configuration Register

The module configuration register contains bits describing the functionality that is available in the CCU6 module.

MCFG
Module Configuration Register **(0004_H)** **Reset Value: Table 206**



Field	Bits	Type	Description
T12	0	r	**T12 Available** This bit indicates if the T12 block is available. 0_B The T12 block is not available. A write access to T12PR is ignored. 1_B The T12 block is available. A write access to T12PR is executed.
T13	1	r	**T13 Available** This bit indicates if the T13 block is available. 0_B The T13 block is not available. A write access to T13PR is ignored. 1_B The T13 block is available. A write access to T13PR is executed.
MCM	2	r	**Multi-Channel Mode Available** This bit indicates if the multi-channel mode functionality is available. 0_B The multi-channel mode functionality is not available. A write access to MCMOUTS is ignored. 1_B The multi-channel mode functionality is available. A write access to MCMOUTS is executed.
0	14:3, 15, 31:16	r	**Reserved;** read as 0; should be written with 0.

Table 206 Reset Values of MCFG

Reset Type	Reset Value	Note
Application Reset	$0000\ 0007_H$	

The input monitoring register monitors the occurrence of a selected event for the input signals. If a hardware event triggers the setting of bit IMON.x and the same bit is written with 1 via software at the same time, then the corresponding bit is cleared (software overrules hardware). The lost bit event is indicated if an event is detected again at one or more input signals with its lost indicator enabled.

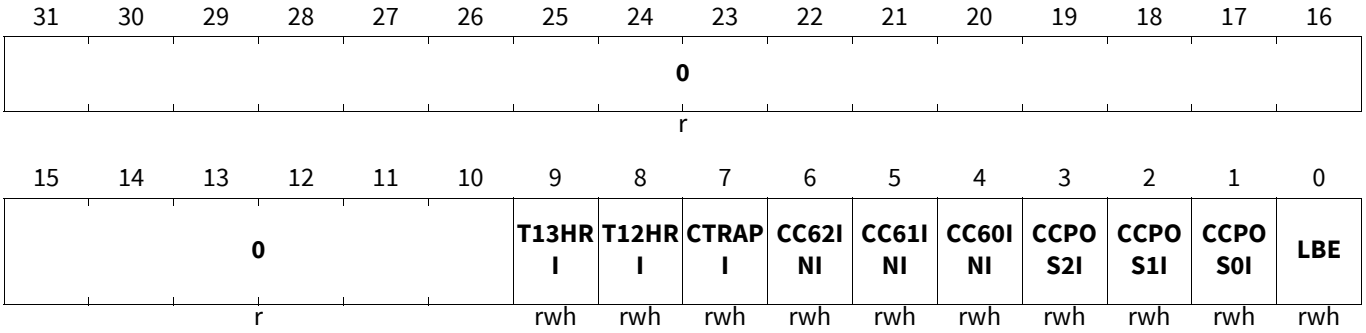
Note: The register is only applicable in capture modes if the edges are selected through T12MSEL.MSEL6x.

Input Monitoring Register

The input monitoring register monitors the occurrence of a selected event for the input signals. If a hardware event triggers the setting of bit IMON.x and the same bit is written with 1 via software at the same time, then the corresponding bit is cleared (software overrules hardware). The lost bit event is indicated if an event is detected again at one or more input signals with its lost indicator enabled.

Note: The register is only applicable in capture modes if the edges are selected through T12MSEL.MSEL6x.

IMON
Input Monitoring Register **(0098_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
LBE	0	rwh	**Lost Bit Event** This bit determines if a lost bit event has occurred. A lost bit event occurs when a selected event occurs again with the previous event captured (IMON.x remains set) and its lost indicator is enabled, for at least one of the monitored input signals. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B The lost bit event has not occurred. 1_B The lost bit event has occurred.
CCPOSxI (x=0-2)	x+1	rwh	**Event indication for input signal CCPOSx** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. *Note: The dedicated edge is indicated for a selected event if Hysteretic-like Control or Capture modes are initialized in T12MSEL.MSEL6x. If these modes are not selected, then all edges will be indicated as an event for the inputs.* 0_B A selected event has not occurred. 1_B Edge detection indicates a selected event has occurred.
CC6xINI (x=0-2)	x+4	rwh	**Event indication for input signal CC6xIN** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B A selected event has not occurred. 1_B Edge detection indicates a selected event has occurred.
CTRAPI	7	rwh	**Event indication for input signal CTRAP** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B An event has not occurred. 1_B Edge detection indicates an event has occurred.

Field	Bits	Type	Description
T12HRI	8	rwh	**Event indication for input signal T12HR** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B An event has not occurred. 1_B Edge detection indicates an event has occurred.
T13HRI	9	rwh	**Event indication for input signal T13HR** The bit determines if the selected event has occurred via an edge detection. The bit can be cleared by writing a 1 to the same bit position, while writing a 0 has no effect. 0_B An event has not occurred. 1_B Edge detection indicates an event has occurred.
0	31:10	r	**Reserved;** Returns 0 if read; should be written with 0.

Register Access Notes

Tricore atomic instructions (LDMST, ST.T, SWAP.W, SWAPMASK.W, CMPSWAP.W) only write back bits that are changing their level. This leads to the fact that bits that are already set cannot be written with a 1 when using RMW instructions.

This affects the bits in register IMON which are cleared by writing 1s.

No problem exists when using direct write instructions (e.g. ST.W).

Lost Indicator Register

The lost indicator register has the lost indicator enable bits for its detected event at the input signals. The lost bit event can then be enabled as an output signal through one of the service request lines.

LI

Lost Indicator Register **($009C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INPLBE		LBEEN	0			T13HREN	T12HREN	CTRAPEN	CC62INEN	CC61INEN	CC60INEN	CCPOS2EN	CCPOS1EN	CCPOS0EN	0
rw		rw	r			rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Field	Bits	Type	Description
CCPOSxEN (x=0-2)	x+1	rw	**Lost Indicator Enable for input signal CCPOSx** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.

Field	Bits	Type	Description
CC6xINEN (x=0-2)	x+4	rw	**Lost Indicator Enable for input signal CC6xIN** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.
CTRAPEN	7	rw	**Lost Indicator Enable for input signal CTRAP** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.
T12HREN	8	rw	**Lost Indicator Enable for input signal T12HR** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.
T13HREN	9	rw	**Lost Indicator Enable for input signal T13HR** This bit determines if the monitored event at the input signal is enabled for the detection of a lost bit event. 0_B Input signal is disabled for a lost bit event detection. 1_B Input signal is enabled for a lost bit event detection.
LBEEN	13	rw	**Interrupt Enable for Lost Bit Event** This bit determines if a SRx line is activated if lost bit event is detected. 0_B Lost bit event is disabled for the activation of a SRx line. 1_B Lost bit event is enabled for the activation of a SRx line.
INPLBE	15:14	rw	**Interrupt Node Pointer for lost bit event** This bit field defines which service request output line is selected to output an lost event alert for an enabled lost bit event. 00_B Service request output SR0 is selected. 01_B Service request output SR1 is selected. 10_B Service request output SR2 is selected. 11_B Service request output SR3 is selected.
0	0, 12:10, 31:16	r	**Reserved;** Returns 0 if read; should be written with 0.

Kernel State Control Sensitivity Register

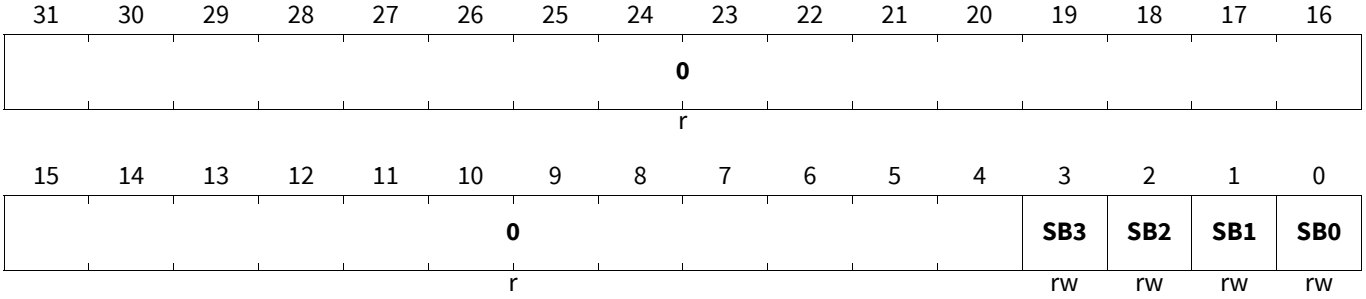
The kernel state control sensitivity register bits define which internal block is affected by Stop Modes 0 and 1, as described in section **Soft Suspend Mode**.

The Kernel State Control Sensitivity Register (KSCSR) is cleared by Debug Reset.

The register can only be written while the OCDS is enabled (OCDS enable = ´1´).

KSCSR

Kernel State Control Sensitivity Register (001C$_H$) Reset Value: Table 207



Field	Bits	Type	Description
SBx (x=0-3)	x	rw	**Sensitivity Block x SBx (x=0,1,2,3)** This bit defines if block x of the CCU6 kernel is sensitive to Stop Mode 0 or Stop Mode 1. The functional definition of the blocks is given in Table 203. 0_B Block x is not sensitive to Stop Mode 0 or Stop Mode 1. It continues normal operation without respecting the defined stop condition. 1_B Block x is sensitive to Stop Mode 0 or Stop Mode 1. It is respecting the defined stop condition.
0	31:4	r	**Reserved;** Returns 0 if read; should be written with 0.

Table 207 Reset Values of KSCSR

Reset Type	Reset Value	Note
Debug Reset	0000 0000$_H$	

Module Output Trigger Selection

Module CCU6 outputs 3 trigger signals (TRIG0, TRIG1, TRIG2) which can be controlled by output signals of the CCU60 and CCU61 kernels. Register **MOSEL** selects the sources of these trigger signals.

Note: To avoid unintended trigger events in the EVADC when modifying bitfields TRIGiSEL, make sure that the respective trigger target is enabled only after selecting the trigger source via register MOSEL.

These trigger signals are connected to the EVADC module.

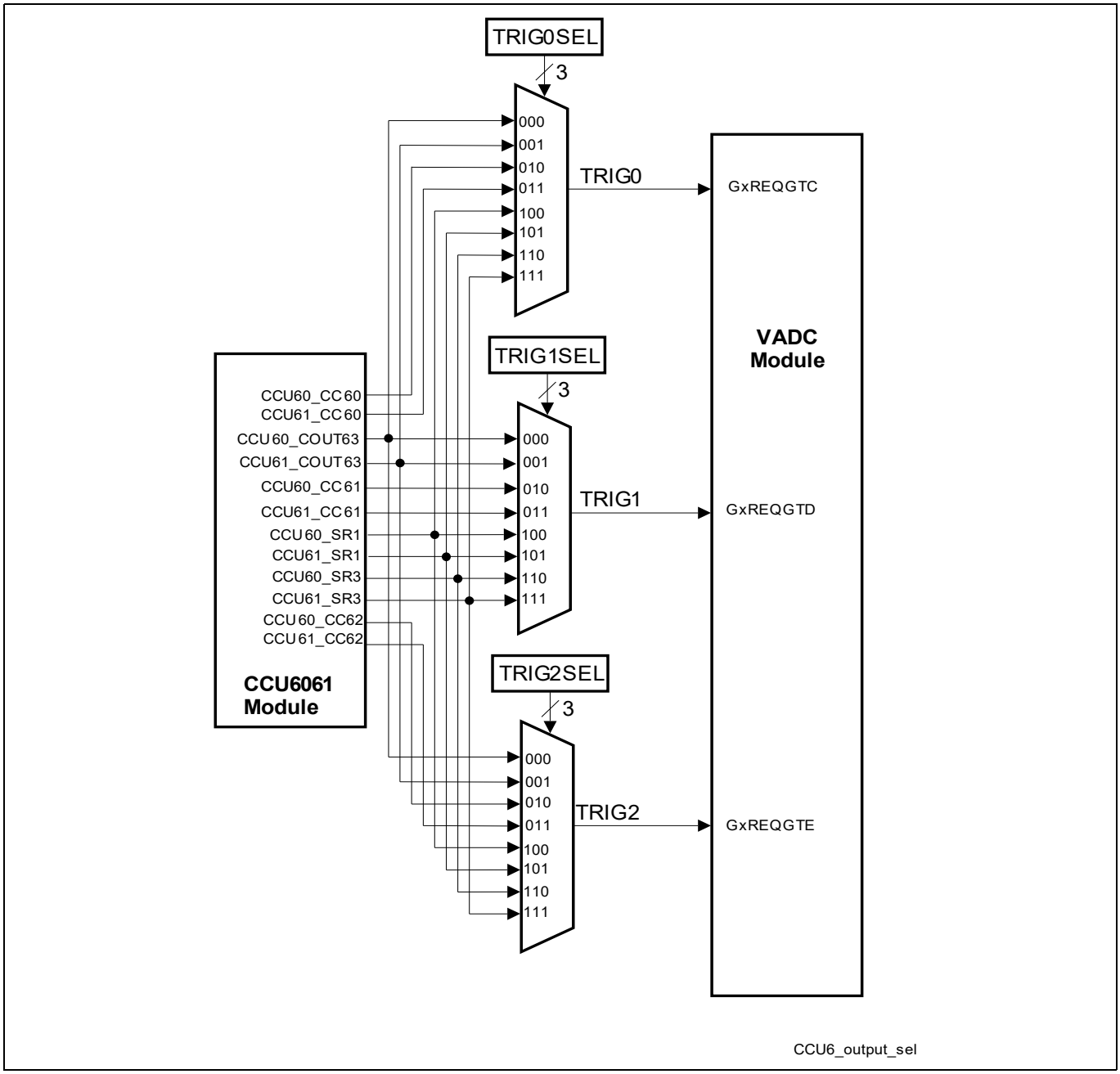


Figure 199 Output Selection for Trigger Signals

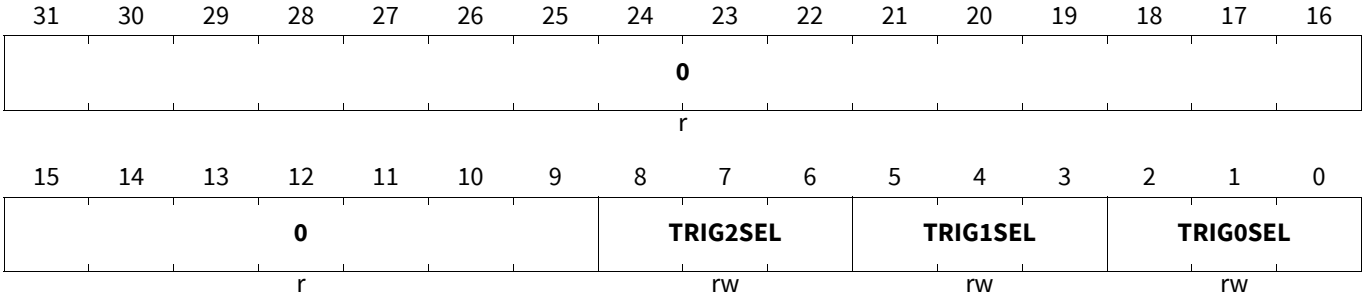
CCU60 Module Output Select Register

MOSEL contains bit fields to select the output signal from module CCU6061 for the trigger signals to the A/D converters in the EVADC module.

*Note: CCU60_MOSEL is located in the address space of kernel CCU60. Therefore, when a reset of kernel CCU60 is triggered via registers CCU60_**KRST0** and CCU60_**KRST1**, register CCU60_MOSEL is also reset. CCU60_MOSEL is **not** reset when a reset of kernel CCU61 is triggered via registers CCU61_KRST0 and CCU61_KRST1. Because CCU60_MOSEL controls both signals from kernel CCU60 and CCU61, the output signals TRIG0 2 will be set to their inactive levels during **any** kernel reset of CCU60 as well as of CCU61 (see also **Figure 199**). Depending on the application, it may make sense to always reset both kernels in this case.*

MOSEL

CCU60 Module Output Select Register (000C$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TRIGxSEL (x=0-2)	3*x+2:3*x	rw	**Output Trigger Select for CCU6061 TRIGx** This bit field defines the output signal from the CCU6061 module used as the trigger signal to the EVADC inputs. 000_B The signal CCU60_COUT63 is selected. 001_B The signal CCU61_COUT63 is selected. 010_B The signal CCU60_CC6x is selected. 011_B The signal CCU61_CC6x is selected. 100_B The signal CCU60_SR1 is selected. 101_B The signal CCU61_SR1 is selected. 110_B The signal CCU60_SR3 is selected. 111_B The signal CCU61_SR3 is selected.
0	31:9	r	**Reserved** Returns 0 if read, should be written with 0.

29.11.6 System Registers

This section describes the registers of the BPI (Bus Peripheral Interface).

Note: Register bits marked "r" in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. Register CLC controls the module clock signal and the reactivity to the sleep signal.

Note: Upon an accepted Sleep Mode request (with EDIS = '1'), or upon a disable request (DISR = '1'), the CCU6 kernel clock is switched off immediately. Therefore, software should ensure that the system controlled by the CCU6 kernel has reached a safe state before triggering a Sleep Mode or module disable request.

CLC
Clock Control Register **(0000_H)** **Application Reset Value: $0000\ 0003_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested. 1_B Module disable is requested.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module. 0_B Module is enabled. 1_B Module is disabled.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode. 0_B Sleep Mode request is regarded. Module is enabled to go into Sleep Mode. 1_B Sleep Mode request is disregarded: Sleep Mode cannot be entered upon a request.
0	2, 15:4, 31:16	r	**Reserved** Read as 0; should be written with 0.

It is recommended not to write to or read from module registers (except CLC) while the module is disabled. Write operations and read operations from registers that require a clock will generate a bus error.

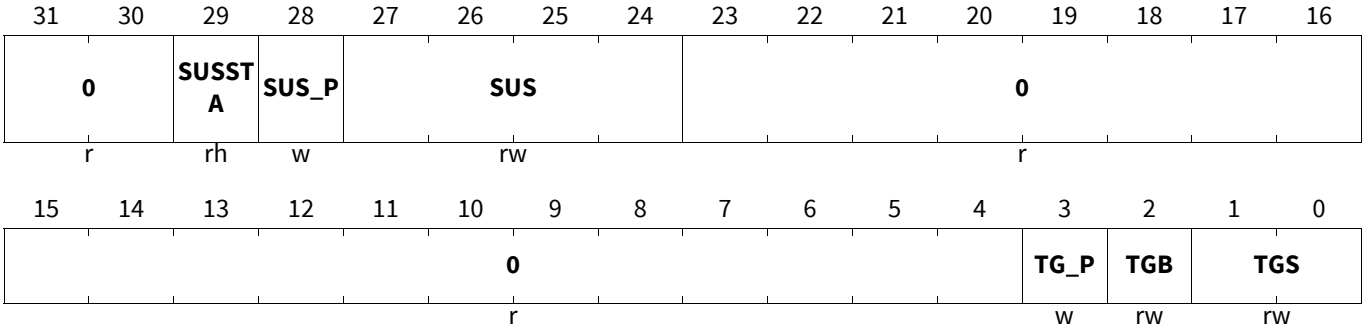
OCDS Control and Status Register

The OCDS Control and Status register OCS controls the module's behavior in suspend mode (used for debugging). The OCS register is cleared by Debug Reset.

The OCS register can only be written when the OCDS is enabled. When OCDS is disabled the OCS suspend control is ineffective.

The default state of the OTGB bus outputs of a peripheral is disabled (all bits are 0). This allows to have an ORed combination of all peripheral OTGBs within the OTGM instead of a multiplexer.

OCS
OCDS Control and Status Register **($00E8_H$)** **Reset Value: Table 209**



Field	Bits	Type	Description
TGS	1:0	rw	**Trigger Set for OTGB0/1** 00_B No Trigger Set output 01_B Trigger Set TS16_CCU6 (see **Table 204**) **others**, reserved
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1
TG_P	3	w	**TGS, TGB Write Protection** TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) Effects of soft suspend options on CCU6 Functional Blocks are described in section **Soft Suspend Mode** 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. 2_H Soft suspend, Stop Mode 0 3_H Soft suspend, Stop Mode 1 **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended

Field	Bits	Type	Description
0	23:4, 31:30	r	**Reserved** Read as 0; must be written with 0.

Table 208 Access Mode Restrictions of OCS sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **SUS_P**	rw	SUS	set SUS_P during write access
write 1 to **TG_P**	rw	TGB, TGS	set TG_P during write access
(default)	r	SUS, TGB, TGS	read only

Table 209 Reset Values of OCS

Reset Type	Reset Value	Note
Debug Reset	$0000\ 0000_H$	

Note: For additional hints, refer to ***"OCDS Suspend" on Page 84****.*

Access Enable Register 0

The Access Enable Register 0 controls write access[1] for transactions with the on chip bus master TAG ID $00\ 0000_B$ to $01\ 1111_B$ (see On Chip Bus chapter for the mapping of products TAG ID <-> master peripheral). Register ACCEN0 provides one enable bit for each possible TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENy: EN0 -> TAG ID $00\ 0000_B$, EN1 -> TAG ID $00\ 0001_B$, … , EN31 -> TAG ID $01\ 1111_B$ (TAG IDs $1X\ XXXX_B$, are not used.

ACCEN0
Access Enable Register 0 **($00FC_H$)** **Application Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENy (y=0-31)	y	rw	**Access Enable for Master TAG ID y** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID y 0_B No write access Write access will not be executed 1_B Write access will be executed

1) The BPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

Individual Module Reset

The Kernel Reset Registers **KRST0**/**KRST1** are used to reset the related module kernel. Kernel Reset Registers 0 and 1 each include bit RST. To reset a module kernel it is necessary to set the RST bits in both Kernel Reset Registers. They will be cleared automatically with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set in the same clock cycle the RST bit is cleared. This bit indicates that a kernel reset was processed. Bit RSTSTAT can be cleared by setting bit CLR in register **KRSTCLR**.

A module kernel reset has the following effects:

Table 210 Effects of a Module Kernel Reset

Register	Executed Action
CLC, OCS, ACCEN0	No influence
ID	No influence
KRST0	Bit RST is cleared automatically after reset execution, RSTSTAT indicates a module kernel reset, cleared via KRSTCLR
KRST1	Bit RST is cleared automatically after reset execution
KRSTCLR	No influence
Other registers	Reset to their defined reset values

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.
*A kernel reset, initiated via registers KRST0/KRST1, does not affect the port logic. Because register **PSLR** is set to its default value, the passive level 0 is selected for the outputs CC60…CC62 and COUT60…COUT63 upon a kernel reset. Therefore, software must ensure appropriate levels for the external system at the port pins in case they are different from the default values selected via PSLR.*

Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers related to the module kernel. The RST bit will be re-set (cleared to ‘0’) by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI in the same clock cycle the RST bit is re-set by the BPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the CLR bit in the related KRSTCLR register.

Notes

1. *During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge*
2. *A kernel reset, initiated via registers KRST0/KRST1, does not affect the port logic. Because register **PSLR** is set to its default value, the passive level 0 is selected for the outputs CC60…CC62 and COUT60…COUT63 upon a kernel reset. Therefore, software must ensure appropriate levels for the external system at the port pins in case they are different from the default values selected via PSLR.*

KRST0

Kernel Reset Register 0 **(00F4$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RSTSTAT	RST
r														rh	rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set after the execution of a kernel reset in the same clock cycle in which the reset bits are cleared. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers (KRSTx1.RST and KRSTx0.RST) related to the module kernel that should be reset. The RST bit will be re-set (cleared to ‘0’) by the BPI with the end of the BPI kernel reset sequence.

KRST1

Kernel Reset Register 1 **(00F0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															RST
r															rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear Register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **($00EC_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**CLR**
r															w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

29.12 Revision History

Table 211 Revision History

Reference	Change to Previous Version	Comment
V3.0.0		
Page 100	Describe details of the module kernel reset	
Page 98	Add reference to register OCS	
Page 97	Add info concerning access to register CLC	

30 General Purpose Timer Unit (GPT12)

The General Purpose Timer Unit blocks GPT1 and GPT2 have very flexible multifunctional timer structures which may be used for timing, event counting, pulse width measurement, pulse generation, frequency multiplication, and other purposes.

They incorporate five 16-bit timers that are grouped into the two timer blocks GPT1 and GPT2. Each timer in each block may operate independently in a number of different modes such as Gated Timer or Counter Mode, or may be concatenated with another timer of the same block.

Each block has alternate input/output functions and specific interrupts (service requests) associated with it. Input signals can be selected from several sources by register PISEL.

The GPT12 module is clocked with clock f_{GPT}.

30.1 Feature List

Timer Block GPT1 contains three timers/counters: The core timer T3 and the two auxiliary timers T2 and T4. The maximum resolution is $f_{GPT}/4$. The auxiliary timers of GPT1 may optionally be configured as reload or capture registers for the core timer. These registers are listed in **Section 30.2.6**.

The following list summarizes the supported features:

- $f_{GPT}/4$ maximum resolution
- 3 independent timers/counters
- Timers/counters can be concatenated
- 4 operating modes:
  - Timer Mode
  - Gated Timer Mode
  - Counter Mode
  - Incremental Interface Mode
- Reload and Capture functionality
- Separate interrupts

Timer Block GPT2 contains two timers/counters: The core timer T6 and the auxiliary timer T5. The maximum resolution is $f_{GPT}/2$. An additional Capture/Reload register (CAPREL) supports capture and reload operation with extended functionality. These registers are listed in **Section 30.3.7**.

The following list summarizes the supported features:

- $f_{GPT}/2$ maximum resolution
- 2 independent timers/counters
- Timers/counters can be concatenated
- 3 operating modes:
  - Timer Mode
  - Gated Timer Mode
  - Counter Mode
- Extended capture/reload functions via 16-bit capture/reload register CAPREL
- Separate interrupts

You will find the following major sections within this chapter:

“GPT12 Kernel Register Overview” on Page 53, **“General Module Operation” on Page 55**

“Implementation of the GPT12 Module” on Page 63, **“Revision History” on Page 65**.

30.2 Timer Block GPT1

All three timers of block GPT1 (T2, T3, T4) can run in one of 4 basic modes: Timer Mode, Gated Timer Mode, Counter Mode, or Incremental Interface Mode. All timers can count up or down. Each timer of GPT1 is controlled by a separate control register TxCON.

Each timer has an input pin TxIN (alternate pin function) associated with it, which serves as the gate control in Gated Timer Mode, or as the count input in Counter Mode. The count direction (up/down) may be programmed via software or may be dynamically altered by a signal at the External Up/Down control input TxEUD (alternate pin function). An overflow/underflow of core timer T3 is indicated by the Output Toggle Latch T3OTL, whose state may be output on the associated pin T3OUT (alternate pin function). The auxiliary timers T2 and T4 may additionally be concatenated with the core timer T3 (through T3OTL) or may be used as capture or reload registers for the core timer T3.

The current contents of each timer can be read or modified by the CPU by accessing the corresponding timer count registers T2, T3, or T4. When any of the timer registers is written to by the CPU in the state immediately preceding a timer increment, decrement, reload, or capture operation, the CPU write operation has priority in order to guarantee correct results.

The interrupt requests of GPT1 are signalled on service request lines SR0, SR1, and SR2.

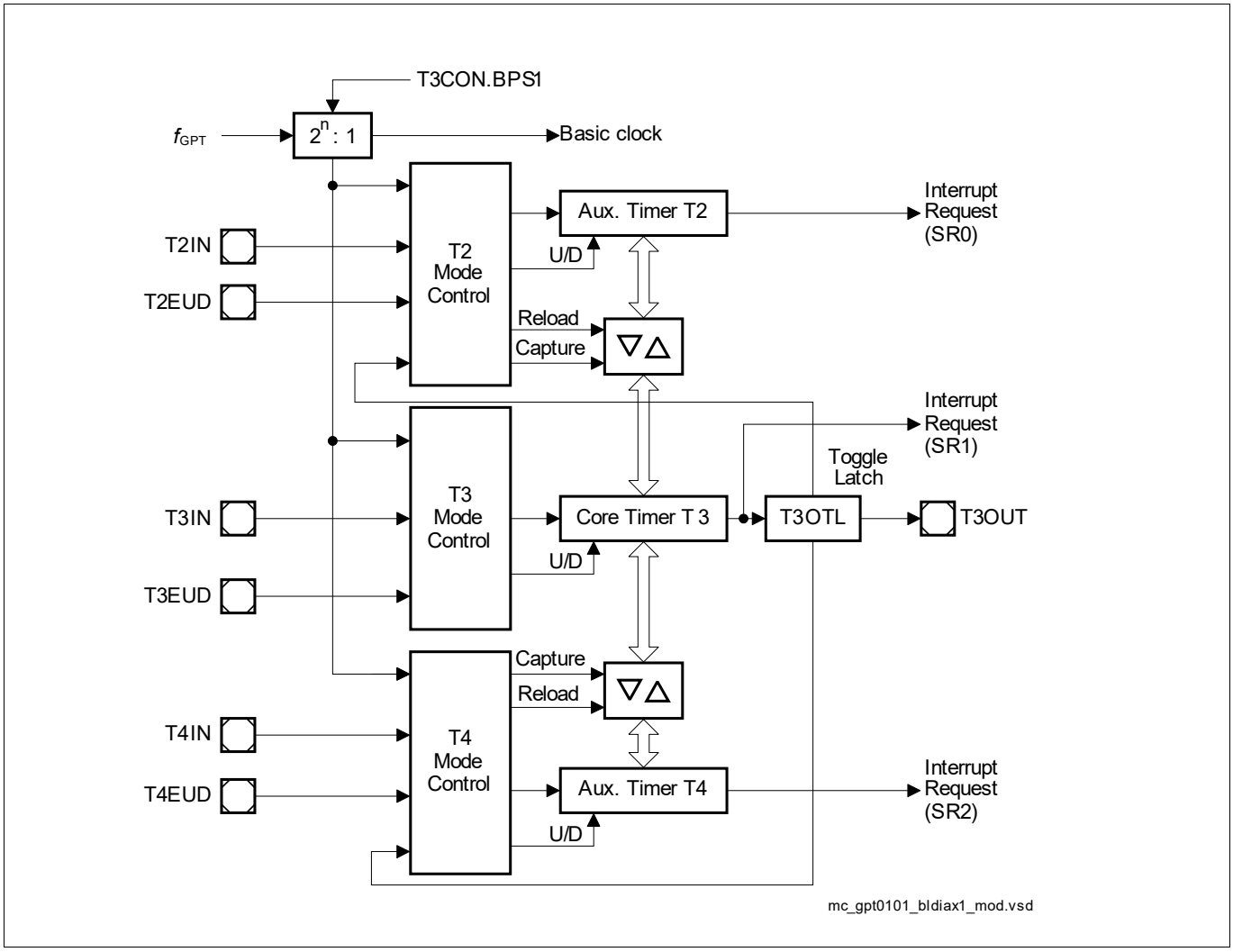


Figure 200 GPT1 Block Diagram

The input and output lines of GPT1 are connected to pins. The control registers for the port functions are located in the respective port modules.

*Note: The timing requirements for external input signals can be found in **Section 30.2.5**, **Section 30.6** summarizes the module interface signals, including pins and interrupt request signals.*

30.2.1 GPT1 Core Timer T3 Control

The current contents of the core timer T3 are reflected by its count register T3. This register can also be written to by the CPU, for example, to set the initial start value.

The core timer T3 is configured and controlled via its control register T3CON.

Timer T3 Run Control

The core timer T3 can be started or stopped by software through bit T3R (Timer T3 Run Bit). This bit is relevant in all operating modes of T3. Setting bit T3R will start the timer, clearing bit T3R stops the timer.

In Gated Timer Mode, the timer will only run if T3R = 1 and the gate is active (high or low, as programmed).

Note: When bit T2RC or T4RC in timer control register T2CON or T4CON is set, bit T3R will also control (start and stop) the auxiliary timer(s) T2 and/or T4.

Count Direction Control

The count direction of the GPT1 timers (core timer and auxiliary timers) can be controlled either by software or by the external input pin TxEUD (Timer Tx External Up/Down Control Input). These options are selected by bits TxUD and TxUDE in the respective control register TxCON. When the up/down control is provided by software (bit TxUDE = 0), the count direction can be altered by setting or clearing bit TxUD. When bit TxUDE = 1, pin TxEUD is selected to be the controlling source of the count direction. However, bit TxUD can still be used to reverse the actual count direction, as shown in **Table 217**. The count direction can be changed regardless of whether or not the timer is running.

Note: When pin TxEUD is used as external count direction control input, it must be configured as input.

Timer T3 Output Toggle Latch

The overflow/underflow signal of timer T3 is connected to a block named ‘Toggle Latch’, shown in the Timer Mode diagrams. **Figure 201** illustrates the details of this block. An overflow or underflow of T3 will clock two latches: The first latch represents bit T3OTL in control register T3CON. The second latch is an internal latch toggled by T3OTL’s output. Both latch outputs are connected to the input control blocks of the auxiliary timers T2 and T4. The output level of the shadow latch will match the output level of T3OTL, but is delayed by one clock cycle. When the T3OTL value changes, this will result in a temporarily different output level from T3OTL and the shadow latch, which can trigger the selected count event in T2 and/or T4.

When software writes to T3OTL, both latches are set or cleared simultaneously. In this case, both signals to the auxiliary timers carry the same level and no edge will be detected. Bit T3OE (overflow/underflow output enable) in register T3CON enables the state of T3OTL to be monitored via an external pin T3OUT. When T3OTL is linked to an external port pin (must be configured as output), T3OUT can be used to control external HW. If T3OE = 1, pin T3OUT outputs the state of T3OTL. If T3OE = 0, pin T3OUT outputs a high level (as long as the T3OUT alternate function is selected for the port pin).

The trigger signals can serve as an input for the counter function or as a trigger source for the reload function of the auxiliary timers T2 and T4.

As can be seen from **Figure 201**, when latch T3OTL is modified by software to determine the state of the output line, also the internal shadow latch is set or cleared accordingly. Therefore, no trigger condition is detected by T2/T4 in this case.

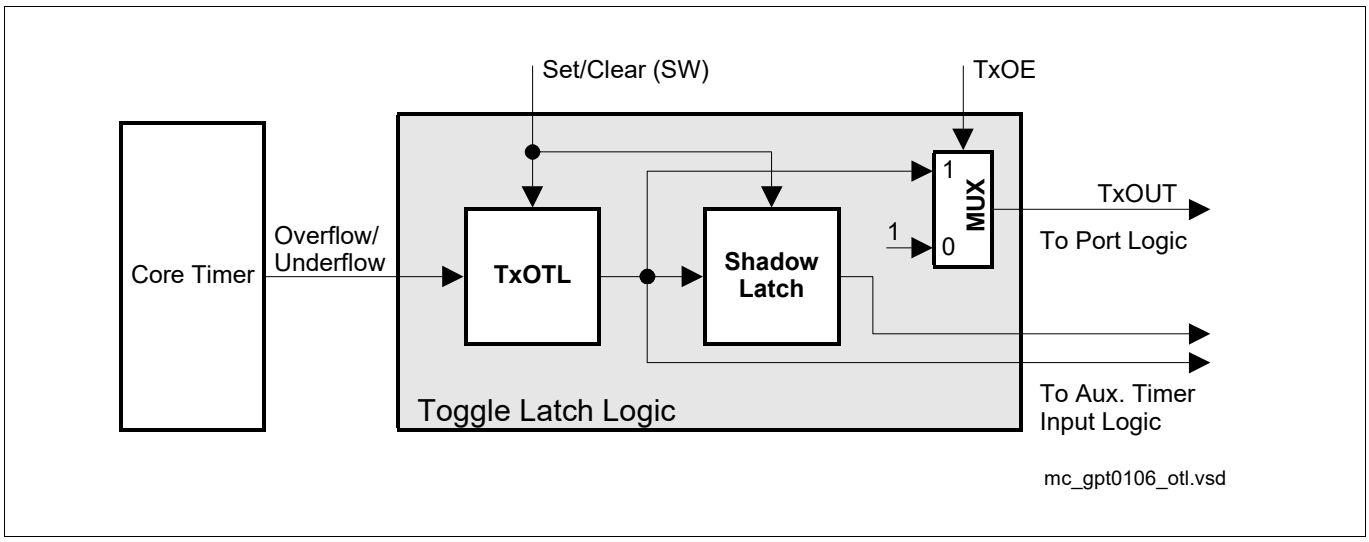


Figure 201 Block Diagram of the Toggle Latch Logic of Core Timer T3 (x = 3)

30.2.2 GPT1 Core Timer T3 Operating Modes

Timer T3 can operate in one of several modes.

Timer T3 in Timer Mode

Timer Mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 000_B. In Timer Mode, T3 is clocked with the module's input clock f_{GPT} divided by two programmable prescalers controlled by bitfields BPS1 and T3I in register T3CON. Please see **Section 30.2.5** for details on the input clock options.

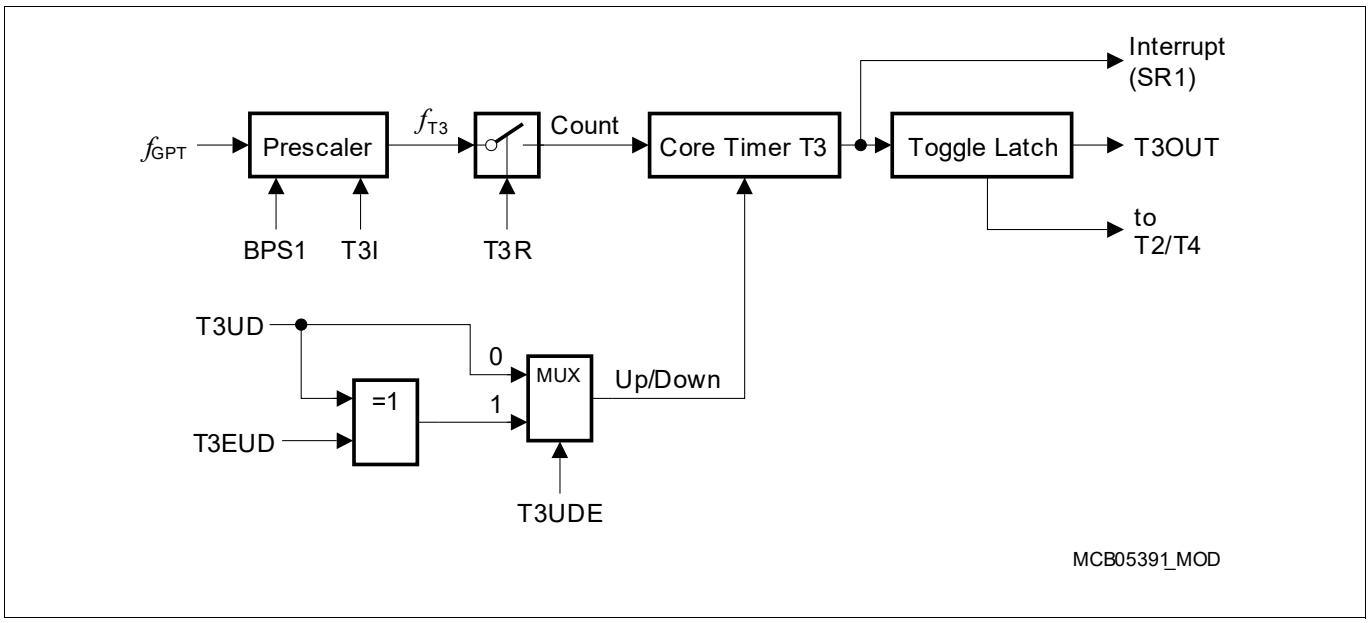


Figure 202 Block Diagram of Core Timer T3 in Timer Mode

Timer T3 in Gated Timer Mode

Gated Timer Mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 010_B or 011_B. Bit T3M.0 (T3CON.3) selects the active level of the gate input. The same options for the input frequency are available in Gated Timer Mode as in Timer Mode (see **Section 30.2.5**). However, the input clock to the timer in this mode is gated by the external input pin T3IN (Timer T3 External Input).

To enable this operation, the associated pin T3IN must be configured as input.

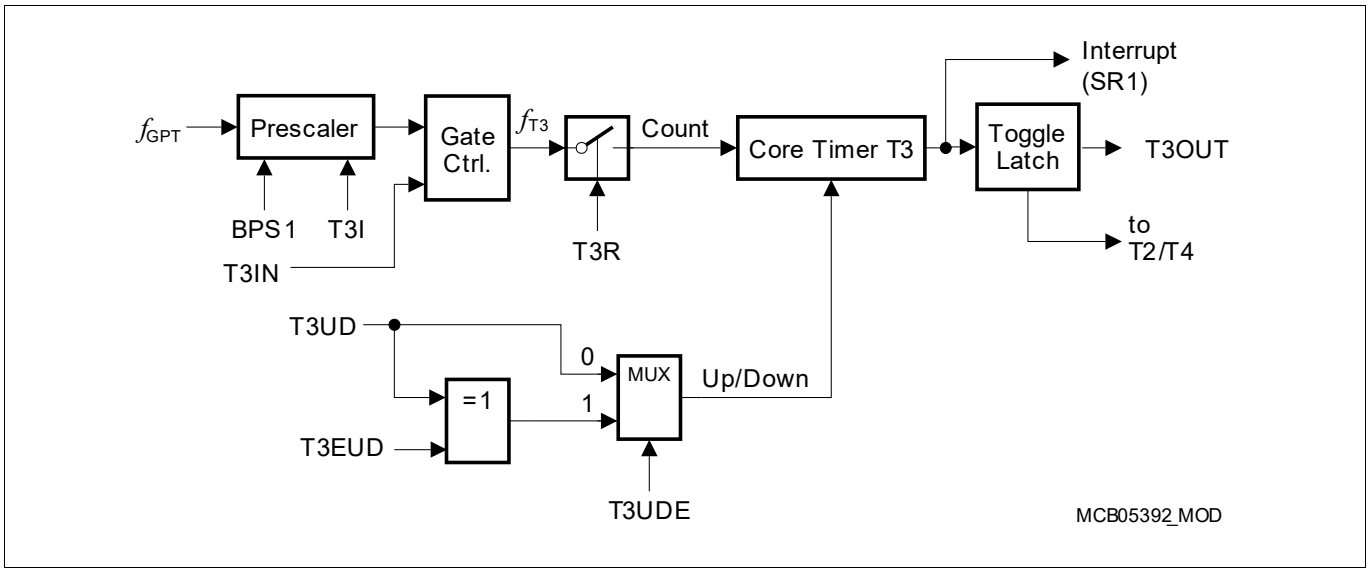


Figure 203 Block Diagram of Core Timer T3 in Gated Timer Mode

If T3M = 010_B, the timer is enabled when T3IN shows a low level. A high level at this line stops the timer. If T3M = 011_B, line T3IN must have a high level in order to enable the timer. Additionally, the timer can be turned on or off by software using bit T3R. The timer will only run if T3R is 1 and the gate is active. It will stop if either T3R is 0 or the gate is inactive.

Note: A transition of the gate signal at pin T3IN does not cause a service request via SR1.

Timer T3 in Counter Mode

Counter Mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 001_B. In Counter Mode, timer T3 is clocked by a transition at the external input pin T3IN. The event causing an increment or decrement of the timer can be a positive, a negative, or both a positive and a negative transition at this line. Bitfield T3I in control register T3CON selects the triggering transition (see **Table 219**).

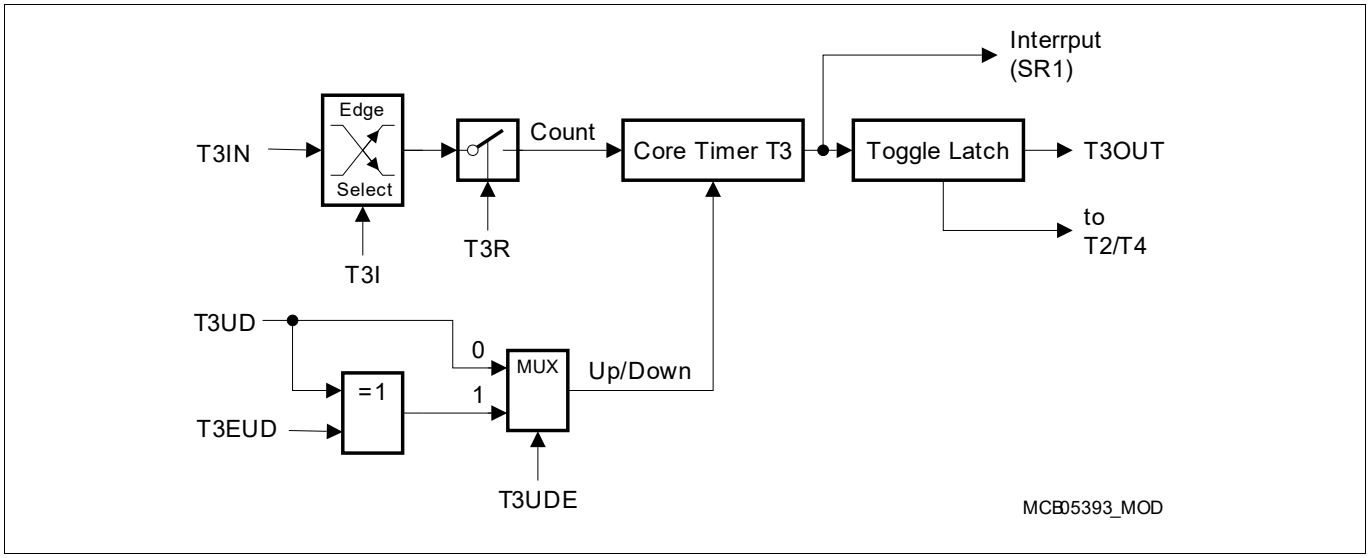


Figure 204 Block Diagram of Core Timer T3 in Counter Mode

For Counter Mode operation, pin T3IN must be configured as input. The maximum input frequency allowed in Counter Mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to T3IN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 30.2.5**.

Timer T3 in Incremental Interface Mode

Incremental Interface Mode for the core timer T3 is selected by setting bitfield T3M in register T3CON to 110_B or 111_B. In Incremental Interface Mode, the two inputs associated with core timer T3 (T3IN, T3EUD) are used to interface to an incremental encoder. T3 is clocked by each transition on one or both of the external input pins to provide 2-fold or 4-fold resolution of the encoder input.

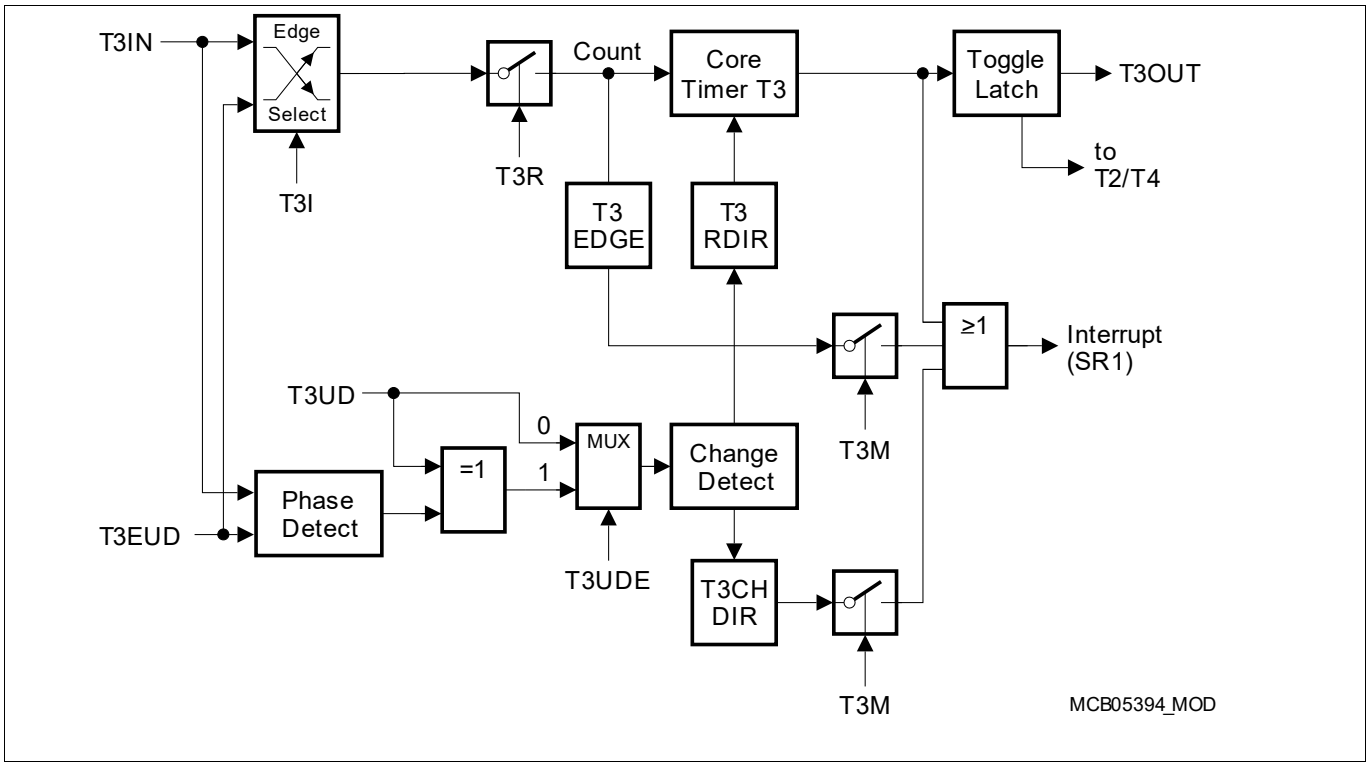


Figure 205 Block Diagram of Core Timer T3 in Incremental Interface Mode

Bitfield T3I in control register T3CON selects the triggering transitions (see **Table 221**). The sequence of the transitions of the two input signals is evaluated and generates count pulses as well as the direction signal. So T3 is modified automatically according to the speed and the direction of the incremental encoder and, therefore, its contents always represent the encoder's current position.

The service request generation can be selected: In Rotation Detection Mode (T3M = 110_B), a service request is generated each time the count direction of T3 changes. In Edge Detection Mode (T3M = 111_B), a service request is generated each time a count edge for T3 is detected. Count direction, changes in the count direction, and count requests are monitored by status bits T3RDIR, T3CHDIR, and T3EDGE in register T3CON.

The incremental encoder can be connected directly to the without external interface logic. In a standard system, however, comparators will be employed to convert the encoder's differential outputs (such as A, $\overline{A}$) to digital signals (such as A). This greatly increases noise immunity.

Note: The third encoder output T0, that indicates the mechanical zero position, may be connected to an external interrupt input and trigger a reset of timer T3.
*If input T4IN is available, T0 can be connected there and clear T3 automatically without requiring an interrupt (see bit CLRT3EN in register **T4CON**).*

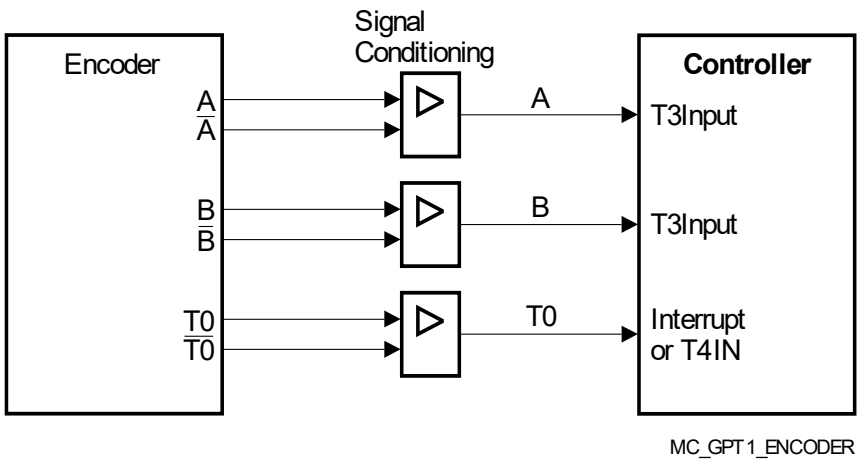


Figure 206 Connection of the Encoder to the AURIX™ TC3xx Platform

For Incremental Interface Mode operation, the following conditions must be met:

- Bitfield T3M must be 110_B or 111_B.
- Both pins T3IN and T3EUD must be configured as input.
- Pin T4IN must be configured as input, if used for T0.
- Bit T3UDE must be 1 to enable automatic external direction control.

The maximum count frequency allowed in Incremental Interface Mode depends on the selected prescaler value. To ensure that a transition of any input signal is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 30.2.5**.

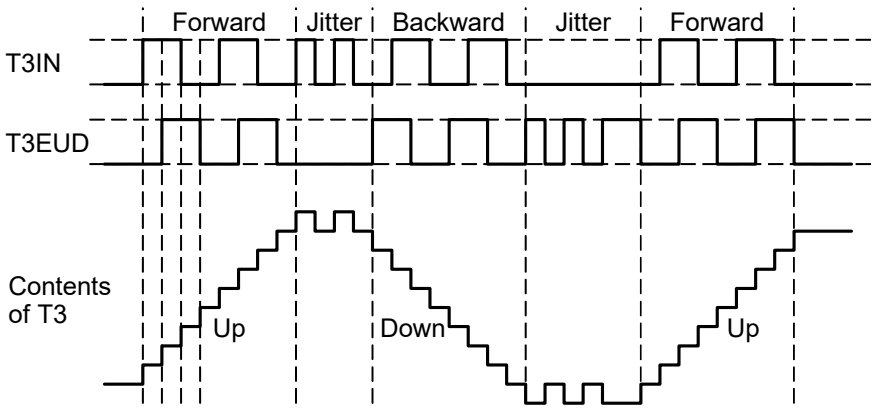
As in Incremental Interface Mode two input signals with a 90° phase shift are evaluated, their maximum input frequency can be half the maximum count frequency.

In Incremental Interface Mode, the count direction is automatically derived from the sequence in which the input signals change, which corresponds to the rotation direction of the connected sensor. **Table 212** summarizes the possible combinations.

Table 212 GPT1 Core Timer T3 (Incremental Interface Mode) Count Direction

Level on Respective other Input	T3IN Input		T3EUD Input	
	Rising ↑	Falling ↓	Rising ↑	Falling ↓
High	Down	Up	Up	Down
Low	Up	Down	Down	Up

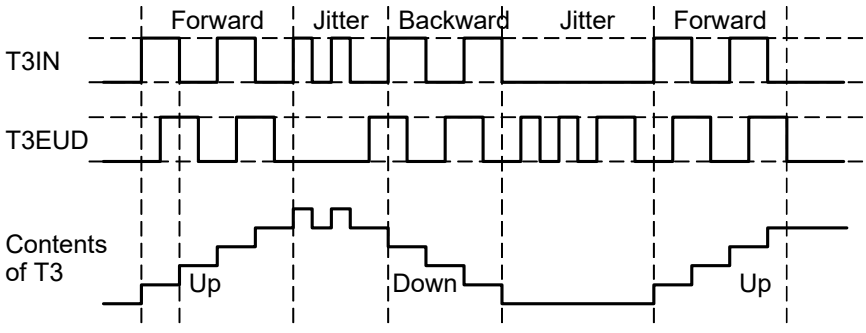
Figure 207 and **Figure 208** give examples of T3's operation, visualizing count signal generation and direction control. They also show how input jitter is compensated, which might occur if the sensor rests near to one of its switching points.



Note: This example shows the timer behaviour assuming that T3 counts upon any transition on any input (T3IN and T3EUD), i.e. T3I = 011_B.

MCT04373

Figure 207 Evaluation of Incremental Encoder Signals, 2 Count Inputs



Note: This example shows the timer behaviour assuming that T3 counts upon any transition on input T3IN, i.e. T3I = 001_B.

MCT04374

Figure 208 Evaluation of Incremental Encoder Signals, 1 Count Input

Note: Timer T3 operating in Incremental Interface Mode automatically provides information on the sensor's current position. Dynamic information (speed, acceleration, deceleration) may be obtained by measuring the incoming signal periods (see ***"Combined Capture Modes" on Page 43****).*

30.2.3 GPT1 Auxiliary Timers T2/T4 Control

Auxiliary timers T2 and T4 have exactly the same functionality. They can be configured for Timer Mode, Gated Timer Mode, Counter Mode, or Incremental Interface Mode with the same options for the timer frequencies and the count signal as the core timer T3. In addition to these 4 counting modes, the auxiliary timers can be concatenated with the core timer, or they may be used as reload or capture registers in conjunction with the core timer. The start/stop function of the auxiliary timers can be remotely controlled by the T3 run control bit. Several timers may thus be controlled synchronously.

The current contents of an auxiliary timer are reflected by its count register T2 or T4, respectively. These registers can also be written to by the CPU, for example, to set the initial start value.

The individual configurations for timers T2 and T4 are determined by their control registers T2CON and T4CON, that are organized identically. Note that functions which are present in all 3 timers of block GPT1 are controlled in the same bit positions and in the same manner in each of the specific control registers.

Note: The auxiliary timers have no output toggle latch and no alternate output function.

Timer T2/T4 Run Control

Each of the auxiliary timers T2 and T4 can be started or stopped by software in two different ways:

- Through the associated timer run bit (T2R or T4R). In this case it is required that the respective control bit TxRC = 0.
- Through the core timer's run bit (T3R). In this case the respective remote control bit must be set (TxRC = 1).

The selected run bit is relevant in all operating modes of T2/T4. Setting the bit will start the timer, clearing the bit stops the timer.

In Gated Timer Mode, the timer will only run if the selected run bit is set and the gate is active (high or low, as programmed).

Note: If remote control is selected T3R will start/stop timer T3 and the selected auxiliary timer(s) synchronously.

Count Direction Control

The count direction of the GPT1 timers (core timer and auxiliary timers) is controlled in the same way, either by software or by the external input pin TxEUD. Please refer to the description in **Table 217**.

Note: When pin TxEUD is used as external count direction control input, it must be configured as input.

30.2.4 GPT1 Auxiliary Timers T2/T4 Operating Modes

The operation of the auxiliary timers in the basic operating modes is almost identical with the core timer's operation, with very few exceptions. Additionally, some combined operating modes can be selected.

Timers T2 and T4 in Timer Mode

Timer Mode for an auxiliary timer Tx is selected by setting its bitfield TxM in register TxCON to 000_B.

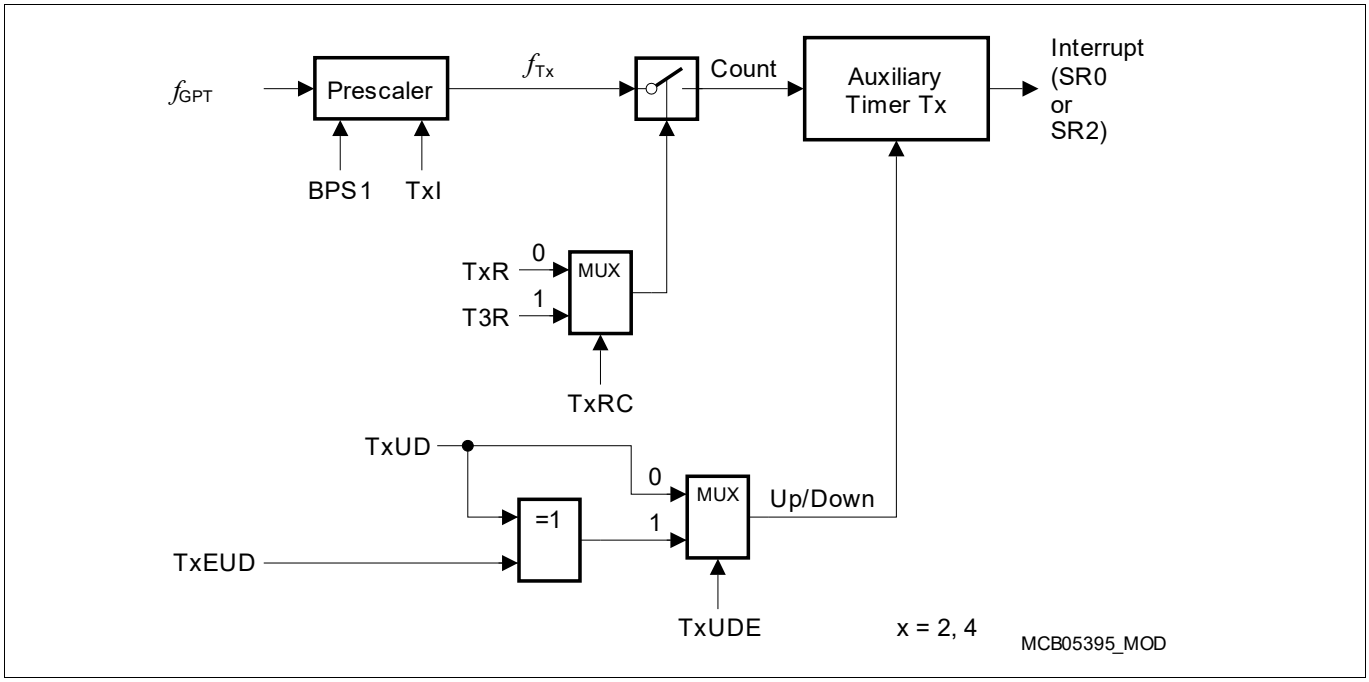


Figure 209 Block Diagram of an Auxiliary Timer in Timer Mode

Timers T2 and T4 in Gated Timer Mode

Gated Timer Mode for an auxiliary timer Tx is selected by setting bitfield TxM in register TxCON to 010_B or 011_B. Bit TxM.0 (TxCON.3) selects the active level of the gate input.

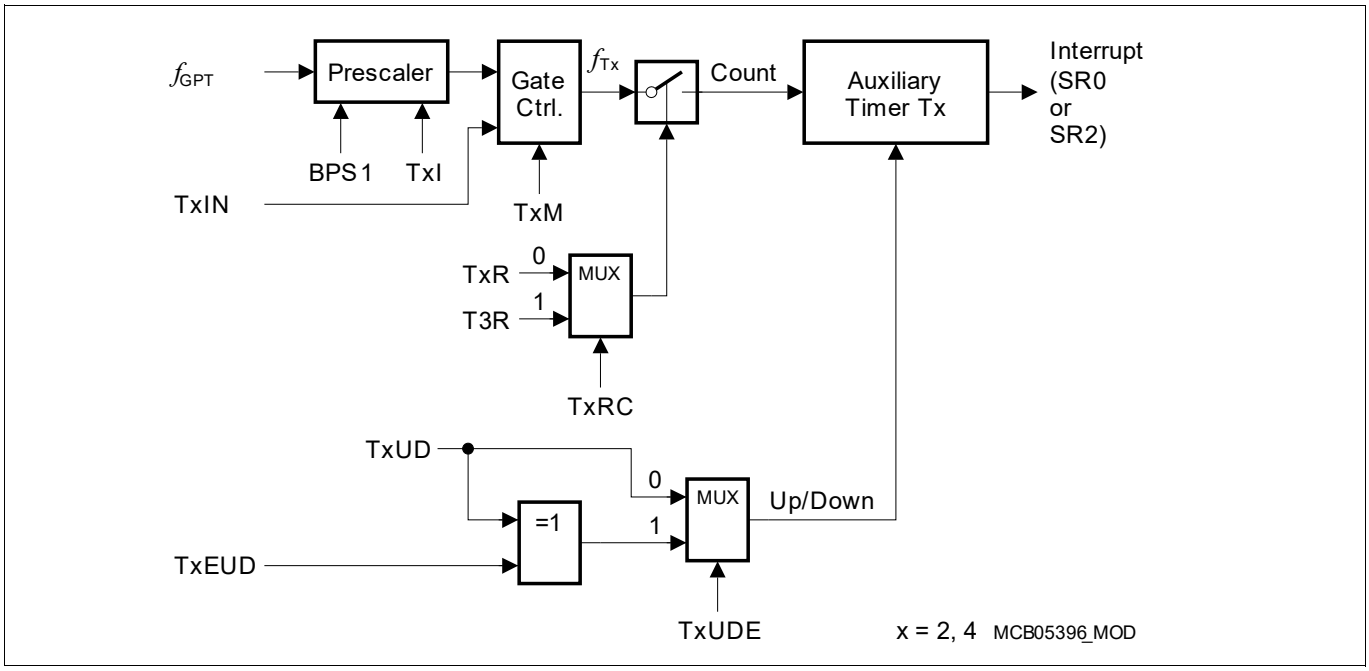


Figure 210 Block Diagram of an Auxiliary Timer in Gated Timer Mode

Note: A transition of the gate signal at TxIN does not cause a service request. Service requests of timer T2 are handled via SR0, and service requests of timer T4 are handled via SR2.
There is no output toggle latch for T2 and T4.
Start/stop of an auxiliary timer can be controlled locally or remotely.

Timers T2 and T4 in Counter Mode

Counter Mode for an auxiliary timer Tx is selected by setting bitfield TxM in register TxCON to 001_B. In Counter Mode, an auxiliary timer can be clocked either by a transition at its external input line TxIN, or by a transition of timer T3's toggle latch T3OTL. The event causing an increment or decrement of a timer can be a positive, a negative, or both a positive and a negative transition at either the respective input pin or at the toggle latch. Bitfield TxI in control register TxCON selects the triggering transition (see **Table 220**).

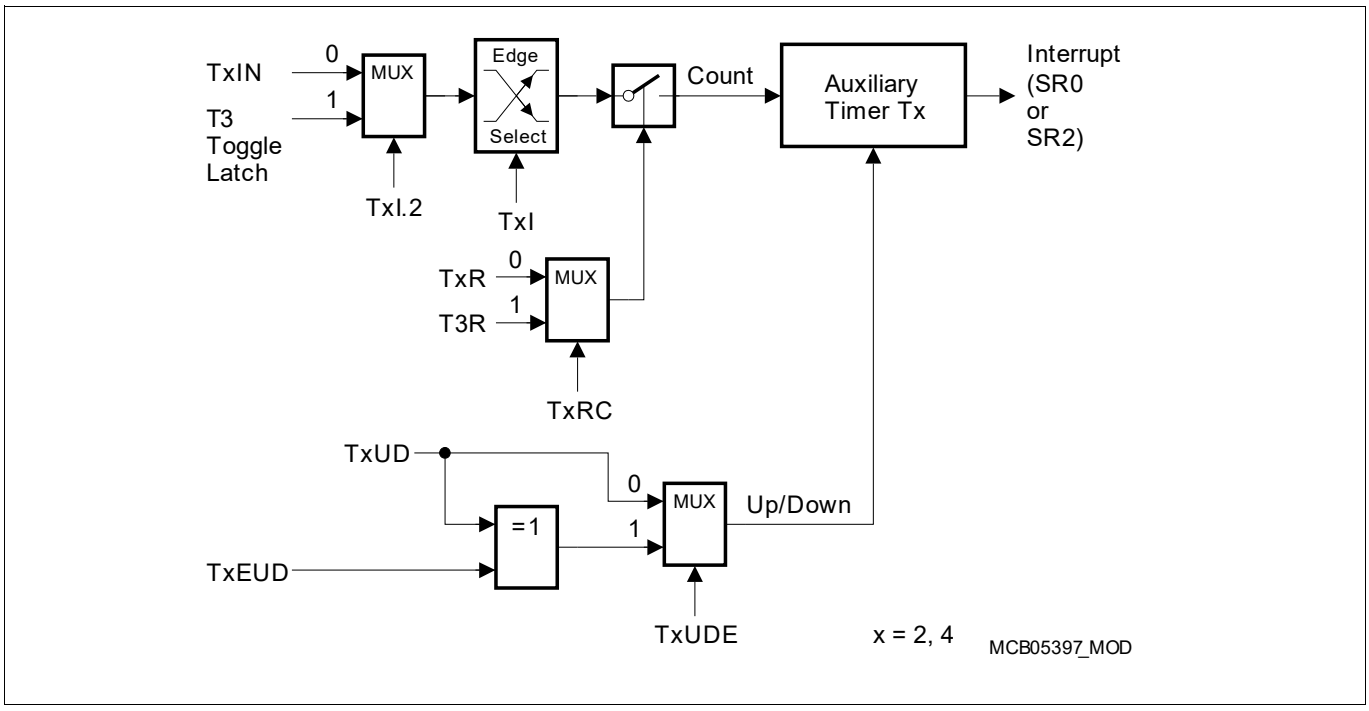


Figure 211 Block Diagram of an Auxiliary Timer in Counter Mode

Note: Only state transitions of T3OTL which are caused by the overflows/underflows of T3 will trigger the counter function of T2/T4. Modifications of T3OTL via software will NOT trigger the counter function of T2/T4.

For counter operation, pin TxIN must be configured as input. The maximum input frequency allowed in Counter Mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to TxIN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 30.2.5**.

Timer Concatenation

Using the toggle bit T3OTL as a clock source for an auxiliary timer in Counter Mode concatenates the core timer T3 with the respective auxiliary timer. This concatenation forms either a 32-bit or a 33-bit timer/counter, depending on which transition of T3OTL is selected to clock the auxiliary timer.

- **32-bit Timer/Counter:** If both a positive and a negative transition of T3OTL are used to clock the auxiliary timer, this timer is clocked on every overflow/underflow of the core timer T3. Thus, the two timers form a 32-bit timer.
- **33-bit Timer/Counter:** If either a positive or a negative transition of T3OTL is selected to clock the auxiliary timer, this timer is clocked on every second overflow/underflow of the core timer T3. This configuration forms a 33-bit timer (16-bit core timer + T3OTL + 16-bit auxiliary timer).
  As long as bit T3OTL is not modified by software, it represents the state of the internal toggle latch, and can be regarded as part of the 33-bit timer.

The count directions of the two concatenated timers are not required to be the same. This offers a wide variety of different configurations.

T3, which represents the low-order part of the concatenated timer, can operate in Timer Mode, Gated Timer Mode or Counter Mode in this case.

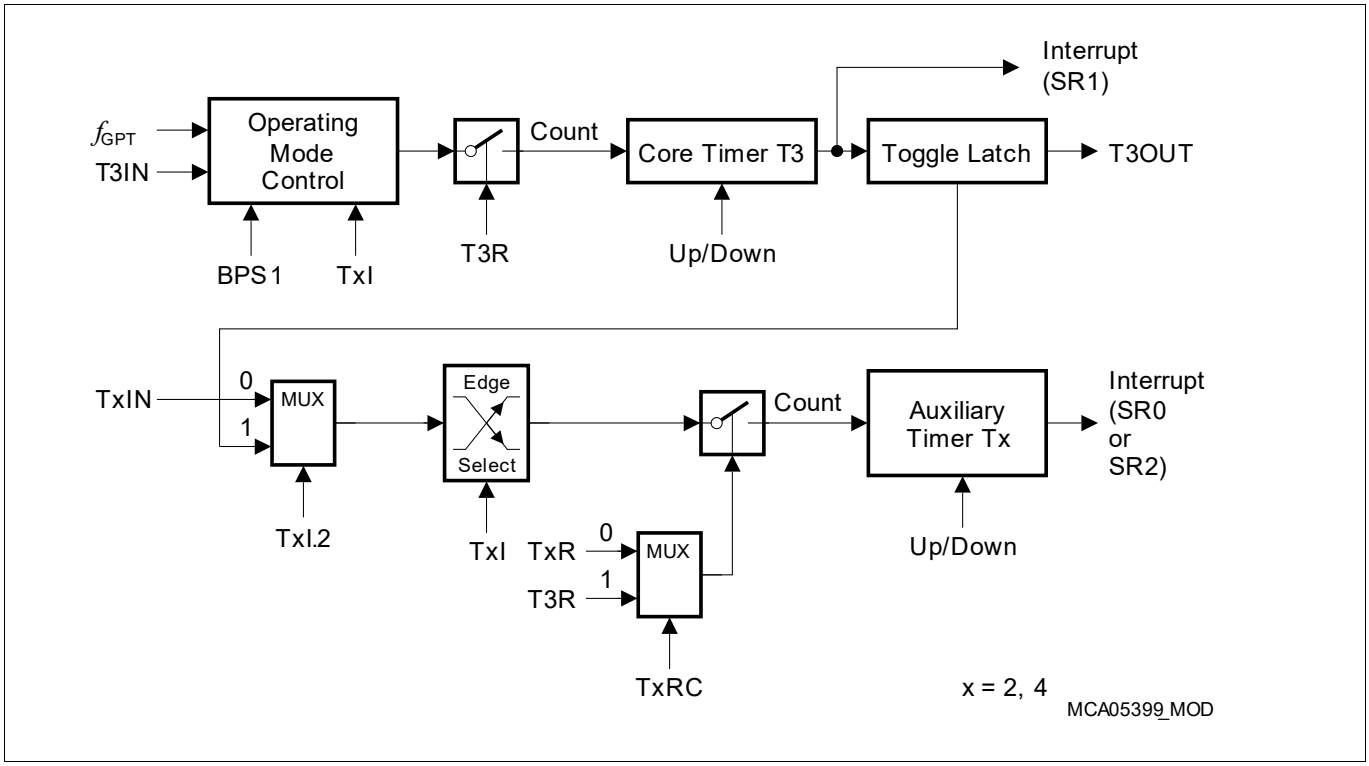


Figure 212 Concatenation of Core Timer T3 and an Auxiliary Timer

When reading the low and high parts of the concatenated timer, care must be taken to obtain consistent values in particular after a timer overflow/underflow (e.g. one part may already have considered an overflow, while the other has not).

Note: This is a general issue when reading multi-word results with consecutive instructions, and not necessarily unique to the GPT12 module architecture.

The following algorithm may be used to read concatenated GPT1 timers, represented by TIMER_HIGH (for auxiliary timer, here T2) and TIMER_LOW (for core timer T3). The high part is read twice, and reading of the low part is repeated if two different values were read for the high part:

- TIMER_HIGH_TMP = T2
- TIMER_LOW = T3
- Wait two basic clock cycles (to allow increment/decrement of auxiliary timer in case of core timer overflow/underflow) - see **Table 213**
- TIMER_HIGH = T2
- If TIMER_HIGH is not equal to TIMER_HIGH_TMP then TIMER_LOW = T3

After execution of this algorithm, TIMER_HIGH and TIMER_LOW represent a consistent time stamp of the concatenated timers.

The equivalent number of module clock cycles corresponding to two basic clock cycles is shown in **Table 213**.

Table 213 Number of Module Clock Cycles to Wait for Two Basic Clock Cycles$_B$

Block Prescaler	BPS1 = 01_B	BPS1 = 00_B	BPS1 = 11_B	BPS1 = 10_B
Number of Module clocks	8	16	32	64

In case the required timer resolution can be achieved with different combinations of the Block Prescaler BPS1 and the Individual Prescalers TxI (see **Table 218**), the variant with the smallest value for the Block Prescaler may be chosen to minimize the waiting time.

Timers T2 and T4 in Incremental Interface Mode

Incremental Interface Mode for an auxiliary timer Tx is selected by setting bitfield TxM in the respective register TxCON to 110_B or 111_B. In Incremental Interface Mode, the two inputs associated with an auxiliary timer Tx (TxIN, TxEUD) are used to interface to an incremental encoder. Tx is clocked by each transition on one or both of the external input pins to provide 2-fold or 4-fold resolution of the encoder input.

The operation of the auxiliary timers T2 and T4 in Incremental Interface Mode and the interrupt generation are the same as described for the core timer T3. The descriptions, figures and tables apply accordingly.

Note: Timers T2 and T4 operating in Incremental Interface Mode automatically provide information on the sensor's current position. For dynamic information (speed, acceleration, deceleration) see ***"Combined Capture Modes" on Page 43****).*

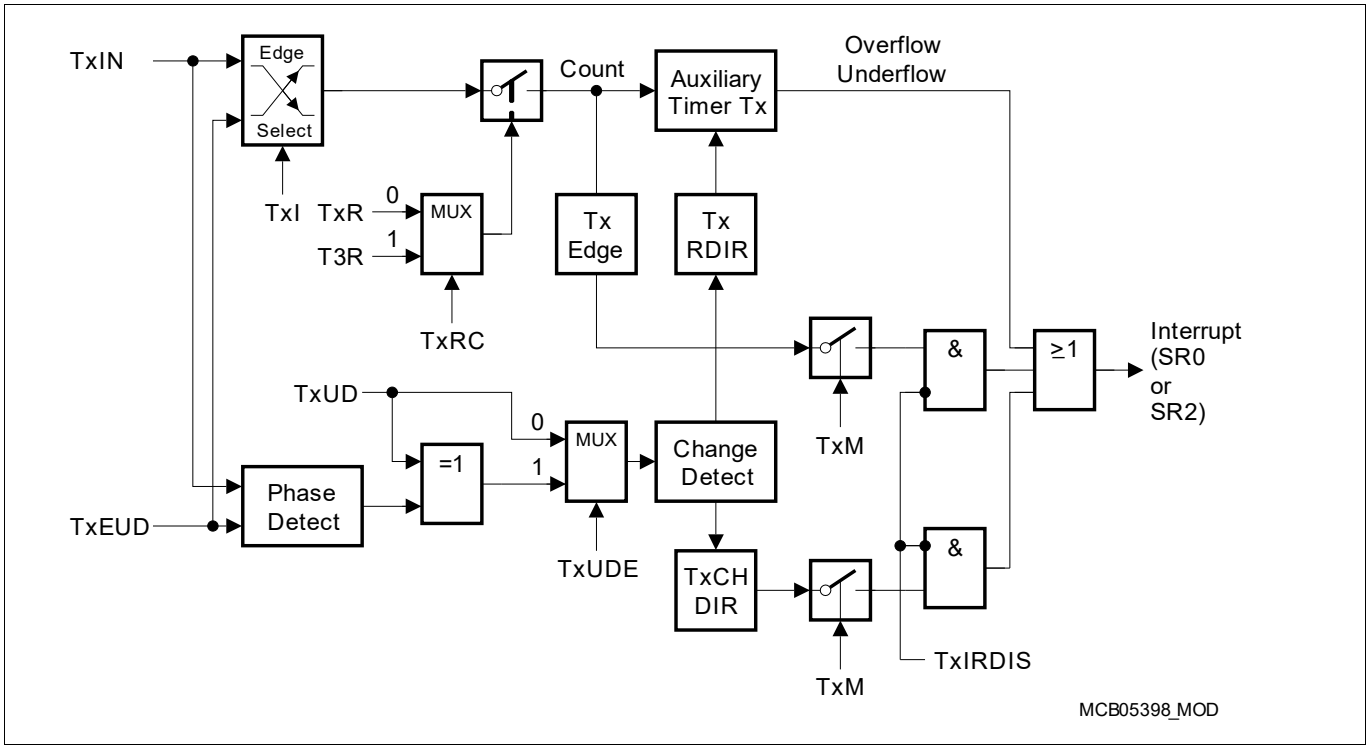


Figure 213 Block Diagram of an Auxiliary Timer in Incremental Interface Mode

Timers T2 and T4 in Reload Mode

Reload Mode for an auxiliary timer Tx is selected by setting bitfield TxM in the respective register TxCON to 100_B. In Reload Mode, the core timer T3 is reloaded with the contents of an auxiliary timer register, triggered by one of two different signals. The trigger signal is selected the same way as the clock source for Counter Mode (see **Table 220**), i.e. a transition of the auxiliary timer's input TxIN or the toggle latch T3OTL may trigger the reload.

Note: When programmed for Reload Mode, the respective auxiliary timer (T2 or T4) stops independently of its run flag T2R or T4R.
The timer input pin TxIN must be configured as input if it shall trigger a reload operation.

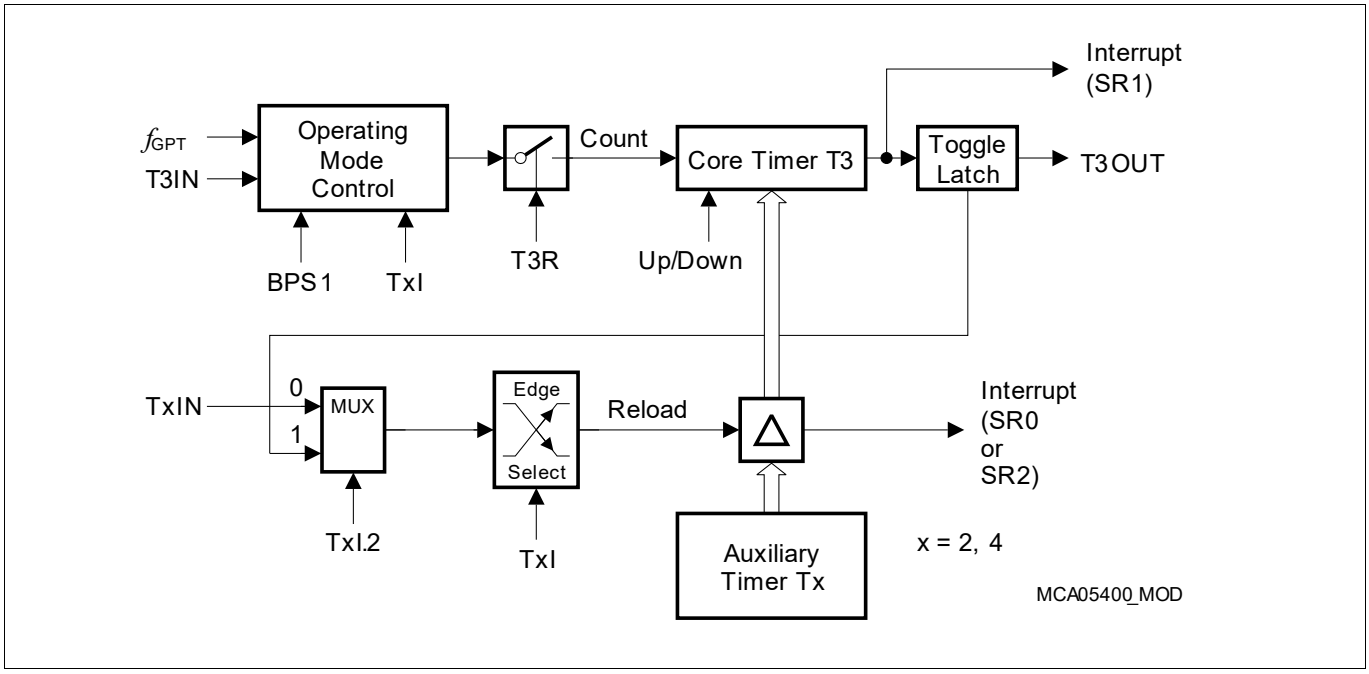


Figure 214 GPT1 Auxiliary Timer in Reload Mode

Upon a trigger signal, T3 is loaded with the contents of the respective timer register (T2 or T4) and the respective service request SR0 or SR2 is activated.

Note: When a T3OTL transition is selected for the trigger signal, service request SR1 will also become active, indicating T3's overflow or underflow. Modifications of T3OTL via software will NOT trigger the counter function of T2/T4.

To ensure that a transition of the reload input signal applied to TxIN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles, detailed in **Section 30.2.5**.

The Reload Mode triggered by the T3 toggle latch can be used in a number of different configurations. The following functions can be performed, depending on the selected active transition:

- If both a positive and a negative transition of T3OTL are selected to trigger a reload, the core timer will be reloaded with the contents of the auxiliary timer each time it overflows or underflows. This is the standard Reload Mode (reload on overflow/underflow).
- If either a positive or a negative transition of T3OTL is selected to trigger a reload, the core timer will be reloaded with the contents of the auxiliary timer on every second overflow or underflow.
- Using this "single-transition" mode for both auxiliary timers allows to perform very flexible Pulse Width Modulation (PWM). One of the auxiliary timers is programmed to reload the core timer on a positive transition

of T3OTL, the other is programmed for a reload on a negative transition of T3OTL. With this combination the core timer is alternately reloaded from the two auxiliary timers.

Figure 215 shows an example for the generation of a PWM signal using the “single-transition” reload mechanism. T2 defines the high time of the PWM signal (reloaded on positive transitions) and T4 defines the low time of the PWM signal (reloaded on negative transitions). The PWM signal can be output on pin T3OUT if T3OE = 1. With this method, the high and low time of the PWM signal can be varied in a wide range.

Note: The output toggle latch T3OTL is accessible via software and may be changed, if required, to modify the PWM signal.
However, this will NOT trigger the reloading of T3.

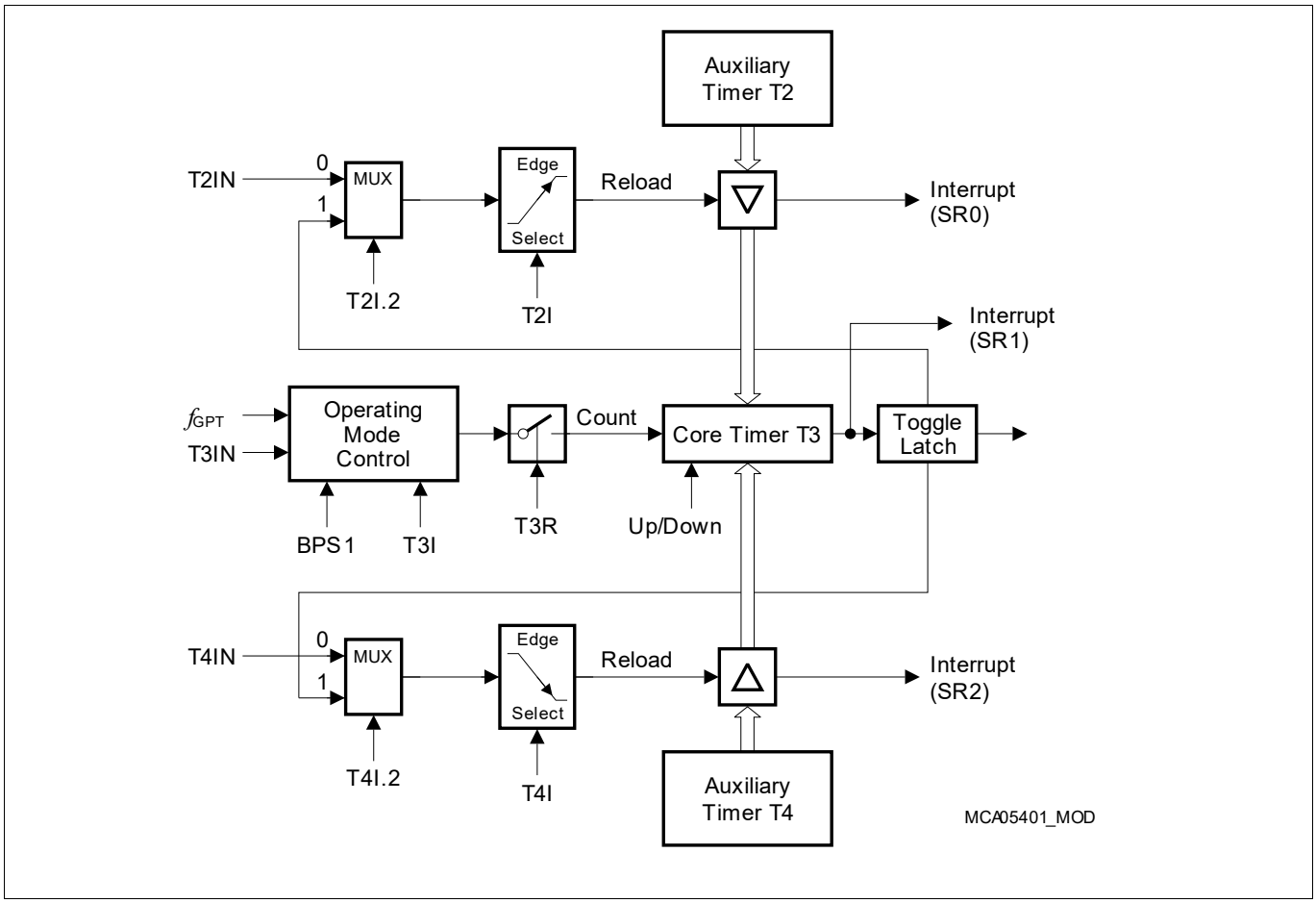


Figure 215 GPT1 Timer Reload Configuration for PWM Generation

Note: Although possible, selecting the same reload trigger event for both auxiliary timers should be avoided. In such a case, both reload registers would try to load the core timer at the same time. If this combination is selected, T2 is disregarded and the contents of T4 is reloaded.

The implementation of the GPT1 finite state machine may require special consideration in following cases.

In Case 1, when T2 or T4 are used to reload T3 on overflow/underflow, and T3 is read by software on the fly, the following unexpected values may be read from T3:

- When T3 is counting up, 0000_H or 0001_H may be read from T3 directly after an overflow, although the reload value in T2/T4 is higher (0001_H may be read in particular if BPS1 = 01_B and T3I = 000_B).
- When T3 is counting down, $FFFF_H$ or $FFFE_H$ may be read from T3 directly after an underflow, although the reload value in T2/T4 is lower ($FFFE_H$ may be read in particular if BPS1 = 01_B and T3I = 000_B).

Note: All timings derived from T3 in this configuration (e.g. distance between service requests, PWM waveform on T3OUT, etc.) are accurate except for the specific case described below.

Recommendation:

- When T3 counts up, and T3_value < reload value is read from T3, T3_value should be replaced with the reload value for further calculations.
- When T3 counts down, and T3_value > reload value is read from T3, T3_value should be replaced with the reload value for further calculations.

Alternatively, if the intention is to identify the overflow/underflow of T3, the T3 service request (SR1) may be used.

In Case 2, when T2 is used to reload T3 in the configuration with BPS1 = 01_B and T3I = 000_B (i.e. fastest configuration/highest resolution of T3), the reload of T3 is performed with a delay of one basic clock cycle.

Recommendation:

To compensate the delay and achieve correct timing,

- Increment the reload value in T2 by 1 when T3 is configured to count up,
- Decrement the reload value in T2 by 1 when T3 is configured to count down.

Alternatively, use T4 instead of T2 as reload register for T3. In this configuration the reload of T3 is not delayed, i.e. the effect described above does not occur with T4.

Timers T2 and T4 in Capture Mode

Capture Mode for an auxiliary timer Tx is selected by setting bitfield TxM in the respective register TxCON to 101_B. In Capture Mode, the contents of the core timer T3 are latched into an auxiliary timer register in response to a signal transition at the respective auxiliary timer's external input pin TxIN. The capture trigger signal can be a positive, a negative, or both a positive and a negative transition.

The two least significant bits of bitfield TxI select the active transition (see **Table 220**). Bit 2 of TxI is irrelevant for Capture Mode and must be cleared (TxI.2 = 0).

Note: When programmed for Capture Mode, the respective auxiliary timer (T2 or T4) stops independently of its run flag T2R or T4R.

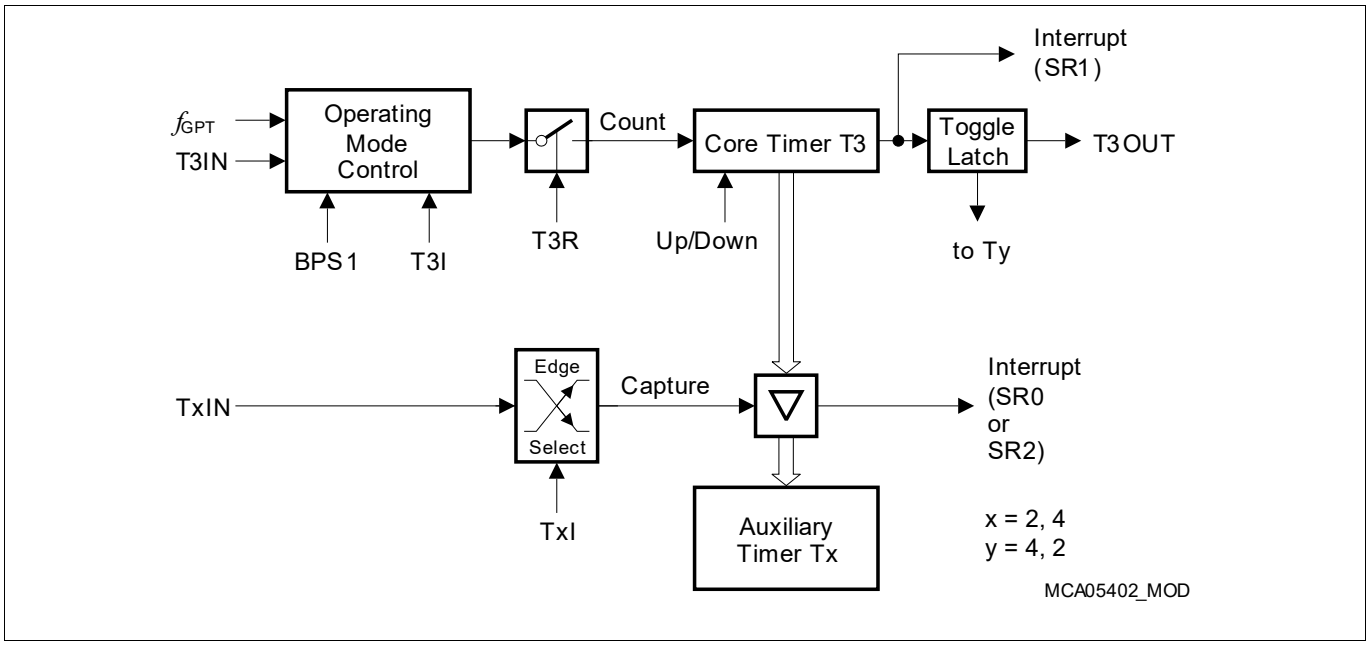


Figure 216 GPT1 Auxiliary Timer in Capture Mode

Upon a trigger (selected transition) at the corresponding input pin TxIN the contents of the core timer T3 are loaded into the auxiliary timer register Tx and the associated service request (SR0 or SR2) will be activated.

For Capture Mode operation, the respective timer input pin TxIN must be configured as input. To ensure that a transition of the capture input signal applied to TxIN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles, detailed in **Section 30.2.5**.

30.2.5 GPT1 Clock Signal Control

All actions within the timer block GPT1 are triggered by transitions of its basic clock. This basic clock is derived from the module clock f_{GPT} by a basic block prescaler, controlled by bitfield BPS1 in register T3CON (see **Figure 200**). The count clock can be generated in two different ways:

- **Internal count clock**, derived from GPT1's basic clock via a programmable prescaler, is used for (Gated) Timer Mode.
- **External count clock**, derived from the timer's input pin(s), is used for Counter Mode or Incremental Interface Mode.

For both ways, the basic clock determines the maximum count frequency and the timer's resolution:

Table 214 Basic Clock Selection for Block GPT1

Block Prescaler[1]	BPS1 = 01_B	BPS1 = 00_B[2]	BPS1 = 11_B	BPS1 = 10_B
Prescaling Factor for GPT1: F(BPS1)	F(BPS1) = 4	F(BPS1) = 8	F(BPS1) = 16	F(BPS1) = 32
Maximum External Count Frequency	$f_{GPT}/8$	$f_{GPT}/16$	$f_{GPT}/32$	$f_{GPT}/64$
Input Signal Stable Time	4 x t_{GPT}	8 x t_{GPT}	16 x t_{GPT}	32 x t_{GPT}

1) Please note the non-linear encoding of bitfield BPS1.
2) Default after reset.

Notes

1. *The GPT1 block uses a finite state machine to control the actions. Since multiple interactions are possible between the timers (T2, T3, T4), these elements are processed sequentially. However, all actions are normally completed within one basic clock cycle. The GPT1 state machine has 8 states (4 states when BPS1 = 01_B) and processes the timers in the order T3 - T2 (all actions except capture) - T4 - T2 (capture).*
2. *When initializing the GPT1 block after reset, and the block prescaler BPS1 in register T3CON needs to be set to a value different from its default value (00_B), it must be initialized first before any mode involving external trigger signals is configured. These modes include counter, incremental interface, capture, and reload mode. Otherwise, unintended count/capture/reload events may occur during the first basic clock cycle.*
   In this case, or when changing BPS1 during operation of the GPT1 block, disable related interrupts before modification of BPS1, and afterwards clear the corresponding service request flags and re-initialize those registers (T2, T3, T4) that might be affected by a count/capture/reload event.

Internal Count Clock Generation

In Timer Mode and Gated Timer Mode, the count clock for each GPT1 timer is derived from the GPT1 basic clock by a programmable prescaler, controlled by bitfield TxI in the respective timer's control register TxCON.

The count frequency f_{Tx} for a timer Tx and its resolution r_{Tx} are scaled linearly with lower clock frequencies, as can be seen from the following formula:

$$f_{Tx} = \frac{f_{GPT}}{F(BPS1) \times 2^{<TxI>}} \qquad r_{Tx}[\mu s] = \frac{F(BPS1) \times 2^{<TxI>}}{f_{GPT}[MHz]} \tag{30.1}$$

The effective count frequency depends on the common module clock prescaler factor F(BPS1) as well as on the individual input prescaler factor $2^{<TxI>}$. **Table 218** summarizes the resulting overall divider factors for a GPT1 timer that result from these cascaded prescalers.

Table 215 lists GPT1 timer's parameters (such as count frequency, resolution, and period) resulting from the selected overall prescaler factor F(BPS1) x $2^{<TxI>}$ and the module clock f_{GPT}. Note that some numbers may be rounded.

Table 215 GPT1 Timer Parameters

Overall Prescaler Factor	Example 1: Module Clock f_{GPT} = 100 MHz			Example 2: Module Clock f_{GPT} = 66.5 MHz		
	Frequency	Resolution	Period	Frequency	Resolution	Period
4	25 MHz	40 ns	2.621 ms	16.625 MHz	60.2 ns	3.94 ms
8	12.5 MHz	80 ns	5.243 ms	8.313 MHz	120.3 ns	7.88 ms
16	6.25 MHz	160 ns	10.49 ms	4.156 MHz	240.6 ns	15.77 ms
32	3.125 MHz	320 ns	20.97 ms	2.078 MHz	481.2 ns	31.54 ms
64	1.563 MHz	640 ns	41.94 ms	1.039 MHz	962.4 ns	63.07 ms
128	781.25 kHz	1.28 µs	83.89 ms	519.53 kHz	1.924 µs	126.14 ms
256	390.6 kHz	2.56 µs	167.8 ms	259.77 kHz	3.850 µs	252.29 ms
512	195.3 kHz	5.12 µs	335.5 ms	129.88 kHz	7.699 µs	504.58 ms
1024	97.7 kHz	10.24 µs	671.1 ms	64.94 kHz	15.398 µs	1.009 s
2048	48.8 kHz	20.48 µs	1.342 s	32.47 kHz	30.797 µs	2.018 s
4096	24.4 kHz	40.96 µs	2.684 s	16.24 kHz	61.594 µs	4.037 s

External Count Clock Input

The external input signals of the GPT1 block are sampled with the GPT1 basic clock (see **Figure 200**). To ensure that a signal is recognized correctly, its current level (high or low) must be held active for at least one complete sampling period, before changing. A signal transition is recognized if two subsequent samples of the input signal represent different levels. Therefore, a minimum of two basic clock periods is required for the sampling of an external input signal. Thus, the maximum frequency of an input signal must not be higher than half the basic clock.

Table 216 summarizes the resulting requirements for external GPT1 input signals.

Table 216 GPT1 External Input Signal Limits

GPT1 Divider BPS1	Input Freq. Factor	Input Phase Duration	Example 1: Module Clock f_{GPT} = 100 MHz		Example 2: Module Clock f_{GPT} = 66.5 MHz	
			Max. Input Frequency	Min. Level Hold Time	Max. Input Frequency	Min. Level Hold Time
01_B	$f_{GPT}/8$	4 x t_{GPT}	12.5 MHz	40 ns	8.313 MHz	60.2 ns
00_B	$f_{GPT}/16$	8 x t_{GPT}	6.25 MHz	80 ns	4.156 MHz	120.3 ns
11_B	$f_{GPT}/32$	16 x t_{GPT}	3.125 MHz	160 ns	2.078 MHz	240.6 ns
10_B	$f_{GPT}/64$	32 x t_{GPT}	1.563 MHz	320 ns	1.039 MHz	481.2 ns

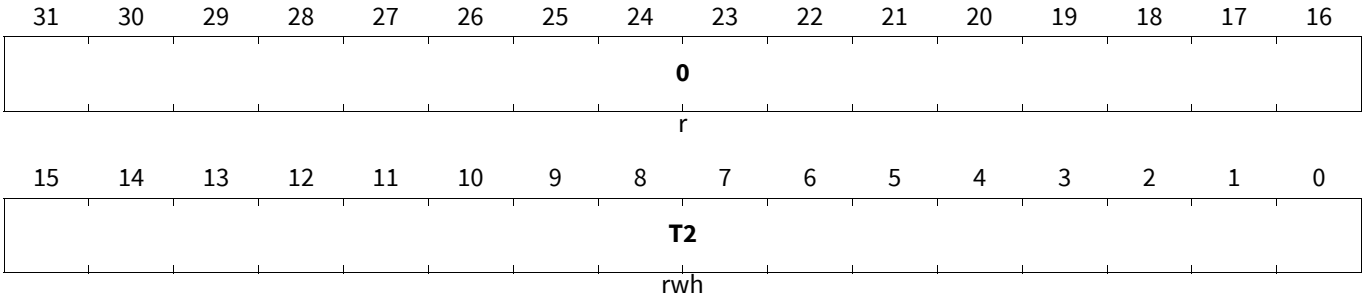
These limitations are valid for all external input signals to GPT1, including the external count signals in Counter Mode and Incremental Interface Mode, the gate input signals in Gated Timer Mode, and the external direction signals.

30.2.6 GPT1 Registers

Block GPT1 includes timer registers T2, T3, T4, and control registers T2CON, T3CON, T4CON.

Timer T2 Register

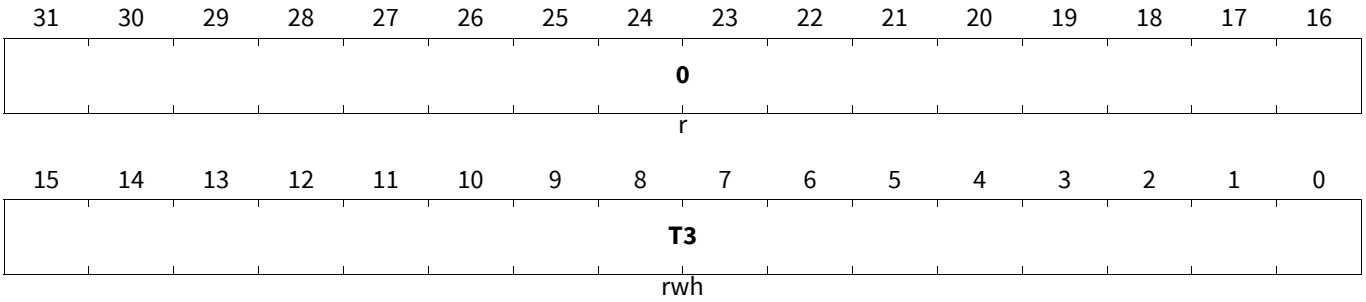
T2
Timer T2 Register **(0034_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T2	15:0	rwh	**Timer T2** Contains the current value of Timer T2.
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Timer T3 Register

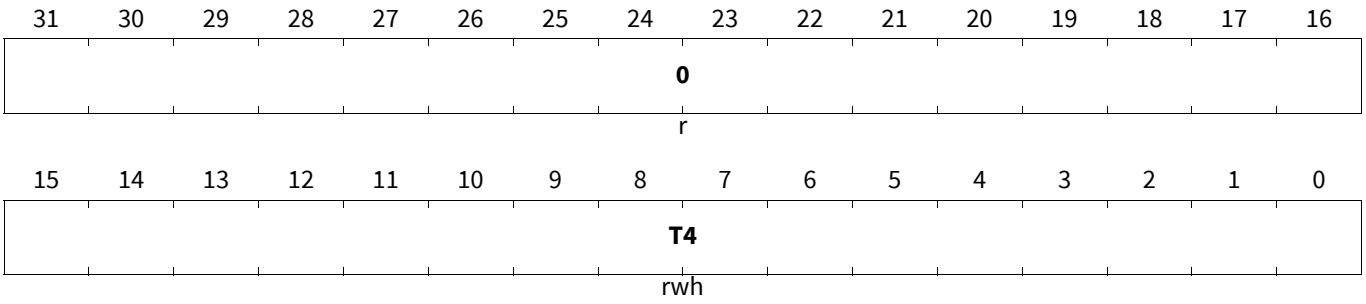
T3
Timer T3 Register **(0038_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T3	15:0	rwh	**Timer T3** Contains the current value of Timer T3.
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Timer T4 Register

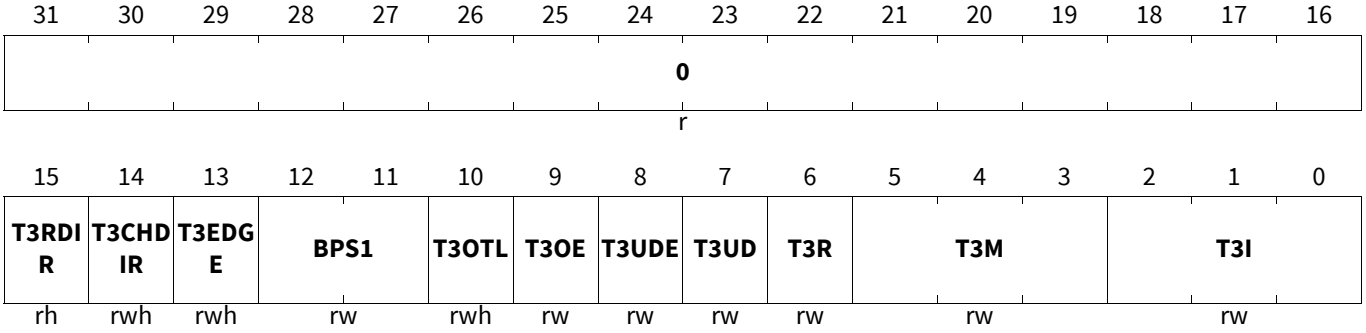
T4
Timer T4 Register **($003C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T4	15:0	rwh	**Timer T4** Contains the current value of Timer T4.
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Timer T3 Control Register

T3CON
Timer T3 Control Register **(0014_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T3I	2:0	rw	**Timer T3 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 218** for Timer Mode and Gated Timer Mode **Table 219** for Counter Mode **Table 221** for Incremental Interface Mode

Field	Bits	Type	Description
T3M	5:3	rw	**Timer T3 Mode Control** 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reserved. Do not use this combination 101_B Reserved. Do not use this combination 110_B Incremental Interface Mode (Rotation Detection Mode) 111_B Incremental Interface Mode (Edge Detection Mode)
T3R	6	rw	**Timer T3 Run Bit** 0_B Timer T3 stops 1_B Timer T3 runs
T3UD	7	rw	**Timer T3 Up/Down Control** *Note: This bit only directly controls count direction of T3 if bit T3UDE = 0.* 0_B Timer T3 counts up 1_B Timer T3 counts down
T3UDE	8	rw	**Timer T3 External Up/Down Enable** 0_B Count direction is controlled by bit T3UD; input T3EUD is disconnected 1_B Count direction is controlled by input T3EUD (see also **Table 217**)
T3OE	9	rw	**Overflow/Underflow Output Enable** 0_B Alternate Output Function Disabled 1_B State of T3 toggle latch is output on pin T3OUT
T3OTL	10	rwh	**Timer T3 Overflow Toggle Latch** Toggles on each overflow/underflow of T3. Can be set or cleared by software (see separate description)
BPS1	12:11	rw	**GPT1 Block Prescaler Control** Selects the basic clock for block GPT1 (see also **Section 30.2.5**) 00_B $f_{GPT}/8$ 01_B $f_{GPT}/4$ 10_B $f_{GPT}/32$ 11_B $f_{GPT}/16$
T3EDGE	13	rwh	**Timer T3 Edge Detection Flag** The bit is set each time a count edge is detected. T3EDGE must be cleared by software. 0_B No count edge was detected 1_B A count edge was detected
T3CHDIR	14	rwh	**Timer T3 Count Direction Change Flag** This bit is set each time the count direction of timer T3 changes. T3CHDIR must be cleared by software. 0_B No change of count direction was detected 1_B A change of count direction was detected

Field	Bits	Type	Description
T3RDIR	15	rh	**Timer T3 Rotation Direction Flag** 0_B Timer T3 counts up 1_B Timer T3 counts down
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Timer T2 Control Register

T2CON

Timer T2 Control Register **(0010_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2RDIR	**T2CHDIR**	**T2EDGE**	**T2IRDIS**	**0**		**T2RC**	**T2UDE**	**T2UD**	**T2R**	**T2M**			**T2I**		
rh	rwh	rwh	rw	r		rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
T2I	2:0	rw	**Timer T2 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 218** for Timer Mode and Gated Timer Mode **Table 220** for Counter Mode **Table 221** for Incremental Interface Mode
T2M	5:3	rw	**Timer T2 Mode Control (Basic Operating Mode)** *Note:* 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reload Mode 101_B Capture Mode 110_B Incremental Interface Mode (Rotation Detection Mode) 111_B Incremental Interface Mode (Edge Detection Mode)
T2R	6	rw	**Timer T2 Run Bit** *Note: This bit only controls timer T2 if bit T2RC = 0.* 0_B Timer T2 stops 1_B Timer T2 runs

Field	Bits	Type	Description
T2UD	7	rw	**Timer T2 Up/Down Control** *Note: This bit only directly controls count direction of T2 if bit T2UDE = 0.* 0_B Timer T2 counts up 1_B Timer T2 counts down
T2UDE	8	rw	**Timer T2 External Up/Down Enable** 0_B Count direction is controlled by bit T2UD; input T2EUD is disconnected 1_B Count direction is controlled by input T2EUD (see also **Table 217**)
T2RC	9	rw	**Timer T2 Remote Control** 0_B Timer T2 is controlled by its own run bit T2R 1_B Timer T2 is controlled by the run bit T3R of core timer T3, not by bit T2R
T2IRDIS	12	rw	**Timer T2 Interrupt Disable** 0_B Interrupt generation for T2CHDIR and T2EDGE interrupts in Incremental Interface Mode is enabled 1_B Interrupt generation for T2CHDIR and T2EDGE interrupts in Incremental Interface Mode is disabled
T2EDGE	13	rwh	**Timer T2 Edge Detection** The bit is set each time a count edge is detected. T2EDGE must be cleared by software. 0_B No count edge was detected 1_B A count edge was detected
T2CHDIR	14	rwh	**Timer T2 Count Direction Change** The bit is set each time the count direction of timer T2 changes. T2CHDIR must be cleared by software. 0_B No change in count direction was detected 1_B A change in count direction was detected
T2RDIR	15	rh	**Timer T2 Rotation Direction** 0_B Timer T2 counts up 1_B Timer T2 counts down
0	11:10, 31:16	r	**Reserved** Read as 0; should be written with 0.

Timer T4 Control Register

T4CON
Timer T4 Control Register **(0018_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T4RDIR	T4CHDIR	T4EDGE	T4IRDIS	CLRT3EN	CLRT2EN	T4RC	T4UDE	T4UD	T4R	T4M			T4I		
rh	rwh	rwh	rw	rw	rw	rw	rw	rw	rw	rw			rw		

Field	Bits	Type	Description
T4I	2:0	rw	**Timer T4 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 218** for Timer Mode and Gated Timer Mode **Table 220** for Counter Mode **Table 221** for Incremental Interface Mode
T4M	5:3	rw	**Timer T4 Mode Control (Basic Operating Mode)** 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reload Mode 101_B Capture Mode 110_B Incremental Interface Mode (Rotation Detection Mode) 111_B Incremental Interface Mode (Edge Detection Mode)
T4R	6	rw	**Timer T4 Run Bit** *Note: This bit only controls timer T4 if bit T4RC = 0.* 0_B Timer T4 stops 1_B Timer T4 runs
T4UD	7	rw	**Timer T4 Up/Down Control** *Note: This bit only directly controls count direction of T4 if bit T4UDE = 0.* 0_B Timer T4 counts up 1_B Timer T4 counts down
T4UDE	8	rw	**Timer T4 External Up/Down Enable** 0_B Count direction is controlled by bit T4UD; input T4EUD is disconnected 1_B Count direction is controlled by input T4EUD (see also **Table 217**)

Field	Bits	Type	Description
T4RC	9	rw	**Timer T4 Remote Control** 0_B Timer T4 is controlled by its own run bit T4R 1_B Timer T4 is controlled by the run bit T3R of core timer T3, but not by bit T4R
CLRT2EN	10	rw	**Clear Timer T2 Enable** Enables the automatic clearing of timer T2 upon a falling edge of the selected T4EUD input. 0_B No effect of T4EUD on timer T2 1_B A falling edge on T4EUD clears timer T2
CLRT3EN	11	rw	**Clear Timer T3 Enable** Enables the automatic clearing of timer T3 upon a falling edge of the selected T4IN input. 0_B No effect of T4IN on timer T3 1_B A falling edge on T4IN clears timer T3
T4IRDIS	12	rw	**Timer T4 Interrupt Disable** 0_B Interrupt generation for T4CHDIR and T4EDGE interrupts in Incremental Interface Mode is enabled 1_B Interrupt generation for T4CHDIR and T4EDGE interrupts in Incremental Interface Mode is disabled
T4EDGE	13	rwh	**Timer T4 Edge Detection** The bit is set each time a count edge is detected. T4EDGE has to be cleared by software. 0_B No count edge was detected 1_B A count edge was detected
T4CHDIR	14	rwh	**Timer T4 Count Direction Change** The bit is set each time the count direction of timer T4 changes. T4CHDIR must be cleared by software. 0_B No change in count direction was detected 1_B A change in count direction was detected
T4RDIR	15	rh	**Timer T4 Rotation Direction** 0_B Timer T4 counts up 1_B Timer T4 counts down
0	31:16	r	**Reserved** Read as 0; should be written with 0.

30.2.7 Encoding of Specific Bitfields of GPT1 Registers

Table 217 GPT1 Timer Count Direction Control

Pin TxEUD	Bit TxUDE	Bit TxUD	Count Direction	Bit TxRDIR
X	0	0	Count Up	0
X	0	1	Count Down	1
0	1	0	Count Up	0
1	1	0	Count Down	1
0	1	1	Count Down	1
1	1	1	Count Up	0

Table 218 GPT1 Overall Prescaler Factors for Internal Count Clock (Timer Mode and Gated Timer Mode)

Individual Prescaler for Tx	Common Prescaler for Module Clock[1]			
	BPS1 = 01_B	BPS1 = 00_B	BPS1 = 11_B	BPS1 = 10_B
TxI = 000_B	4	8	16	32
TxI = 001_B	8	16	32	64
TxI = 010_B	16	32	64	128
TxI = 011_B	32	64	128	256
TxI = 100_B	64	128	256	512
TxI = 101_B	128	256	512	1024
TxI = 110_B	256	512	1024	2048
TxI = 111_B	512	1024	2048	4096

1) Please note the non-linear encoding of bitfield BPS1.

Table 219 GPT1 Core Timer T3 Input Edge Selection (Counter Mode)

TxI	Triggering Edge for Counter Increment/Decrement
000_B	None. Counter T3 is disabled
001_B	Positive transition (rising edge) on T3IN
010_B	Negative transition (falling edge) on T3IN
011_B	Any transition (rising or falling edge) on T3IN
$1XX_B$	Reserved. Do not use this combination

Table 220 GPT1 Auxiliary Timers T2/T4 Input Edge Selection

T2I/T4I	Triggering Edge for Counter Increment/Decrement, Capture[1] or Reload
$X00_B$	None. Counter Tx is disabled
001_B	Positive transition (rising edge) on TxIN
010_B	Negative transition (falling edge) on TxIN
011_B	Any transition (rising or falling edge) on TxIN
101_B	Positive transition (rising edge) of T3 toggle latch T3OTL[2]

Table 220 GPT1 Auxiliary Timers T2/T4 Input Edge Selection (cont'd)

T2I/T4I	Triggering Edge for Counter Increment/Decrement, Capture[1] or Reload
110_B	Negative transition (falling edge) of T3 toggle latch T3OTL[2]
111_B	Any transition (rising or falling edge) of T3 toggle latch T3OTL[2]

1) For Capture Mode, only settings TxI = 0XX may be used.

2) Not for Capture Mode.

Table 221 GPT1 Timer Tx Input Edge Selection (Incremental Interface Mode)

TxI	Triggering Edge for Counter Increment/Decrement
000_B	None. Counter Tx stops.
001_B	Any transition (rising or falling edge) on TxIN.
010_B	Any transition (rising or falling edge) on TxEUD.
011_B	Any transition (rising or falling edge) on any Tx input (TxIN or TxEUD).
$1XX_B$	Reserved. Do not use this combination.

30.3 Timer Block GPT2

Both timers of block GPT2 (T5, T6) can run in one of 3 basic modes: Timer Mode, Gated Timer Mode, or Counter Mode. All timers can count up or down. Each timer of GPT2 is controlled by a separate control register TxCON.

Each timer has an input pin TxIN (alternate pin function) associated with it, which serves as the gate control in Gated Timer Mode, or as the count input in Counter Mode. The count direction (up/down) may be programmed via software or may be dynamically altered by a signal at the External Up/Down control input TxEUD (alternate pin function). An overflow/underflow of core timer T6 is indicated by the Output Toggle Latch T6OTL, whose state may be output on the associated pin T6OUT (alternate pin function). The auxiliary timer T5 may additionally be concatenated with core timer T6 (through T6OTL).

The Capture/Reload register CAPREL can be used to capture the contents of timer T5, or to reload timer T6. A special mode facilitates the use of register CAPREL for both functions at the same time. This mode allows frequency multiplication. The capture function is triggered by the input pin CAPIN, or by GPT1 timer's T3 input lines T3IN and T3EUD. The reload function is triggered by an overflow or underflow of timer T6.

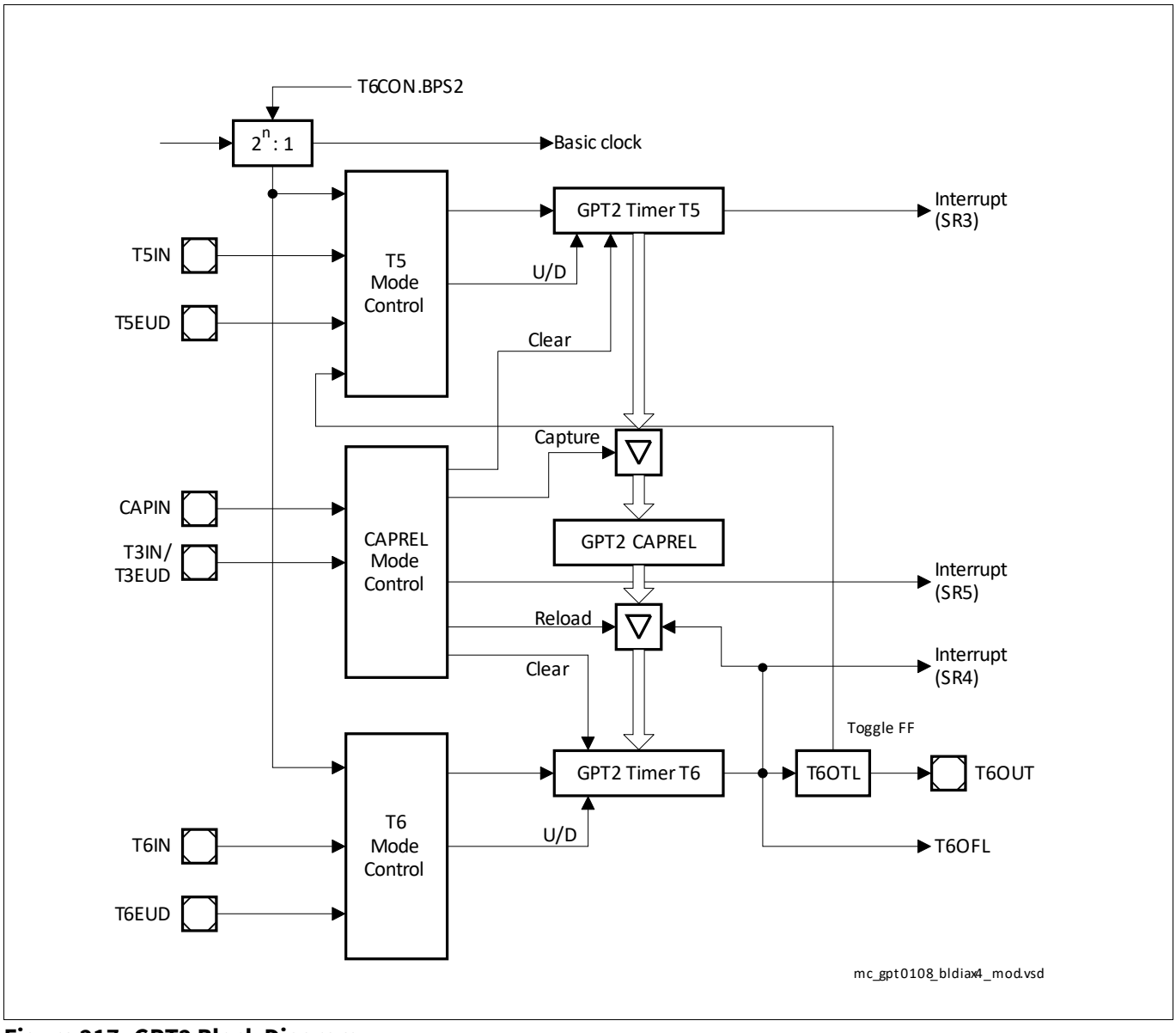


Figure 217 GPT2 Block Diagram

The current contents of each timer can be read or modified by the CPU by accessing the corresponding timer count registers T5 or T6. When any of the timer registers is written by the CPU in the state immediately preceding a timer increment, decrement, reload, or capture operation, the CPU write operation has priority in order to guarantee correct results.

The interrupt requests of GPT2 are signalled on service request lines SR3, SR4, and SR5.

The input and output lines of GPT2 are connected to pins. The control registers for the port functions are located in the respective port modules.

*Note: The timing requirements for external input signals can be found in **Section 30.3.6**, **Section 30.6** summarizes the module interface signals, including pins and interrupt request signals.*

30.3.1 GPT2 Core Timer T6 Control

The current contents of the core timer T6 are reflected by its count register T6. This register can also be written to by the CPU, for example, to set the initial start value.

The core timer T6 is configured and controlled via its control register T6CON.

Timer T6 Run Control

The core timer T6 can be started or stopped by software through bit T6R (timer T6 run bit). This bit is relevant in all operating modes of T6. Setting bit T6R will start the timer, clearing bit T6R stops the timer.

In Gated Timer Mode, the timer will only run if T6R = 1 and the gate is active (high or low, as programmed).

Note: When bit T5RC in timer control register T5CON is set, bit T6R will also control (start and stop) the Auxiliary Timer T5.

Count Direction Control

The count direction of the GPT2 timers (core timer and auxiliary timer) can be controlled either by software or by the external input pin TxEUD (Timer Tx External Up/Down Control Input). These options are selected by bits TxUD and TxUDE in the respective control register TxCON. When the up/down control is provided by software (bit TxUDE = 0), the count direction can be altered by setting or clearing bit TxUD. When bit TxUDE = 1, pin TxEUD is selected to be the controlling source of the count direction. However, bit TxUD can still be used to reverse the actual count direction, as shown in **Table 226**. The count direction can be changed regardless of whether or not the timer is running.

Timer T6 Output Toggle Latch

The overflow/underflow signal of timer T6 is connected to a block named ‘Toggle Latch’, shown in the Timer Mode diagrams. **Figure 218** illustrates the details of this block. An overflow or underflow of T6 will clock two latches: The first latch represents bit T6OTL in control register T6CON. The second latch is an internal latch toggled by T6OTL’s output. Both latch outputs are connected to the input control block of the auxiliary timer T5. The output level of the shadow latch will match the output level of T6OTL, but is delayed by one clock cycle. When the T6OTL value changes, this will result in a temporarily different output level from T6OTL and the shadow latch, which can trigger the selected count event in T5.

When software writes to T6OTL, both latches are set or cleared simultaneously. In this case, both signals to the auxiliary timer carry the same level and no edge will be detected. Bit T6OE (overflow/underflow output enable) in register T6CON enables the state of T6OTL to be monitored via an external pin T6OUT. When T6OTL is linked to an external port pin (must be configured as output), T6OUT can be used to control external HW. If T6OE = 1, pin T6OUT outputs the state of T6OTL. If T6OE = 0, pin T6OUT outputs a high level (while it selects the timer output signal).

As can be seen from **Figure 218**, when latch T6OTL is modified by software to determine the state of the output line, also the internal shadow latch is set or cleared accordingly. Therefore, no trigger condition is detected by T5 in this case.

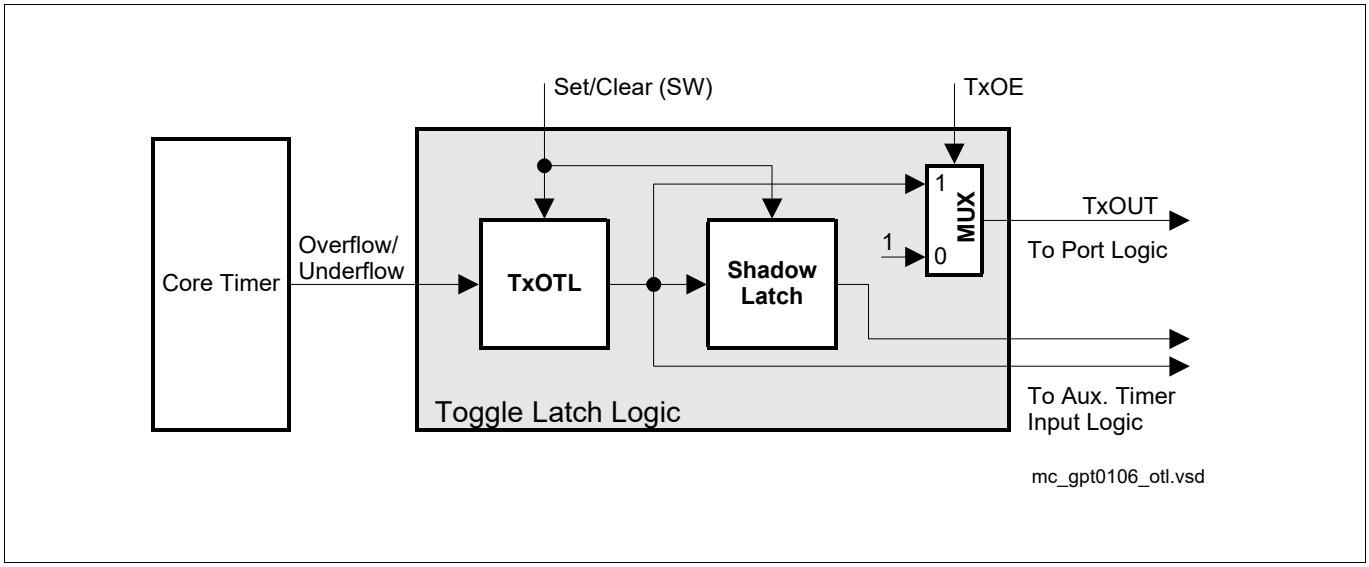


Figure 218 Block Diagram of the Toggle Latch Logic of Core Timer T6 (x = 6)

30.3.2 GPT2 Core Timer T6 Operating Modes

Timer T6 can operate in one of several modes.

Timer T6 in Timer Mode

Timer Mode for the core timer T6 is selected by setting bitfield T6M in register T6CON to 000_B. In this mode, T6 is clocked with the module's input clock f_{GPT} divided by two programmable prescalers controlled by bitfields BPS2 and T6I in register T6CON. Please see **Section 30.3.6** for details on the input clock options.

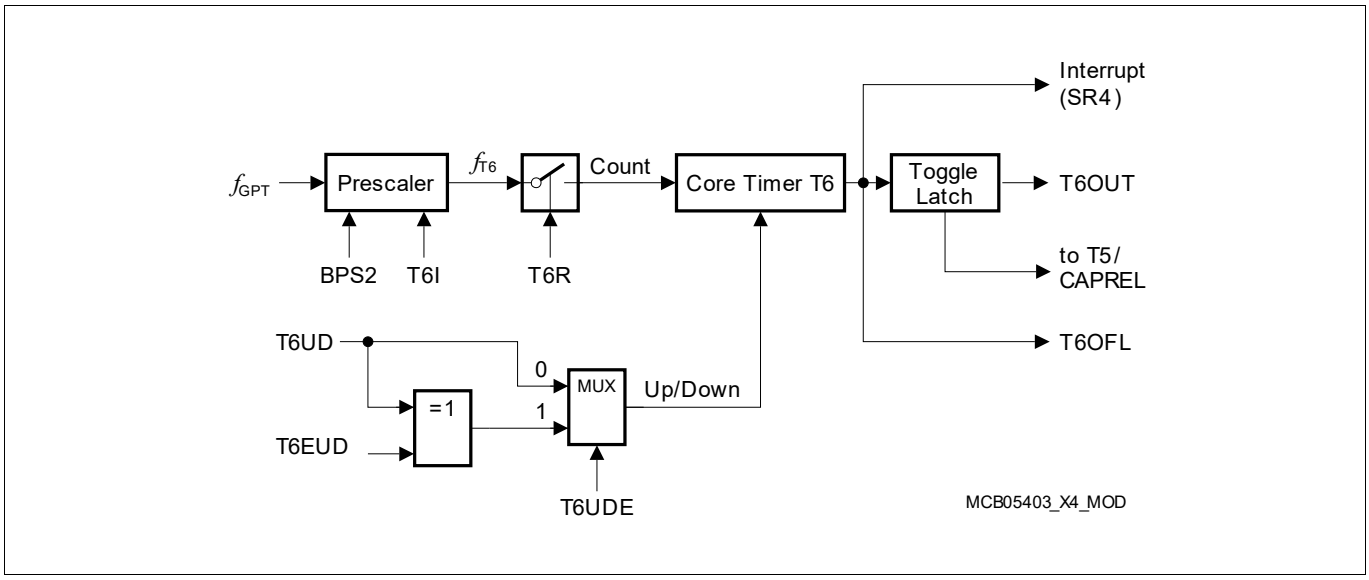


Figure 219 Block Diagram of Core Timer T6 in Timer Mode

Timer 6 in Gated Timer Mode

Gated Timer Mode for the core timer T6 is selected by setting bitfield T6M in register T6CON to 010_B or 011_B. Bit T6M.0 (T6CON.3) selects the active level of the gate input. The same options for the input frequency are available in Gated Timer Mode as in Timer Mode (see **Section 30.3.6**). However, the input clock to the timer in this mode is gated by the external input pin T6IN (Timer T6 External Input).
To enable this operation, the associated pin T6IN must be configured as input.

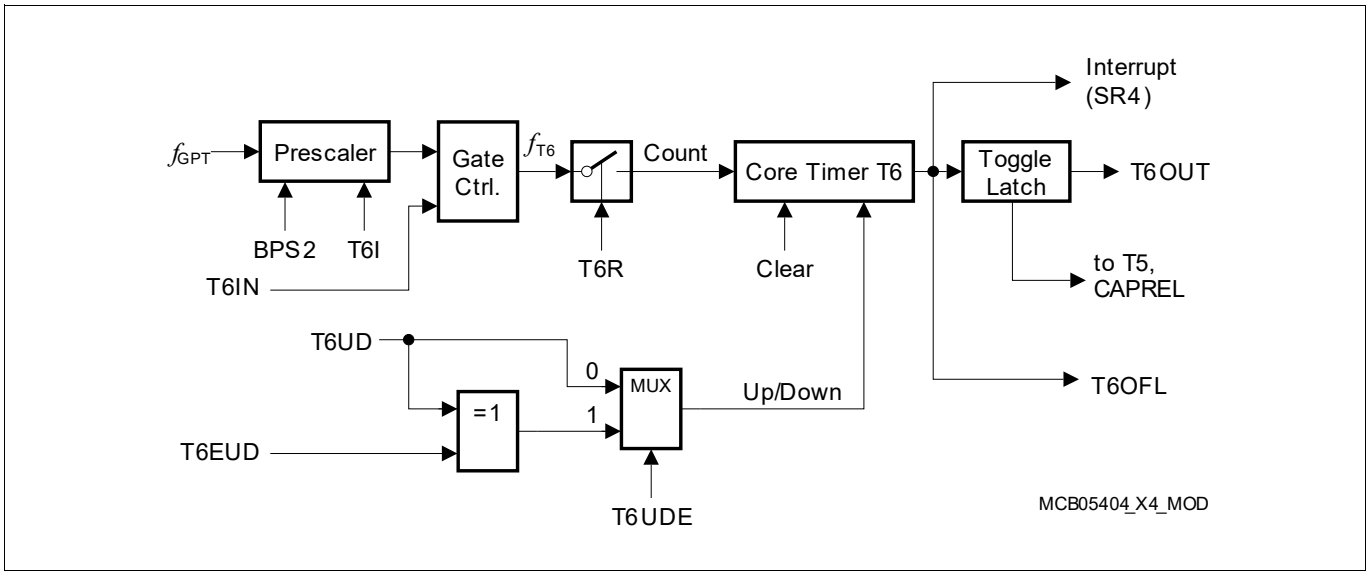


Figure 220 Block Diagram of Core Timer T6 in Gated Timer Mode

If T6M = 010_B, the timer is enabled when T6IN shows a low level. A high level at this line stops the timer. If T6M = 011_B, line T6IN must have a high level in order to enable the timer. Additionally, the timer can be turned on or off by software using bit T6R. The timer will only run if T6R is 1 and the gate is active. It will stop if either T6R is 0 or the gate is inactive.

Note: A transition of the gate signal at pin T6IN does not cause a service request.

Timer 6 in Counter Mode

Counter Mode for the core timer T6 is selected by setting bitfield T6M in register T6CON to 001_B. In Counter Mode, timer T6 is clocked by a transition at the external input pin T6IN. The event causing an increment or decrement of the timer can be a positive, a negative, or both a positive and a negative transition at this line. Bitfield T6I in control register T6CON selects the triggering transition (see **Table 228**).

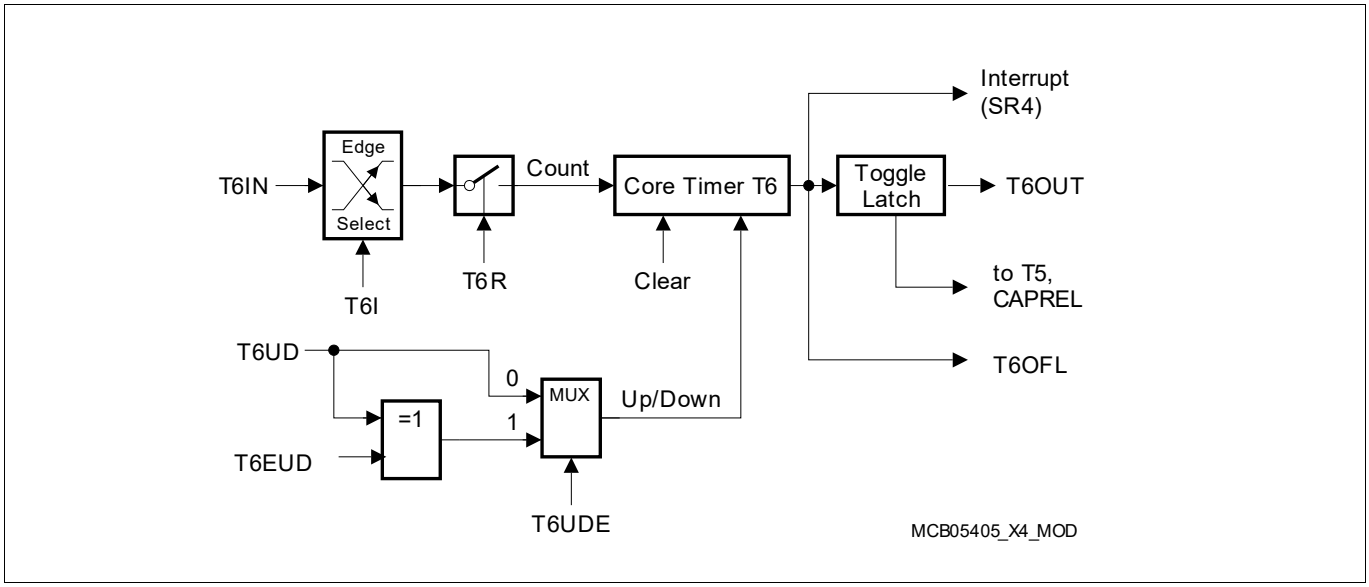


Figure 221 Block Diagram of Core Timer T6 in Counter Mode

For Counter Mode operation, pin T6IN must be configured as input. The maximum input frequency allowed in Counter Mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to T6IN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 30.3.6**.

30.3.3 GPT2 Auxiliary Timer T5 Control

Auxiliary timer T5 can be configured for Timer Mode, Gated Timer Mode, or Counter Mode with the same options for the timer frequencies and the count signal as the core timer T6. In addition to these 3 counting modes, the auxiliary timer can be concatenated with the core timer. The contents of T5 may be captured to register CAPREL upon an external or an internal trigger. The start/stop function of the auxiliary timer can be remotely controlled by the T6 run control bit. Both timers may thus be controlled synchronously.

The current contents of the auxiliary timer are reflected by its count register T5. This register can also be written to by the CPU, for example, to set the initial start value.

The individual configurations for timer T5 are determined by its control register T5CON. Some bits in this register also control the function of the CAPREL register. Note that functions which are present in all timers of block GPT2 are controlled in the same bit positions and in the same manner in each of the specific control registers.

Note: The auxiliary timer has no output toggle latch and no alternate output function.

Timer T5 Run Control

The auxiliary timer T5 can be started or stopped by software in two different ways:

- Through the associated timer run bit (T5R). In this case it is required that the respective control bit T5RC = 0.
- Through the core timer's run bit (T6R). In this case the respective remote control bit must be set (T5RC = 1).

The selected run bit is relevant in all operating modes of T5. Setting the bit will start the timer, clearing the bit stops the timer.

In Gated Timer Mode, the timer will only run if the selected run bit is set and the gate is active (high or low, as programmed).

Note: If remote control is selected T6R will start/stop timer T6 and the auxiliary timer T5 synchronously.

30.3.4 GPT2 Auxiliary Timer T5 Operating Modes

The operation of the auxiliary timer in the basic operating modes is almost identical with the core timer's operation, with very few exceptions. Additionally, some combined operating modes can be selected.

Timer T5 in Timer Mode

Timer Mode for the auxiliary timer T5 is selected by setting its bitfield T5M in register T5CON to 000_B.

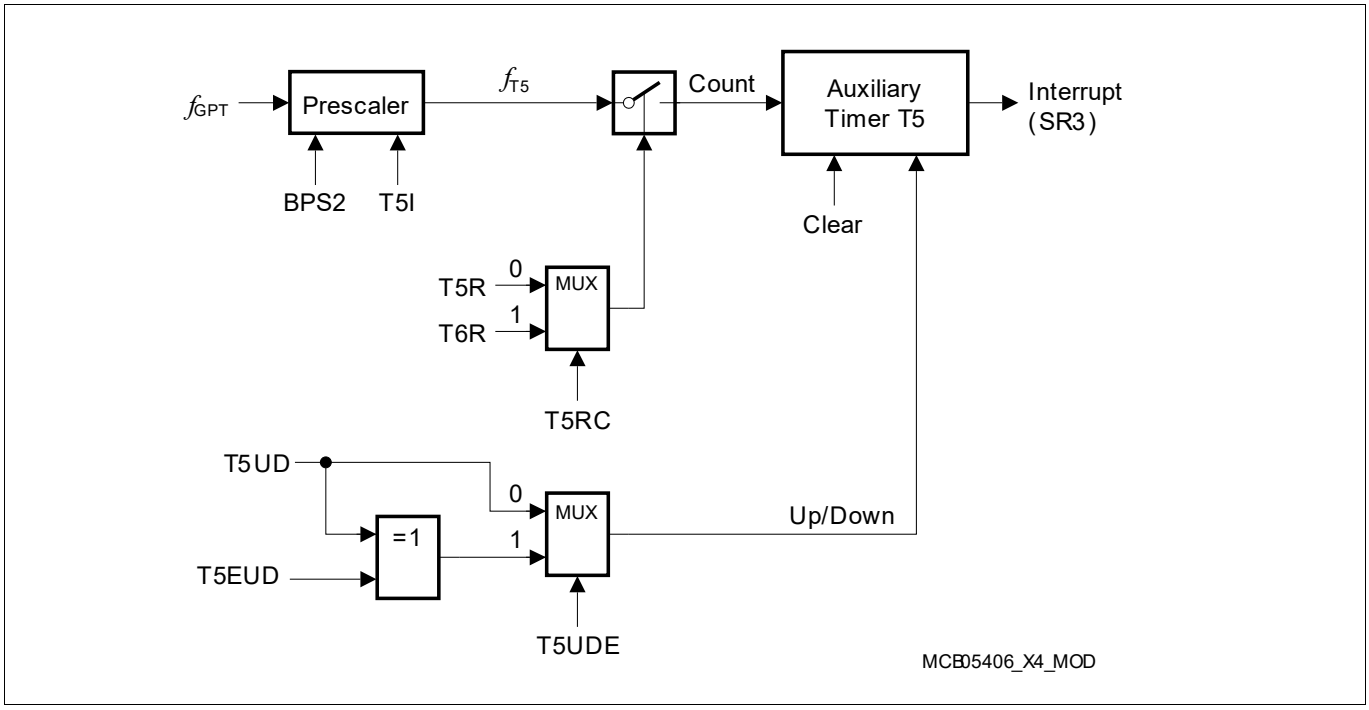


Figure 222 Block Diagram of Auxiliary Timer T5 in Timer Mode

Timer T5 in Gated Timer Mode

Gated Timer Mode for the auxiliary timer T5 is selected by setting bitfield T5M in register T5CON to 010_B or 011_B. Bit T5M.0 (T5CON.3) selects the active level of the gate input.

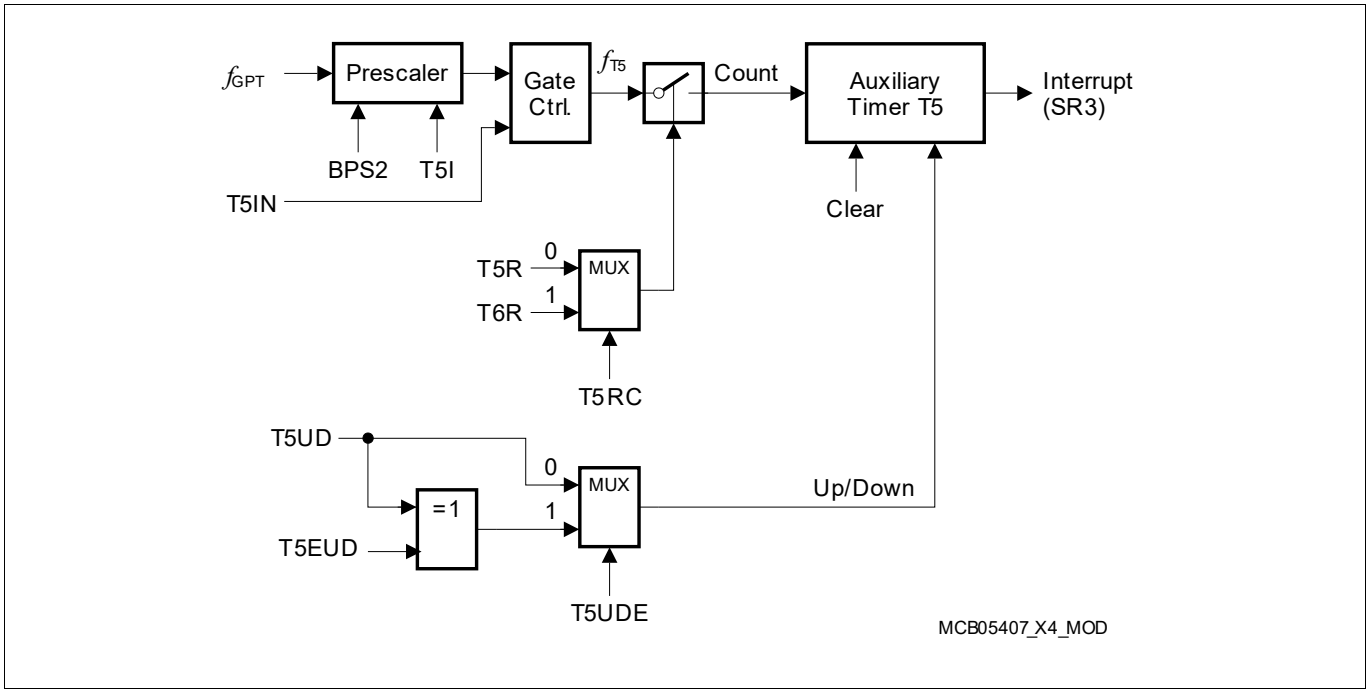


Figure 223 Block Diagram of Auxiliary Timer T5 in Gated Timer Mode

Note: A transition of the gate signal at line T5IN does not cause a service request.
There is no output toggle latch for T5.
Start/stop of the auxiliary timer can be controlled locally or remotely.

Timer T5 in Counter Mode

Counter Mode for auxiliary timer T5 is selected by setting bitfield T5M in register T5CON to 001_B. In Counter Mode, the auxiliary timer can be clocked either by a transition at its external input line T5IN, or by a transition of timer T6's toggle latch T6OTL. The event causing an increment or decrement of a timer can be a positive, a negative, or both a positive and a negative transition at either the respective input pin or at the toggle latch. Bitfield T5I in control register T5CON selects the triggering transition (see **Table 229**).

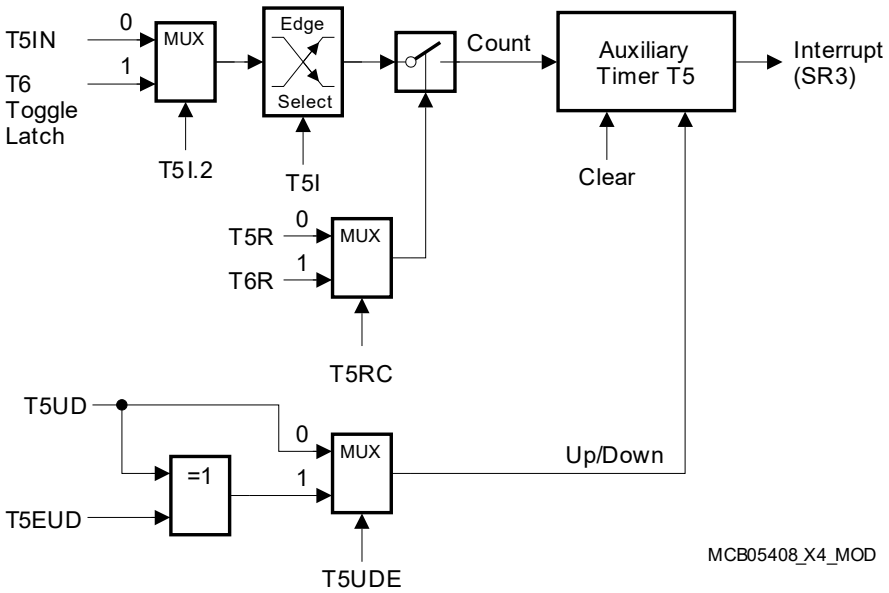


Figure 224 Block Diagram of Auxiliary Timer T5 in Counter Mode

Note: Only state transitions of T6OTL which are caused by the overflows/underflows of T6 will trigger the counter function of T5. Modifications of T6OTL via software will NOT trigger the counter function of T5.

For counter operation, pin T5IN must be configured as input. The maximum input frequency allowed in Counter Mode depends on the selected prescaler value. To ensure that a transition of the count input signal applied to T5IN is recognized correctly, its level must be held high or low for a minimum number of module clock cycles before it changes. This information can be found in **Section 30.3.6**.

Timer Concatenation

Using the toggle bit T6OTL as a clock source for the auxiliary timer in Counter Mode concatenates the core timer T6 with the auxiliary timer T5. This concatenation forms either a 32-bit or a 33-bit timer/counter, depending on which transition of T6OTL is selected to clock the auxiliary timer.

- **32-bit Timer/Counter:** If both a positive and a negative transition of T6OTL are used to clock the auxiliary timer, this timer is clocked on every overflow/underflow of the core timer T6. Thus, the two timers form a 32-bit timer.
- **33-bit Timer/Counter:** If either a positive or a negative transition of T6OTL is selected to clock the auxiliary timer, this timer is clocked on every second overflow/underflow of the core timer T6. This configuration forms a 33-bit timer (16-bit core timer + T6OTL + 16-bit auxiliary timer).
  As long as bit T6OTL is not modified by software, it represents the state of the internal toggle latch, and can be regarded as part of the 33-bit timer.

The count directions of the two concatenated timers are not required to be the same. This offers a wide variety of different configurations.

T6, which represents the low-order part of the concatenated timer, can operate in Timer Mode, Gated Timer Mode or Counter Mode in this case.

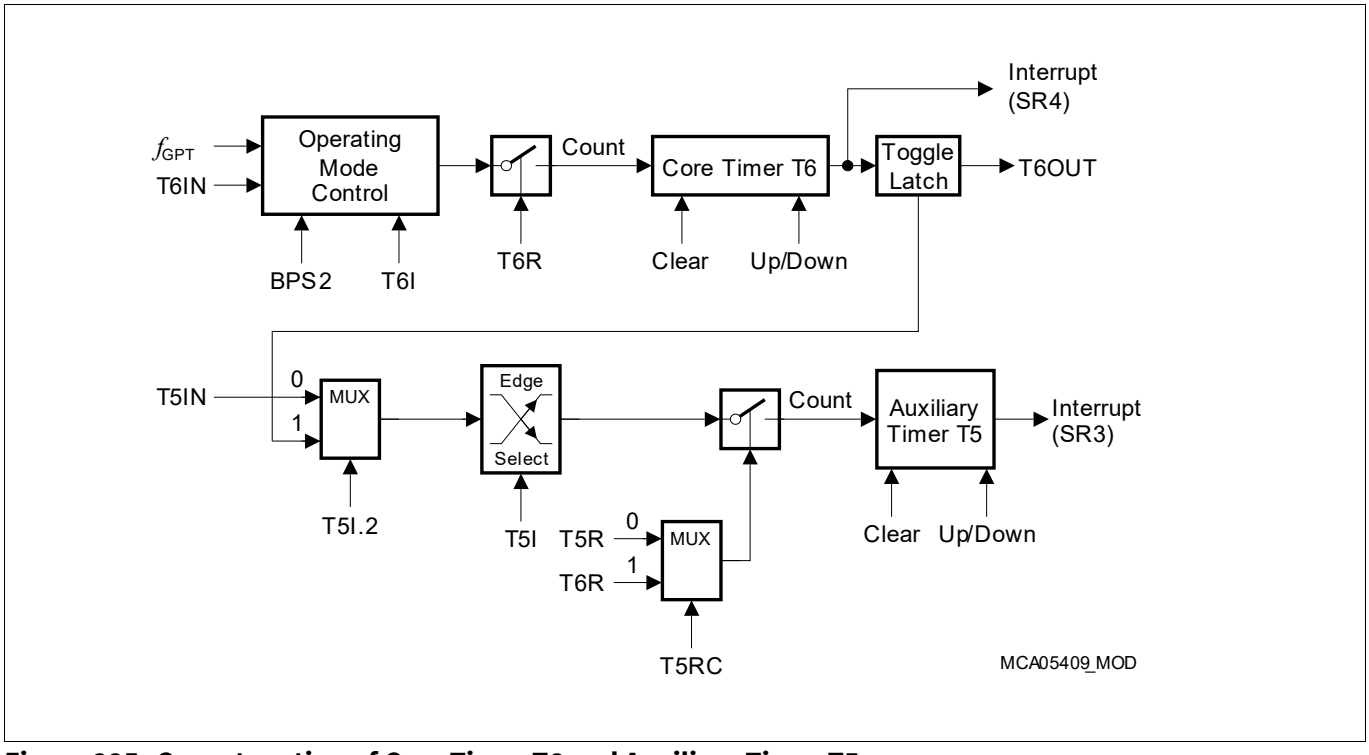


Figure 225 Concatenation of Core Timer T6 and Auxiliary Timer T5

When reading the low and high parts of the concatenated timer, care must be taken to obtain consistent values in particular after a timer overflow/underflow (e.g. one part may already have considered an overflow, while the other has not).

Note: This is a general issue when reading multi-word results with consecutive instructions, and not necessarily unique to the GPT12 module architecture.

The following algorithm may be used to read concatenated GPT2 timers, represented by TIMER_HIGH (for auxiliary timer T5) and TIMER_LOW (for core timer T6). The high part is read twice, and reading of the low part is repeated if two different values were read for the high part:

- TIMER_HIGH_TMP = T5
- TIMER_LOW = T6
- Wait two basic clock cycles (to allow increment/decrement of auxiliary timer in case of core timer overflow/underflow) - see **Table 222**
- TIMER_HIGH = T5
- If TIMER_HIGH is not equal to TIMER_HIGH_TMP then TIMER_LOW = T6

After execution of this algorithm, TIMER_HIGH and TIMER_LOW represent a consistent time stamp of the concatenated timers.

The equivalent number of module clock cycles corresponding to two basic clock cycles is shown in **Table 222**.

Table 222 Number of Module Clock Cycles to Wait for Two Basic Clock Cycles

Block Prescaler	BPS2 = 01_B	BPS2 = 00_B	BPS2 = 11_B	BPS2 = 10_B
Number of module clocks	4	8	16	32

In case the required timer resolution can be achieved with different combinations of the Block Prescaler BPS2 and the Individual Prescalers TxI (see **Table 227**), the variant with the smallest value for the Block Prescaler may be chosen to minimize the waiting time. E.g. in order to run T6 at $f_{SYS}/512$, select BPS2 = 00_B, T6I = 111_B, and insert 8 NOPs (or other instructions) to ensure the required waiting time before reading TIMER_HIGH the second time.

30.3.5 GPT2 Register CAPREL Operating Modes

The Capture/Reload register CAPREL can be used to capture the contents of timer T5, or to reload timer T6. A special mode facilitates the use of register CAPREL for both functions at the same time. This mode allows frequency multiplication. The capture function is triggered by the input pin CAPIN, by GPT1 timer's T3 input lines T3IN and T3EUD, or by read accesses to GPT1 timers. The reload function is triggered by an overflow or underflow of timer T6.

The capture trigger signal can also be used to clear the contents of timers T5 and T6 individually.

The functions of register CAPREL are controlled via several bit(field)s in the timer control registers T5CON and T6CON.

Capture/Reload Register CAPREL in Capture Mode

Capture Mode for register CAPREL is selected by setting bit T5SC in control register T5CON (set bitfield CI in register T5CON to a non-zero value to select a trigger signal). In Capture Mode, the contents of the auxiliary timer T5 are latched into register CAPREL in response to a signal transition at the selected external input pin(s). Bit CT3 selects the external input line CAPIN or the input lines T3IN and/or T3EUD of GPT1 timer T3 as the source for a capture trigger. Either a positive, a negative, or both a positive and a negative transition at line CAPIN can be selected to trigger the capture function, or transitions on input T3IN or input T3EUD or both inputs, T3IN and T3EUD. The active edge is controlled by bitfield CI in register T5CON. **Table 230** summarizes these options.

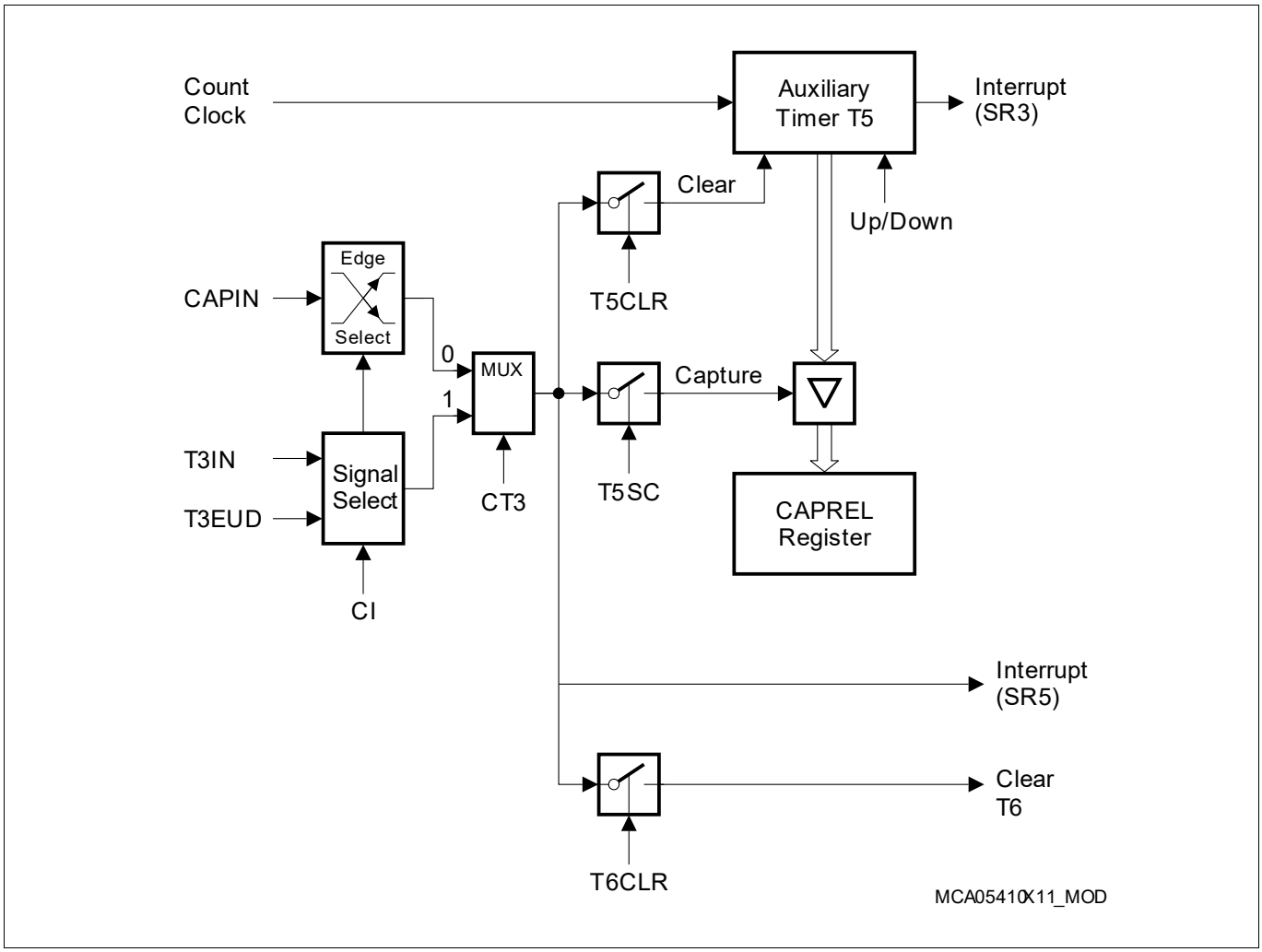


Figure 226 Capture/Reload Register CAPREL in Capture Mode

When a selected trigger is detected, the contents of the auxiliary timer T5 are latched into register CAPREL and the service request is activated. The same event can optionally clear timer T5 and/or timer T6. This option is enabled by bit T5CLR in register T5CON and bit T6CLR in register T6CON, respectively. If TxCLR = 0 the contents of timer Tx is not affected by a capture. If TxCLR = 1 timer Tx is cleared after the current timer T5 value has been latched into register CAPREL.

Note: Bit T5SC only controls whether or not a capture is performed. If T5SC is cleared the external input pin(s) can still be used to clear timer T5 and/or T6, or as external interrupt input(s). This interrupt is signalled by the CAPREL interrupt request SR5.

When capture triggers T3IN or T3EUD are enabled (CT3 = 1), register CAPREL captures the contents of T5 upon transitions of the selected input(s). These values can be used to measure T3's input signals. This is useful, for example, when T3 operates in Incremental Interface Mode, in order to derive dynamic information (speed, acceleration) from the input signals.

For Capture Mode operation, the selected pins CAPIN, T3IN, or T3EUD must be configured as input. To ensure that a transition of a trigger input signal applied to one of these inputs is recognized correctly, its level must be held high or low for a minimum number of module clock cycles, detailed in **Section 30.3.6**.

Capture/Reload Register CAPREL in Reload Mode

Reload Mode for register CAPREL is selected by setting bit T6SR in control register T6CON. In Reload Mode, the core timer T6 is reloaded with the contents of register CAPREL, triggered by an overflow or underflow of T6. This will not activate the service request SR5 associated with the CAPREL register. However, service request SR4 will be activated, indicating the overflow/underflow of T6.

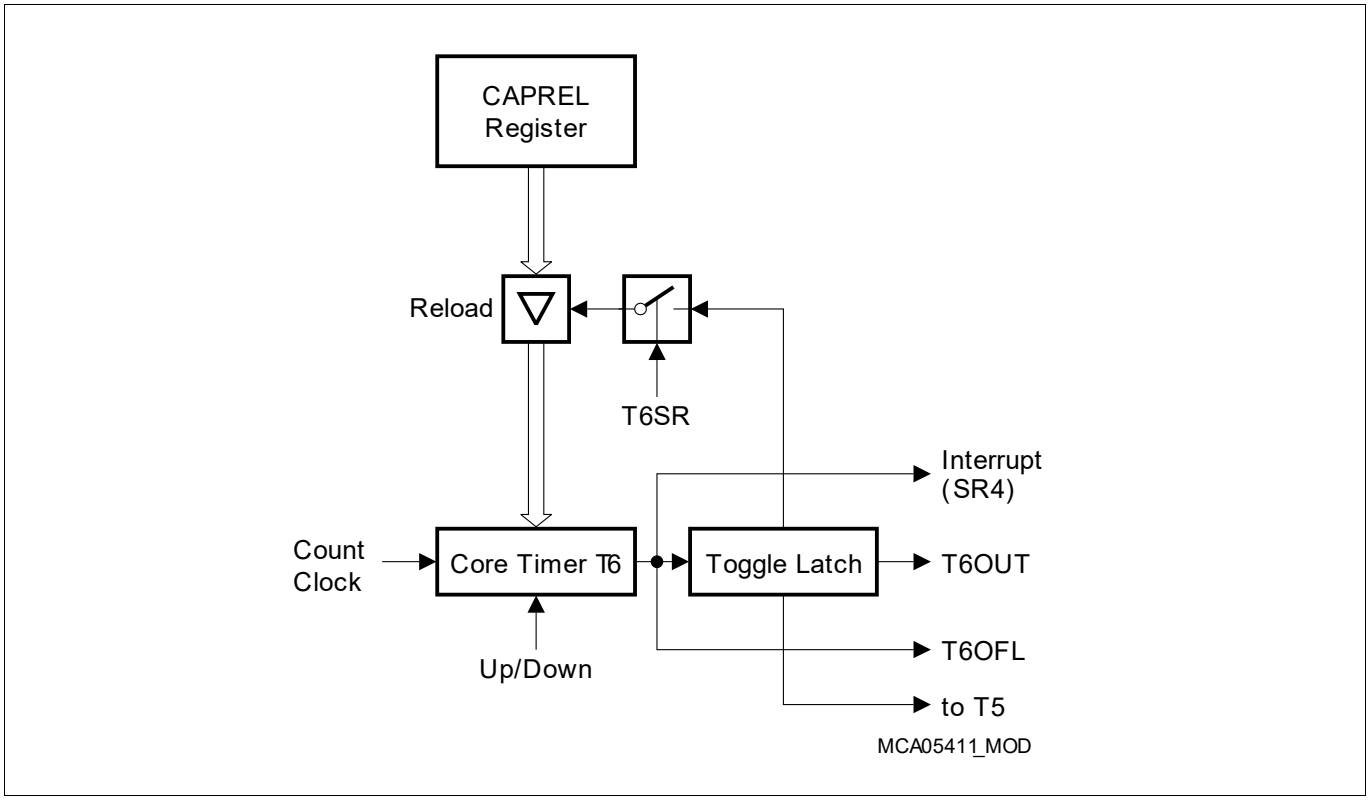


Figure 227 Capture/Reload Register CAPREL in Reload Mode

Capture/Reload Register CAPREL in Capture-And-Reload Mode

Since the reload function and the capture function of register CAPREL can be enabled individually by bits T5SC and T6SR, the two functions can be enabled simultaneously by setting both bits. This feature can be used to generate an output frequency that is a multiple of the input frequency.

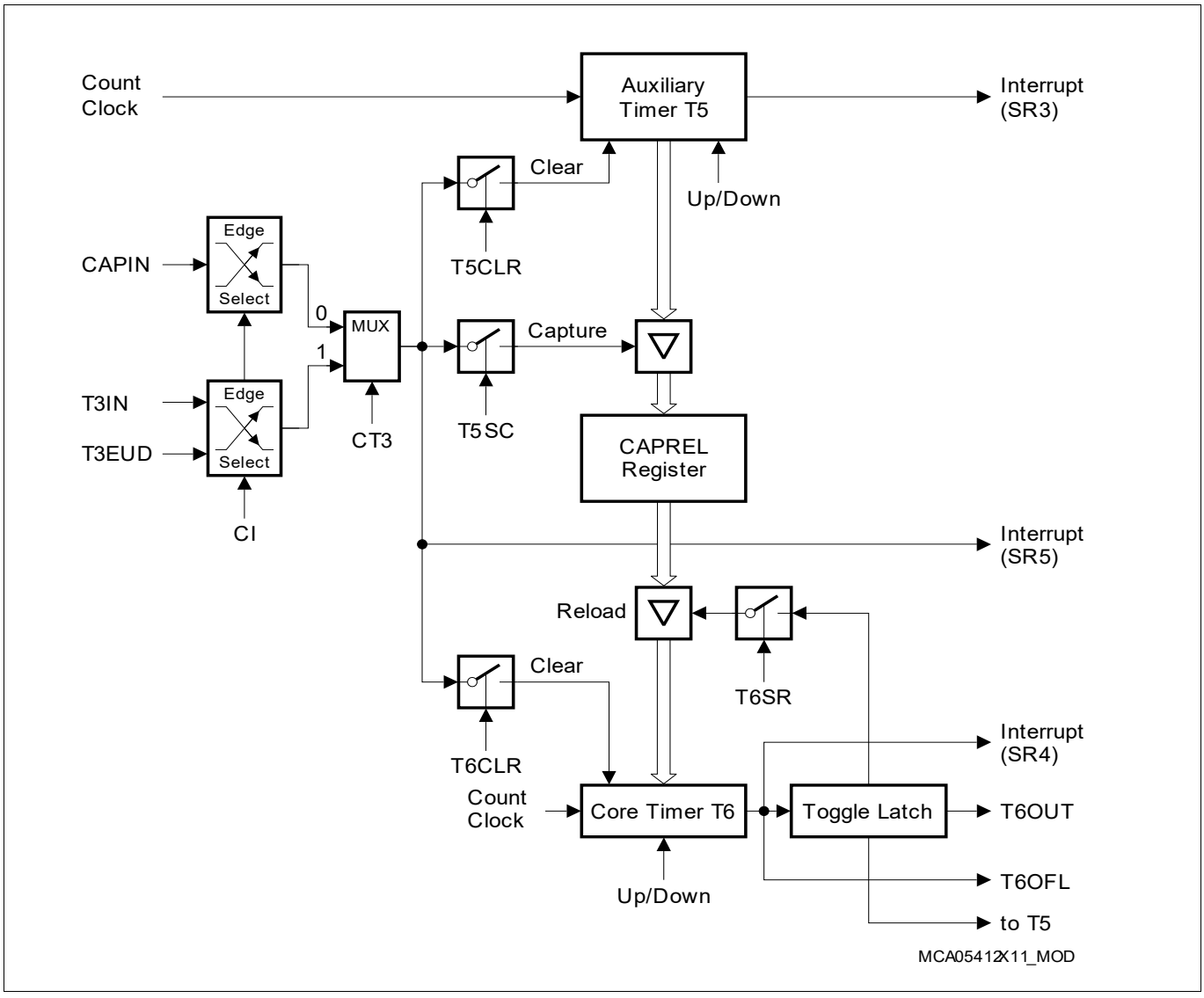


Figure 228 Capture/Reload Register CAPREL in Capture-And-Reload Mode

This combined mode can be used to detect consecutive external events which may occur aperiodically, but where a finer resolution, that means, more ‘ticks’ within the time between two external events is required.

For this purpose, the time between the external events is measured using timer T5 and the CAPREL register. Timer T5 runs in Timer Mode counting up with a frequency of e.g. $f_{GPT}/32$. The external events are applied e.g. to pin CAPIN (or other pins supporting capture mode, see **Table 230**). When an external event occurs, the contents of timer T5 are latched into register CAPREL and timer T5 is cleared (T5CLR = 1). Thus, register CAPREL always contains the correct time between two events, measured in timer T5 increments. Timer T6, which runs in Timer Mode counting down with a frequency of e.g. $f_{GPT}/4$, uses the value in register CAPREL to perform a reload on underflow. This means, the value in register CAPREL represents the time between two underflows of timer T6, now measured in timer T6 increments. Since (in this example) timer T6 runs 8 times faster than timer T5, it will underflow 8 times within the time between two external events. Thus, the underflow signal of timer T6 generates 8 ‘ticks’. Upon each underflow, the interrupt request SR4 will be activated and bit T6OTL will be toggled. The state

of T6OTL may be output on pin T6OUT. This signal on T6OUT has 8 times more transitions than the signal which is applied to the selected input (CAPIN in this example).

*Note: The underflow signal of Timer T6 can furthermore be used to clock functional units in other modules, e.g. one or more of the timers of the capture/compare units CCU6 (connections see **Section 30.6**). This gives the user the possibility to set compare events based on a finer resolution than that of the external events. This connection is accomplished via signal T6OFL.*

Capture Correction

A certain deviation of the output frequency is generated by the fact that timer T5 will count actual time units (e.g. T5 running at 1 MHz will count up to the value $64_H/100_D$ for a 10 kHz input signal), while T6OTL will only toggle upon an underflow of T6 (i.e. the transition from 0000_H to $FFFF_H$). In the above mentioned example, T6 would count down from 64_H, so the underflow would occur after 101 timing ticks of T6. The actual output frequency then is 79.2 kHz, instead of the expected 80 kHz.

This deviation can be compensated for by using T6 overflows. In this case, T5 counts down and T6 counts up. Upon a signal transition at the selected input pin(s), e.g. on CAPIN, the count value in T5 is captured into CAPREL and T5 is cleared to 0000_H. In its next clock cycle, T5 underflows to $FFFF_H$, and continues to count down with the following clocks. T6 is reloaded from CAPREL upon an overflow, and continues to count up with its following clock cycles (8 times faster in the above example). In this case, T5 and T6 count the same number of steps with their respective internal count frequency.

In the above example, T5 running at 1 MHz will count down to the value $FF9C_H/-100_D$ for a 10 kHz input signal applied e.g. at CAPIN, while T6 counts up from $FF9C_H$ through $FFFF_H$ to 0000_H. So the overflow occurs after 100 timing ticks of T6, and the actual output frequency at T6OUT then is the expected 80 kHz.

However, in this case CAPREL does not directly contain the time between two external events, but rather its 2's complement. Software will have to convert this value, if it is required for the operation.

Combined Capture Modes

For incremental interface applications in particular, several timer features can be combined to obtain dynamic information such as speed, acceleration, or deceleration. The current position itself can be obtained directly from the timer register (T2, T3, T4).

The time information to determine the dynamic parameters is generated by capturing the contents of the free-running timer T5 into register CAPREL. Two trigger sources for this event can be selected:

- Capture trigger on sensor signal transitions
- Capture trigger on position read operations

Capturing on sensor signal transitions is available for timer T3 inputs. This mode is selected by setting bit CT3 and selecting the intended signal(s) via bitfield CI in register T5CON. CAPREL then indicates the time between two selected transitions (measured in T5 counts).

Capturing on position read operations is available for timers T2, T3, and T4. This mode is selected by clearing bit CT3 and selecting the rising edge via bitfield CI in register T5CON. Bitfield ISCAPIN in register PISEL then selects either a read access from T3 or a read access from any of T2 or T3 or T4. CAPREL then indicates the time between two read accesses.

In general, GPT12 has no destructive read mechanisms, except for this special operation mode ‘capture on position read operations’, where register CAPREL is intentionally updated after a read of T3 or T2/T4 (and T5 or T6 are optionally cleared). In this case, also the associated service request flag in register SRC_GPT120CIRQ is set.

Note: If mode ‘capturing on position read operations’ is selected, and the corresponding timer (T3, or any of T2, T3, T4) is read by a debugger, a capture event may be triggered (T5 is captured into CAPREL, and T5 or T6 may optionally be cleared (if T5CLR = 1 or T6CLR = 1)).

These operating modes directly support the measurement of position and rotational speed. Acceleration and deceleration can then be determined by evaluating subsequent speed measurements.

30.3.6 GPT2 Clock Signal Control

All actions within the timer block GPT2 are triggered by transitions of its basic clock. This basic clock is derived from the module clock f_{GPT} by a basic block prescaler, controlled by bitfield BPS2 in register T6CON (see **Figure 217**). The count clock can be generated in two different ways:

- **Internal count clock**, derived from GPT2's basic clock via a programmable prescaler, is used for (Gated) Timer Mode.
- **External count clock**, derived from the timer's input pin(s), is used for Counter Mode.

For both ways, the basic clock determines the maximum count frequency and the timer's resolution:

Table 223 Basic Clock Selection for Block GPT2

Block Prescaler[1)]	BPS2 = 01_B	BPS2 = 00_B[2)]	BPS2 = 11_B	BPS2 = 10_B
Prescaling Factor for GPT2: F(BPS2)	F(BPS2) = 2	F(BPS2) = 4	F(BPS2) = 8	F(BPS2) = 16
Maximum External Count Frequency	$f_{GPT}/4$	$f_{GPT}/8$	$f_{GPT}/16$	$f_{GPT}/32$
Input Signal Stable Time	2 x t_{GPT}	4 x t_{GPT}	8 x t_{GPT}	16 x t_{GPT}

1) Please note the non-linear encoding of bitfield BPS2.
2) Default after reset.

Notes

1. *The GPT2 module uses a finite state machine to control the actions. Since multiple interactions are possible between the timers (T5, T6) and register CAPREL, these elements are processed sequentially. However, all actions are normally completed within one basic clock cycle. The GPT2 state machine has 4 states (2 states when BPS2 = 01_B) and processes T6 before T5.*
2. *When initializing the GPT2 block after reset, and the block prescaler BPS2 in register T6CON needs to be set to a value different from its default value (00_B), it must be initialized first before any mode involving external trigger signals is configured. These modes include counter, capture, and reload mode. Otherwise, unintended count/capture/reload events may occur during the first basic clock cycle.*
   In this case, or when changing BPS2 during operation of the GPT2 block, disable related interrupts before modification of BPS2, and afterwards clear the corresponding service request flags and re-initialize those registers (T5, T6, CAPREL) that might be affected by a count/capture/reload event.

Internal Count Clock Generation

In Timer Mode and Gated Timer Mode, the count clock for each GPT2 timer is derived from the GPT2 basic clock by a programmable prescaler, controlled by bitfield TxI in the respective timer's control register TxCON.

The count frequency f_{Tx} for a timer Tx and its resolution r_{Tx} are scaled linearly with lower clock frequencies, as can be seen from the following formula:

$$f_{Tx} = \frac{f_{GPT}}{F(BPS2) \times 2^{<TxI>}} \qquad r_{Tx}[\mu s] = \frac{F(BPS2) \times 2^{<TxI>}}{f_{GPT}[MHz]} \tag{30.2}$$

The effective count frequency depends on the common module clock prescaler factor F(BPS2) as well as on the individual input prescaler factor $2^{<TxI>}$. **Table 227** summarizes the resulting overall divider factors for a GPT2 timer that result from these cascaded prescalers.

Table 224 lists GPT2 timer's parameters (such as count frequency, resolution, and period) resulting from the selected overall prescaler factor (F(BPS2) x $2^{<Txl>}$) and the module clock f_{GPT}. Note that some numbers may be rounded.

Table 224 GPT2 Timer Parameters

Overall Prescaler Factor	Example 1: Module Clock f_{GPT} = 100 MHz			Example 2: Module Clock f_{GPT} = 66.5 MHz		
	Frequency	Resolution	Period	Frequency	Resolution	Period
2	50 MHz	20 ns	1.311 ms	33.25 MHz	30.1 ns	1.97 ms
4	25 MHz	40 ns	2.621 ms	16.625 MHz	60.2 ns	3.94 ms
8	12.5 MHz	80 ns	5.243 ms	8.313 MHz	120.3 ns	7.88 ms
16	6.25 MHz	160 ns	10.49 ms	4.156 MHz	240.6 ns	15.77 ms
32	3.125 MHz	320 ns	20.97 ms	2.078 MHz	481.2 ns	31.54 ms
64	1.563 MHz	640 ns	41.94 ms	1.039 MHz	962.4 ns	63.07 ms
128	781.25 kHz	1.28 µs	83.89 ms	519.53 kHz	1.924 µs	126.14 ms
256	390.6 kHz	2.56 µs	167.8 ms	259.77 kHz	3.850 µs	252.29 ms
512	195.3 kHz	5.12 µs	335.5 ms	129.88 kHz	7.699 µs	504.58 ms
1024	97.7 kHz	10.24 µs	671.1 ms	64.94 kHz	15.398 µs	1.009 s
2048	48.8 kHz	20.48 µs	1.342 s	32.47 kHz	30.797 µs	2.018 s

External Count Clock Input

The external input signals of the GPT2 block are sampled with the GPT2 basic clock (see **Figure 217**). To ensure that a signal is recognized correctly, its current level (high or low) must be held active for at least one complete sampling period, before changing. A signal transition is recognized if two subsequent samples of the input signal represent different levels. Therefore, a minimum of two basic clock periods are required for the sampling of an external input signal. Thus, the maximum frequency of an input signal must not be higher than half the basic clock.

Table 225 summarizes the resulting requirements for external GPT2 input signals.

Table 225 GPT2 External Input Signal Limits

GPT2 Divider BPS2	Input Freq. Factor	Input Phase Duration	Example 1: Module Clock f_{GPT} = 100 MHz		Example 2: Module Clock f_{GPT} = 66.5 MHz	
			Max. Input Frequency	Min. Level Hold Time	Max. Input Frequency	Min. Level Hold Time
01_B	$f_{GPT}/4$	2 x t_{GPT}	25 MHz	20 ns	16.625 MHz	30.1 ns
00_B	$f_{GPT}/8$	4 x t_{GPT}	12.5 MHz	40 ns	8.313 MHz	60.2 ns
11_B	$f_{GPT}/16$	8 x t_{GPT}	6.25 MHz	80 ns	4.156 MHz	120.3 ns
10_B	$f_{GPT}/32$	16 x t_{GPT}	3.125 MHz	160 ns	2.078 MHz	240.6 ns

These limitations are valid for all external input signals to GPT2, including the external count signals in Counter Mode and the gate input signals in Gated Timer Mode as well as the capture/reload signals on input CAPREL.

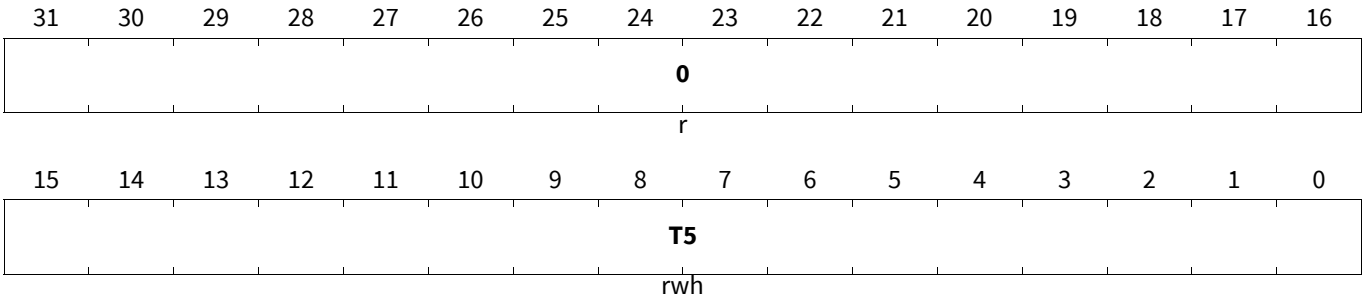
30.3.7 GPT2 Registers

Block GPT2 includes timer registers T5, T6, the capture/reload register CAPREL, and control registers T5CON and T6CON.

Timer T5 Register

T5

Timer T5 Register **(0040_H)** **Application Reset Value: $0000\ 0000_H$**

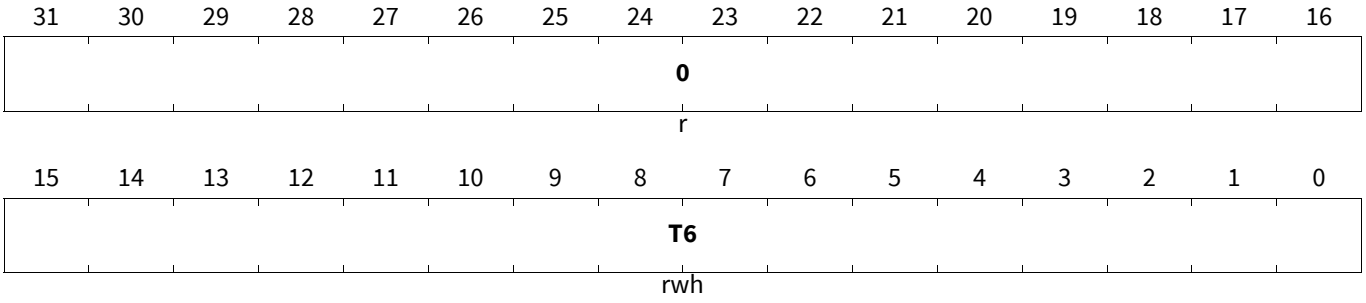


Field	Bits	Type	Description
T5	15:0	rwh	**Timer T5** Contains the current value of Timer T5.
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Timer T6 Register

T6

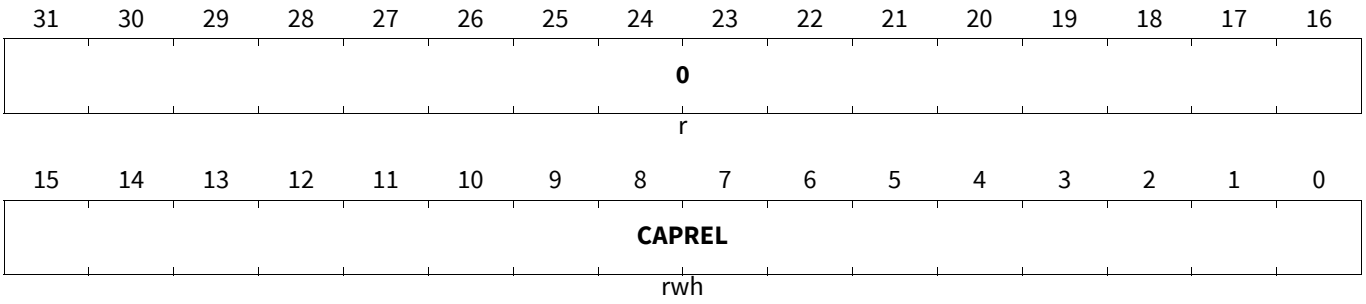
Timer T6 Register **(0044_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T6	15:0	rwh	**Timer T6** Contains the current value of Timer T6.
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Capture and Reload Register

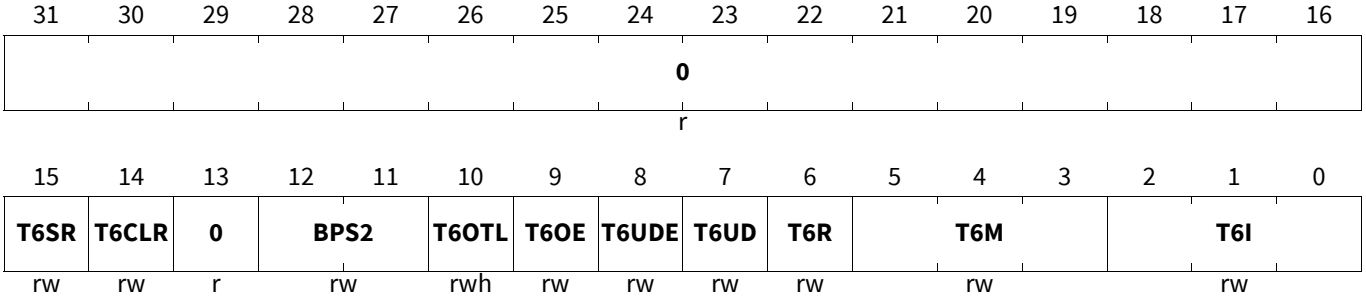
CAPREL
Capture and Reload Register **(0030_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CAPREL	15:0	rwh	**Current reload value or Captured value**
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Timer T6 Control Register

T6CON
Timer T6 Control Register **(0020_H)** **Application Reset Value: $0000\ 0000_H$**

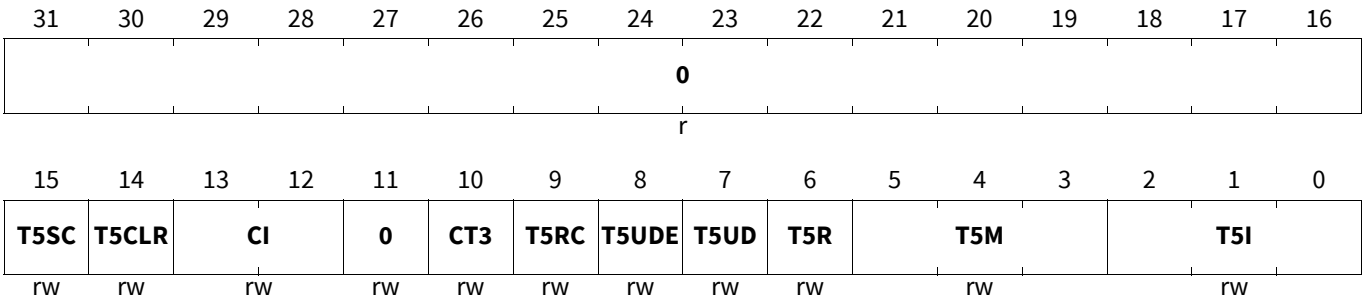


Field	Bits	Type	Description
T6I	2:0	rw	**Timer T6 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 227** for Timer Mode and Gated Timer Mode **Table 228** for Counter Mode
T6M	5:3	rw	**Timer T6 Mode Control (Basic Operating Mode)** 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reserved. Do not use this combination. … 111_B Reserved. Do not use this combination.

Field	Bits	Type	Description
T6R	6	rw	**Timer T6 Run Bit** 0_B Timer T6 stops 1_B Timer T6 runs
T6UD	7	rw	**Timer T6 Up/Down Control** *Note: This bit only directly controls count direction of T6 if bit T6UDE = 0.* 0_B Timer T6 counts up 1_B Timer T6 counts down
T6UDE	8	rw	**Timer T6 External Up/Down Enable** 0_B Count direction is controlled by bit T6UD; input T6EUD is disconnected 1_B Count direction is controlled by input T6EUD (see also **Table 226**)
T6OE	9	rw	**Overflow/Underflow Output Enable** 0_B Alternate Output Function Disabled 1_B State of timer T6 toggle latch is output on pin T6OUT
T6OTL	10	rwh	**Timer T6 Overflow Toggle Latch** Toggles on each overflow/underflow of timer T6. Can be set or cleared by software (see separate description)
BPS2	12:11	rw	**GPT2 Block Prescaler Control** Selects the basic clock for block GPT2 (see also **Section 30.3.6**) 00_B $f_{GPT}/4$ 01_B $f_{GPT}/2$ 10_B $f_{GPT}/16$ 11_B $f_{GPT}/8$
T6CLR	14	rw	**Timer T6 Clear Enable Bit** 0_B Timer T6 is not cleared on a capture event 1_B Timer T6 is cleared on a capture event
T6SR	15	rw	**Timer T6 Reload Mode Enable** 0_B Reload from register CAPREL Disabled 1_B Reload from register CAPREL Enabled
0	13, 31:16	r	**Reserved** Read as 0; should be written with 0.

Timer T5 Control Register

T5CON
Timer T5 Control Register **($001C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T5I	2:0	rw	**Timer T5 Input Parameter Selection** Depends on the operating mode, see respective sections for encoding: **Table 227** for Timer Mode and Gated Timer Mode **Table 229** for Counter Mode
T5M	5:3	rw	**Timer T5 Mode Control (Basic Operating Mode)** 000_B Timer Mode 001_B Counter Mode 010_B Gated Timer Mode with gate active low 011_B Gated Timer Mode with gate active high 100_B Reserved. Do not use this combination … 111_B Reserved. Do not use this combination
T5R	6	rw	**Timer T5 Run Bit** *Note: This bit only controls timer T5 if bit T5RC = 0.* 0_B Timer T5 stops 1_B Timer T5 runs
T5UD	7	rw	**Timer T5 Up/Down Control** *Note: This bit only directly controls count direction of T5 if bit T5UDE = 0.* 0_B Timer T5 counts up 1_B Timer T5 counts down
T5UDE	8	rw	**Timer T5 External Up/Down Enable** 0_B Count direction is controlled by bit T5UD; input T5EUD is disconnected 1_B Count direction is controlled by input T5EUD (see also **Table 226**)
T5RC	9	rw	**Timer T5 Remote Control** 0_B Timer T5 is controlled by its own run bit T5R 1_B Timer T5 is controlled by the run bit T6R of core timer T6, not by bit T5R

Field	Bits	Type	Description
CT3	10	rw	**Timer T3 Capture Trigger Enable** 0_B Capture trigger from input line CAPIN 1_B Capture trigger from T3 input lines T3IN and/or T3EUD
CI	13:12	rw	**Register CAPREL Capture Trigger Selection** 00_B Capture disabled 01_B Positive transition (rising edge) on CAPIN[1] or any transition on T3IN 10_B Negative transition (falling edge) on CAPIN or any transition on T3EUD 11_B Any transition (rising or falling edge) on CAPIN or any transition on T3IN or T3EUD
T5CLR	14	rw	**Timer T5 Clear Enable Bit** *Note:* 0_B Timer T5 is not cleared on a capture event 1_B Timer T5 is cleared on a capture event
T5SC	15	rw	**Timer T5 Capture Mode Enable** 0_B Capture into register CAPREL disabled 1_B Capture into register CAPREL enabled
0	11	rw	**Reserved** Has to be written with 0.
0	31:16	r	**Reserved** Read as 0; should be written with 0.

1) Rising edge must be selected for triggering the capture/clear operation by the internal GPT1 read signals (see bit field ISCAPIN in register **PISEL** and description in section **“Combined Capture Modes” on Page 43**).

30.3.8 Encoding of Specific Bitfields of GPT2 Registers

Table 226 GPT2 Timer Count Direction Control

Pin TxEUD	Bit TxUDE	Bit TxUD	Count Direction
X	0	0	Count Up
X	0	1	Count Down
0	1	0	Count Up
1	1	0	Count Down
0	1	1	Count Down
1	1	1	Count Up

Table 227 GPT2 Overall Prescaler Factors for Internal Count Clock (Timer Mode and Gated Timer Mode)

Individual Prescaler for Tx	Common Prescaler for Module Clock[1)]			
	BPS2 = 01_B	**BPS2 = 00_B**	**BPS2 = 11_B**	**BPS2 = 10_B**
TxI = 000_B	2	4	8	16
TxI = 001_B	4	8	16	32
TxI = 010_B	8	16	32	64
TxI = 011_B	16	32	64	128
TxI = 100_B	32	64	128	256
TxI = 101_B	64	128	256	512
TxI = 110_B	128	256	512	1024
TxI = 111_B	256	512	1024	2048

1) Please note the non-linear encoding of bitfield BPS2.

Table 228 GPT2 Core Timer T6 (Counter Mode) Input Edge Selection

T6I	Triggering Edge for Counter Increment/Decrement
000_B	None. Counter T6 is disabled
001_B	Positive transition (rising edge) on T6IN
010_B	Negative transition (falling edge) on T6IN
011_B	Any transition (rising or falling edge) on T6IN
$1XX_B$	Reserved. Do not use this combination

Table 229 GPT2 Auxiliary Timer T5 (Counter Mode) Input Edge Selection

T5I	Triggering Edge for Counter Increment/Decrement
$X00_B$	None. Counter T5 is disabled
001_B	Positive transition (rising edge) on T5IN
010_B	Negative transition (falling edge) on T5IN
011_B	Any transition (rising or falling edge) on T5IN
101_B	Positive transition (rising edge) of T6 toggle latch T6OTL

Table 229 GPT2 Auxiliary Timer T5 (Counter Mode) Input Edge Selection (cont'd)

T5I	Triggering Edge for Counter Increment/Decrement
110_B	Negative transition (falling edge) of T6 toggle latch T6OTL
111_B	Any transition (rising or falling edge) of T6 toggle latch T6OTL

Encoding of GPT2 Input Selection for Capture and Timer Clear Function

Table 230 CAPREL Register Input Edge Selection

CT3	CI	Triggering Signal/Edge for Capture Mode and/or T5/T6 Clear
X	00_B	None. Capture Mode is disabled.
0	01_B	Positive transition (rising edge) on CAPIN, or read operation on selected GPT1 timers[1)].
0	10_B	Negative transition (falling edge) on CAPIN.
0	11_B	Any transition (rising or falling edge) on CAPIN.
1	01_B	Any transition (rising or falling edge) on T3IN.
1	10_B	Any transition (rising or falling edge) on T3EUD.
1	11_B	Any transition (rising or falling edge) on T3IN or T3EUD.

1) Rising edge must be selected for triggering the capture/clear operation by the internal GPT1 read signals (see bit field ISCAPIN in register **PISEL** and description in section **"Combined Capture Modes" on Page 43**).

30.4 GPT12 Kernel Register Overview

Table 231 summarizes the GPT12 kernel registers, module external registers and BPI registers, and defines their addresses and reset.

Table 231 Register Overview - GPT12 (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	0000_H	U,SV	SV,E,P	Application Reset	58
PISEL	Port Input Select Register	0004_H	U,SV	U,SV,P	Application Reset	55
ID	Identification Register	0008_H	U,SV	BE	See page 57	57
T2CON	Timer T2 Control Register	0010_H	U,SV	U,SV,P	Application Reset	24
T3CON	Timer T3 Control Register	0014_H	U,SV	U,SV,P	Application Reset	22
T4CON	Timer T4 Control Register	0018_H	U,SV	U,SV,P	Application Reset	26
T5CON	Timer T5 Control Register	$001C_H$	U,SV	U,SV,P	Application Reset	50
T6CON	Timer T6 Control Register	0020_H	U,SV	U,SV,P	Application Reset	48
CAPREL	Capture and Reload Register	0030_H	U,SV	U,SV,P	Application Reset	48

Table 231 Register Overview - GPT12 (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
T2	Timer T2 Register	0034_H	U,SV	U,SV,P	Application Reset	21
T3	Timer T3 Register	0038_H	U,SV	U,SV,P	Application Reset	21
T4	Timer T4 Register	$003C_H$	U,SV	U,SV,P	Application Reset	22
T5	Timer T5 Register	0040_H	U,SV	U,SV,P	Application Reset	47
T6	Timer T6 Register	0044_H	U,SV	U,SV,P	Application Reset	47
OCS	OCDS Control and Status Register	$00E8_H$	U,SV	SV,P,OEN	See page 59	59
KRSTCLR	Kernel Reset Status Clear Register	$00EC_H$	U,SV	SV,E,P	Application Reset	62
KRST1	Kernel Reset Register 1	$00F0_H$	U,SV	SV,E,P	Application Reset	62
KRST0	Kernel Reset Register 0	$00F4_H$	U,SV	SV,E,P	Application Reset	61
ACCEN0	Access Enable Register 0	$00FC_H$	U,SV	SV,SE	Application Reset	60

30.5 General Module Operation

This section provides information about:

30.5.1 Input Selection

Each GPT12 input signal can be selected from a vector of two or four possible inputs by programming the port input select register **PISEL**. This permits to adapt the pin functionality of the device to the application requirements.
The output pins for the module output signals are chosen in the ports.

Naming convention (example):
The input vector T3IN[D:A] for input signal T3IN is composed of the signals T3INA to T3IND.

*Note: All functional inputs of the GPT12 module are synchronized to the internal basic clock of the GPT1 and GPT2 block, respectively. An edge of an input signal can only be correctly recognized if the high phase and the low phase are longer than one basic clock period. See **Table 216** for GPT1 external input signal limits, and **Table 225** for GPT2 external input signal limits.*

30.5.2 OCDS Suspend

The behavior of GPT12 upon an OCDS suspend request is controlled by the **OCS** register. GPT12 supports only Hard Suspend Mode.

Hard Suspend Mode

In Hard Suspend Mode the GPT12 kernel clock is switched off immediately. Reading and writing of registers is possible but will enable the kernel clock for a few cycles.

Attention: Register accesses with clocking in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so a GPT12 kernel reset might not be sufficient to bring the system into a defined state.

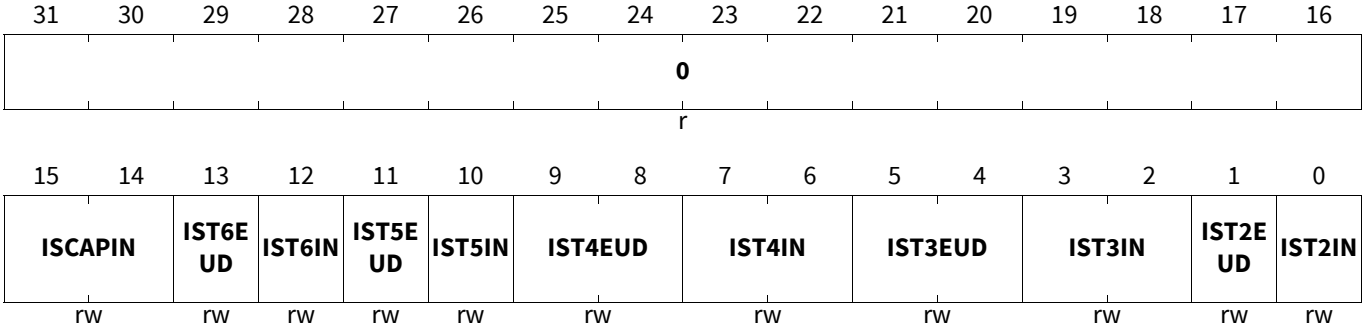
30.5.3 Miscellaneous GPT12 Registers

Port Input Select Register

Register PISEL contains bitfields selecting the actual input signal for the module inputs.

PISEL

Port Input Select Register **(0004_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
IST2IN	0	rw	**Input Select for T2IN** 0_B Signal T2INA is selected 1_B Signal T2INB is selected
IST2EUD	1	rw	**Input Select for T2EUD** 0_B Signal T2EUDA is selected 1_B Signal T2EUDB is selected
IST3IN	3:2	rw	**Input Select for T3IN** 00_B Signal T3INA is selected 01_B Signal T3INB is selected 10_B Signal T3INC is selected 11_B Signal T3IND is selected
IST3EUD	5:4	rw	**Input Select for T3EUD** 00_B Signal T3EUDA is selected 01_B Signal T3EUDB is selected 10_B Signal T3EUDC is selected 11_B Signal T3EUDD is selected
IST4IN	7:6	rw	**Input Select for T4IN** 00_B Signal T4INA is selected 01_B Signal T4INB is selected 10_B Signal T4INC is selected 11_B Signal T4IND is selected
IST4EUD	9:8	rw	**Input Select for T4EUD** 00_B Signal T4EUDA is selected 01_B Signal T4EUDB is selected 10_B Signal T4EUDC is selected 11_B Signal T4EUDD is selected
IST5IN	10	rw	**Input Select for T5IN** 0_B Signal T5INA is selected 1_B Signal T5INB is selected
IST5EUD	11	rw	**Input Select for T5EUD** 0_B Signal T5EUDA is selected 1_B Signal T5EUDB is selected

Field	Bits	Type	Description
IST6IN	12	rw	**Input Select for T6IN** 0_B Signal T6INA is selected 1_B Signal T6INB is selected
IST6EUD	13	rw	**Input Select for T6EUD** 0_B Signal T6EUDA is selected 1_B Signal T6EUDB is selected
ISCAPIN	15:14	rw	**Input Select for CAPIN** 00_B Signal CAPINA is selected 01_B Signal CAPINB is selected 10_B Signal CAPINC (Read trigger from T3) is selected 11_B Signal CAPIND (Read trigger from T2 or T3 or T4) is selected
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Identification Register

ID
Identification Register **(0008_H)** **Reset Value: Table 232**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODNUMBER															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODTYPE								MODREV							
r								r							

Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** This bit field indicates the revision number of the module (01_H = first revision).
MODTYPE	15:8	r	**Module Type** This bit field is $C0_H$. It defines a 32-bit module
MODNUMBER	31:16	r	**Module Number** This bit field defines the module identification number. For the GPT12 module the module identification number is 68_H.

Table 232 Reset Values of ID

Reset Type	Reset Value	Note
Application Reset	$0068\ C0XX_H$	

30.5.4 BPI Registers

This section describes the registers of the BPI (Bus Peripheral Interface).

Note: Register bits marked "r" in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI for the GPT12 module. Where a module kernel is connected to the CLC clock control interface, CLC controls the f_{GPT} module clock signal and Sleep Mode for the module.

Note: Upon an accepted Sleep Mode request (with EDIS = '1'), or upon a disable request (DISR = '1'), the GPT12 kernel clock is switched off immediately. Therefore, software should ensure that the system controlled by the GPT12 kernel has reached a safe state before triggering a Sleep Mode or module disable request.

CLC

Clock Control Register (0000_H) Application Reset Value: $0000\ 0003_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested. 1_B Module disable is requested.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module. 0_B Module is enabled. 1_B Module is disabled.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode. 0_B Sleep Mode request is regarded. Module is enabled to go into Sleep Mode. 1_B Sleep Mode request is disregarded: Sleep Mode cannot be entered upon a request.
0	2, 15:4, 31:16	r	**Reserved** Read as 0; should be written with 0.

It is recommended not to write to or read from module registers (except CLC) while the module is disabled. Write operations and read operations from registers that require a clock will generate a bus error.

OCDS Control and Status Register

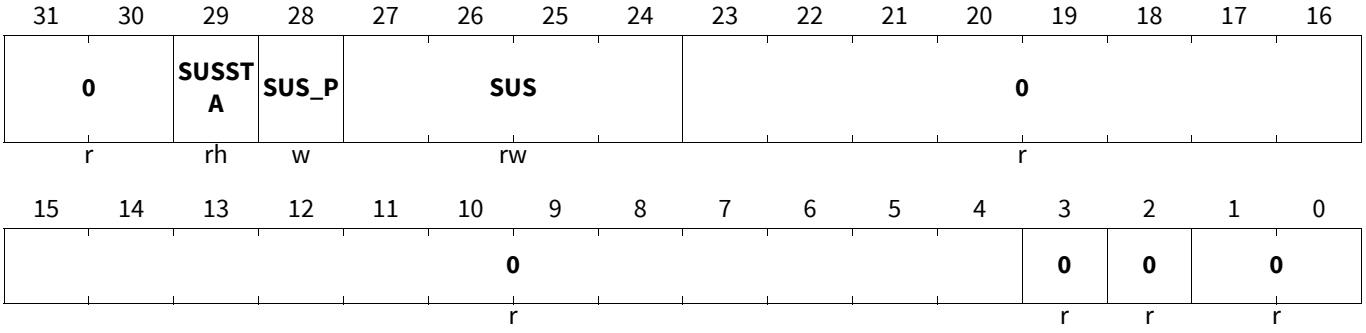
The OCDS Control and Status register OCS controls the module's behavior in suspend mode (used for debugging). The OCS register is cleared by Debug Reset.

The register can only be written when the OCDS is enabled. When OCDS is disabled the OCS suspend control is ineffective.

Currently, no trigger set is defined for GPT12. Therefore control bits for the OCDS Trigger Bus (OTGB) in OCS.[3:0] are described as reserved.

OCS

OCDS Control and Status Register **($00E8_H$)** **Reset Value: Table 234**



Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. **others**, reserved (will not suspend).
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	3:0, 23:4, 31:30	r	**Reserved** Read as 0; must be written with 0.

Table 233 Access Mode Restrictions of OCS sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **SUS_P**	rw	SUS	set SUS_P during write access
write 1 to **TG_P**	r	SUS	set TG_P during write access
	rw		

Table 234 Reset Values of OCS

Reset Type	Reset Value	Note
Debug Reset	0000 0000$_H$	

Note: For additional hints, refer to ***"OCDS Suspend" on Page 55****.*

Access Enable Register 0

The Access Enable Register 0 controls write access[1] for transactions with the on chip bus master TAG IDs 00 0000$_B$ to 01 1111$_B$ (see On Chip Bus chapter for the mapping of product TAG ID <-> master peripheral). Register ACCEN0 provides one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENy: EN0 -> TAG ID 00 0000$_B$, EN1 -> TAG ID 00 0001$_B$, … , EN31 -> TAG ID 01 1111$_B$.

ACCEN0
Access Enable Register 0 (00FC$_H$) Application Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENy (y=0-31)	y	rw	**Access Enable for Master TAG ID y** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID y 0_B No write access Write access will not be executed 1_B Write access will be executed

Individual Module Reset

The Kernel Reset Registers **KRST0**/**KRST1** are used to reset the related module kernel. Kernel Reset Registers 0 and 1 each include bit RST. To reset a module kernel it is necessary to set the RST bits in both Kernel Reset Registers. They will be cleared automatically with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set in the same clock cycle the RST bit is cleared. This bit indicates that a kernel reset was processed. Bit RSTSTAT can be cleared by setting bit CLR in register **KRSTCLR**.

A module kernel reset has the following effects:

1) The BPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

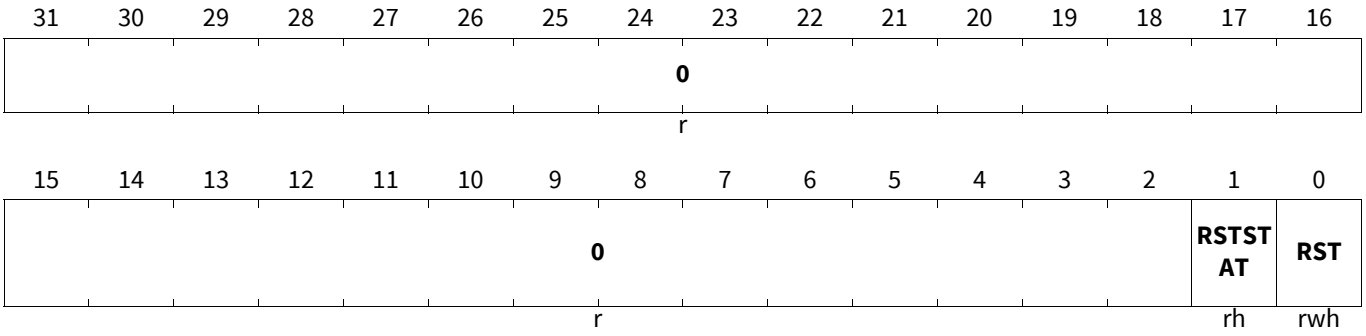
Table 235 Effects of a Module Kernel Reset

Register	Executed Action
CLC, OCS, ACCEN0	No influence
ID	No influence
KRST0	Bit RST is cleared automatically after reset execution, RSTSTAT indicates a module kernel reset, cleared via KRSTCLR
KRST1	Bit RST is cleared automatically after reset execution
KRSTCLR	No influence
Other registers	Reset to their defined reset values

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

Kernel Reset Register 0

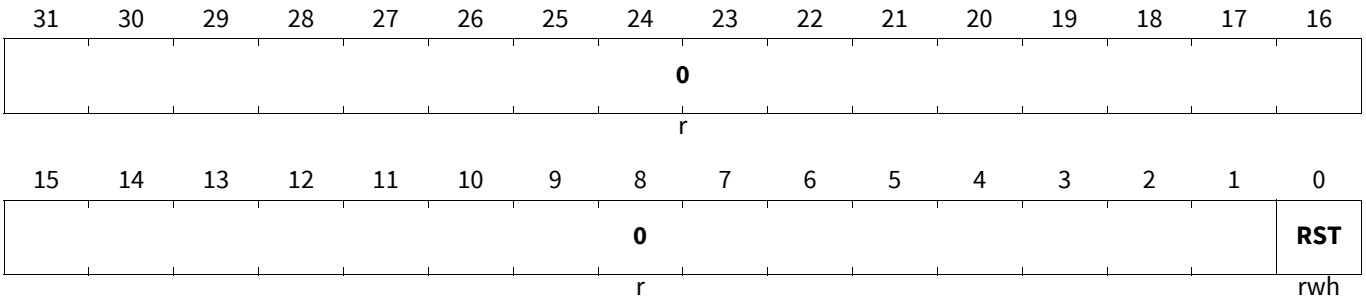
KRST0
Kernel Reset Register 0 (00F4$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set after the execution of a kernel reset in the same clock cycle in which the reset bits are cleared. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

KRST1
Kernel Reset Register 1 **($00F0_H$)** **Application Reset Value: $0000\ 0000_H$**

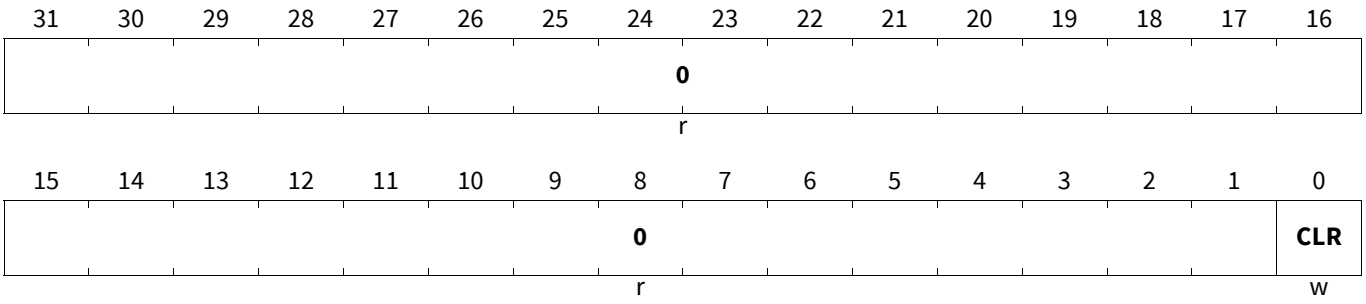


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear Register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR
Kernel Reset Status Clear Register **($00EC_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

30.6 Implementation of the GPT12 Module

The following table shows the interface signals of the GPT12 module, available for connections with other modules or pins in the device.

The GPT12 module is clocked with the SPB_Bus clock, so $f_{GPT} = f_{SPB}$.

Table 236 List of GPT12 Interface Signals

Interface Signals	I/O	Description
CIRQ_INT	out	**GPT120 CAPREL Service Request**
T2_INT	out	**GPT120 T2 Overflow/Underflow Service Request**
T3_INT	out	**GPT120 T3 Overflow/Underflow Service Request**
T4_INT	out	**GPT120 T4 Overflow/Underflow Service Request**
T5_INT	out	**GPT120 T5 Overflow/Underflow Service Request**
T6_INT	out	**GPT120 T6 Overflow/Underflow Service Request**
T3OUT	out	**External output for overflow/underflow detection of core timer T3**
T6OUT	out	**External output for overflow/underflow detection of core timer T6**
SR0	out	**Overflow/underflow service request of timer T2**
SR1	out	**Overflow/underflow service request of core timer T3**
SR2	out	**Overflow/underflow service request of timer T4**
SR3	out	**Overflow/underflow service request of timer T5**
SR4	out	**Overflow/underflow service request of core timer T6**
SR5	out	**Transition detection on CAPIN service request**
T2INA	in	**Trigger/gate input of timer T2**
T2INB		
T3INA	in	**Trigger/gate input of core timer T3**
T3INB		
T3INC		
T3IND		
T4INA	in	**Trigger/gate input of timer T4**
T4INB		
T4INC		
T4IND		
T5INA	in	**Trigger/gate input of timer T5**
T5INB		
T6INA	in	**Trigger/gate input of core timer T6**
T6INB		
CAPINA	in	**Trigger input to capture value of timer T5 into CAPREL register**
CAPINB		
T2EUDA	in	**Count direction control input of timer T2**
T2EUDB		

Table 236 List of GPT12 Interface Signals (cont'd)

Interface Signals	I/O	Description
T3EUDA	in	**Count direction control input of core timer T3**
T3EUDB		
T3EUDC		
T3EUDD		
T4EUDA	in	**Count direction control input of timer T4**
T4EUDB		
T4EUDC		
T4EUDD		
T5EUDA	in	**Count direction control input of timer T5**
T5EUDB		
T6EUDA	in	**Count direction control input of core timer T6**
T6EUDB		
T6OFL	out	**Overflow/underflow signal of timer T6**

30.7 Revision History

Table 237 Revision History

Reference	Change to Previous Version	Comment
V2.2.3		
---	No change	
V3.0.0		
Page 60	Describe details of the module kernel reset	
Page 1	Add crossreferences for easier navigation	
Page 59	Add reference to register OCS	
Page 58	Add info concerning access to register CLC	
Page 48	Bitfield description for “T6M” in register “Timer T6 Control Register” has changed	–
Page 50	Bitfield description for “T5M” in register “Timer T5 Control Register” has changed	–
V3.0.1		
–	No functional changes.	–
V3.0.2		
	No functional changes.	

31 Converter Control Block (CONVCTRL)

The converter control block summarizes control functions which are common for all converters implemented in the product. The following functions are provided:

- **Phase Synchronizer (PhSync)**
  provides a clock enable signal to synchronize the clock signals of all analog blocks

The converter control block contains the registers which are required to configure the associated functions.

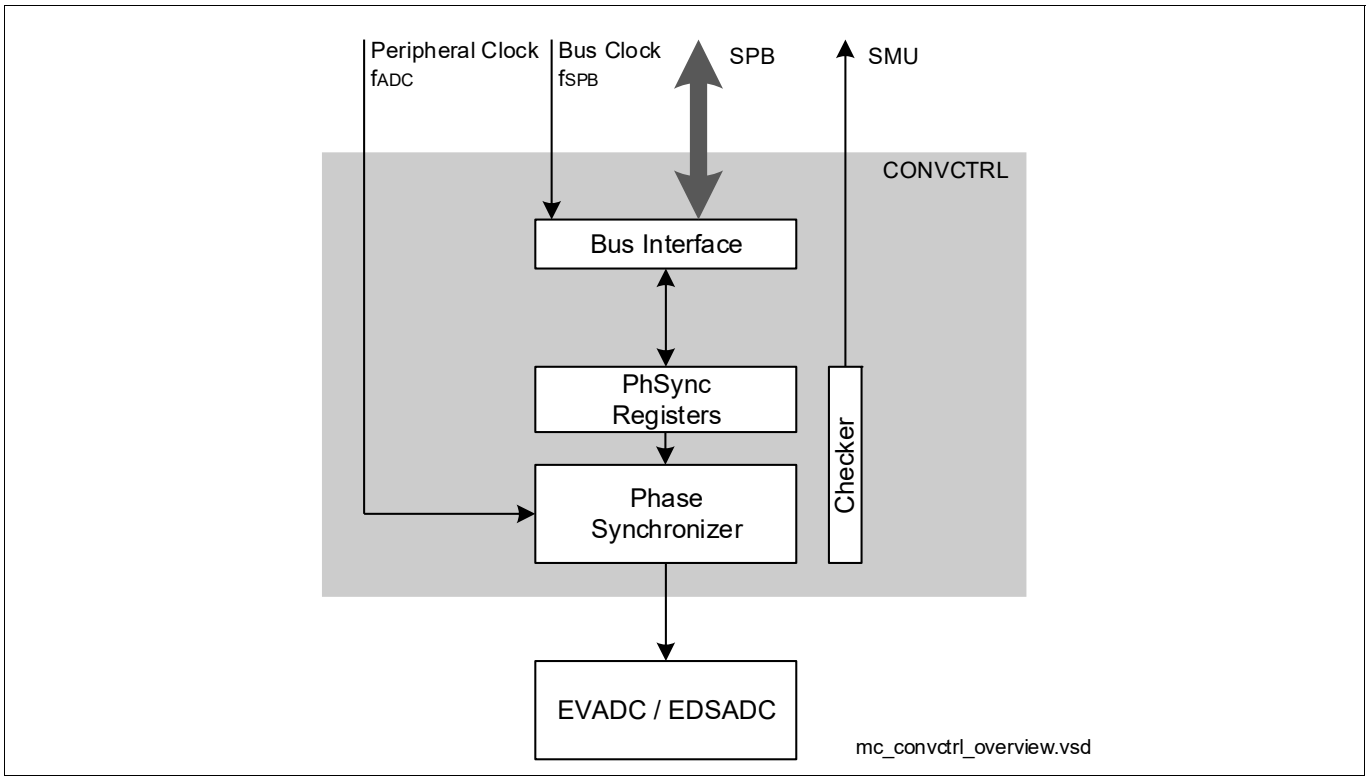


Figure 229 CONVCTRL Module Overview

In addition, the following sections can be found:

- **“Application Considerations” on Page 17**
- **“Summary of Registers and Locations” on Page 20**

Attention: ***This chapter describes the CONVCTRL of the TC3xx family, encompassing the features and functions of all family members.***
The specific characteristics of a product are described in the product-specific appendix.
This product-specific appendix specifies deviations (e.g. downgrades, etc.) from this family documentation.

31.1 Configuration of the CONVCTRL

The configuration of the functional units is done via dedicated registers which are defined in the corresponding functional section. General registers which are common for all blocks are defined here.

Module Identification Register

ID
Module Identification Register **(008_H)** **PowerOn Reset Value: 00FF C001$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MOD_NUMBER															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOD_TYPE								MOD_REV							
r								r							

Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision** Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	15:8	r	**Module Type** This internal marker is fixed to $C0_H$.
MOD_NUMBER	31:16	r	**Module Number** Indicates the module identification number ($00FF_H$ = CONVCTRL)

Clock Control Register

The Clock Control Register allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. It controls the module clock signal and the reactivity to the sleep signal.

CLC
Clock Control Register **(000_H)** **Application Reset Value: 0000 0003$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B On request: enable the module clock 1_B Off request: stop the module clock
DISS	1	rh	**Module Disable Status Bit** 0_B Module clock is enabled 1_B Off: module is not clocked
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's reaction to sleep mode. 0_B Sleep mode request is enabled and functional 1_B Module disregards the sleep mode control signal
0	2, 31:4	r	**Reserved, write 0, read as 0**

It is recommended not to write to or read from module registers (except CLC) while the module is disabled. Write operations will generate a bus error.

When the module is disabled (DISR = 1_B) while in suspend state, the corresponding status bit (DISS = 1_B) will only be set after several clock cycles. To generate them, writes to CLC need to be repeated. See OCS note on **Page 4**.

OCDS Control and Status Register

The OCDS Control and Status register OCS controls the module's behavior in suspend mode (used for debugging).

Register OCS is cleared by Debug Reset. It can only be written when OCDS is enabled.

If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective.

OCS

OCDS Control and Status Register **(028_H)** **Reset Value: Table 239**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**SUSSTA**	**SUS_P**	**SUS**				**0**							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Suspend mode 1: Stop generating synchronization pulses 2_H Suspend mode 2: Disable the phase synchronizer (Synchronization signal constantly active) 3_H Reserved … F_H Reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:0, 31:30	r	**Reserved, write 0, read as 0**

Table 238 Access Mode Restrictions of OCS sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **SUS_P**	rw	SUS	Set SUS_P during write access
(default)	r	SUS	

Table 239 Reset Values of OCS

Reset Type	Reset Value	Note
PowerOn Reset	$0000\ 0000_H$	
Debug Clear	$0000\ 0000_H$	

*Note: During write access in suspend state the clocks are activated for a few cycles to accomplish the access. To prevent state machines from advancing due to this, it is recommended to stop the phase synchronizer by clearing bitfield PHSDIV = 0000_B in register **PHSCFG**.*

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions with the on-chip bus master TAG IDs $00\ 0000_B$ to $01\ 1111_B$ (see On-Chip Bus chapter for the mapping of product TAG ID <-> master peripheral). Register ACCEN0 provides one enable bit for each possible TAG ID encoding. Register ACCEN0 itself is protected by the Safety Endinit feature.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID $00\ 0000_B$, EN1 -> TAG ID $00\ 0001_B$, … , EN31 -> TAG ID $01\ 1111_B$ (TAG IDs $1X\ XXXX_B$ are not used).

1) The Access Enable functionality controls only write transactions to registers CLC, OCS, KRSTx, and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

ACCEN0
Access Enable Register 0 **(03C$_H$)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B No Write access 1_B Write access will be executed

Individual Module Reset

The Kernel Reset Registers **KRST0**/**KRST1** are used to reset the related module kernel. Kernel Reset Registers 0 and 1 each include bit RST. To reset a module kernel it is necessary to set the RST bits in both Kernel Reset Registers. They will be cleared automatically with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set in the same clock cycle the RST bit is cleared. This bit indicates that a kernel reset was processed. Bit RSTSTAT can be cleared by setting bit CLR in register **KRSTCLR**.

A module kernel reset has the following effects:

Table 240 Effects of a Module Kernel Reset

Register	Executed Action
CLC, OCS, ACCEN0	No influence
ID	No influence
KRST0	Bit RST is cleared automatically after reset execution, RSTSTAT indicates a module kernel reset, cleared via KRSTCLR
KRST1	Bit RST is cleared automatically after reset execution
KRSTCLR	No influence
Other registers	Reset to their defined reset values

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

Kernel Reset Register 0

KRST0

Kernel Reset Register 0 (034_H) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RSTSTAT	RST
r														rh	rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** Request a kernel reset. The reset is executed if the reset bits of both kernel reset registers are set. RST is cleared after the kernel reset was executed. 0_B No action 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** Indicates an executed kernel reset. RSTSTAT is set after the execution of a kernel reset in the same clock cycle in which the reset bits are cleared. Clear RSTSTAT by setting bit CLR in register KRSTCLR. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved, write 0, read as 0**

Kernel Reset Register 1

KRST1

Kernel Reset Register 1 (030_H) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															RST
r															rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** Request a kernel reset. The reset is executed if the reset bits of both kernel reset registers are set. RST is cleared after the kernel reset was executed. 0_B No action 1_B A kernel reset was requested
0	31:1	r	**Reserved, write 0**

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **($02C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**CLR**
r															w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved, write 0, read as 0**

Converter Control Block Control Register

To prevent an inadvertent activation or modification of converter control functions, all registers are accessible only when access is enabled via register CCCTRL.

CCCTRL

Converter Control Block Control Register **($07C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TC				**0**											
rw				r											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
TC	31:28	rw	**Test Control** Not listed combinations write-protect CONVCTRL registers. B_H Access to CONVCTRL registers is enabled
0	27:0	r	**Reserved, write 0, read as 0**

31.2 Phase Synchronizer (PhSync)

The phase synchronizer provides a specific clock enable signal to synchronize the operation of all analog blocks within modules EVADC and EDSADC.

These analog blocks use the voltage references (V_{AREF} and V_{AGND}) and synchronizing their clock edges avoids performance losses caused by mutual cross-coupling via the reference lines.

The CONVCTRL ensures that all switching activities within the connected analog blocks occur at the same clock edge of f_{PER} so the ringing that occurs on the reference lines does not disturb the operation of another analog block.

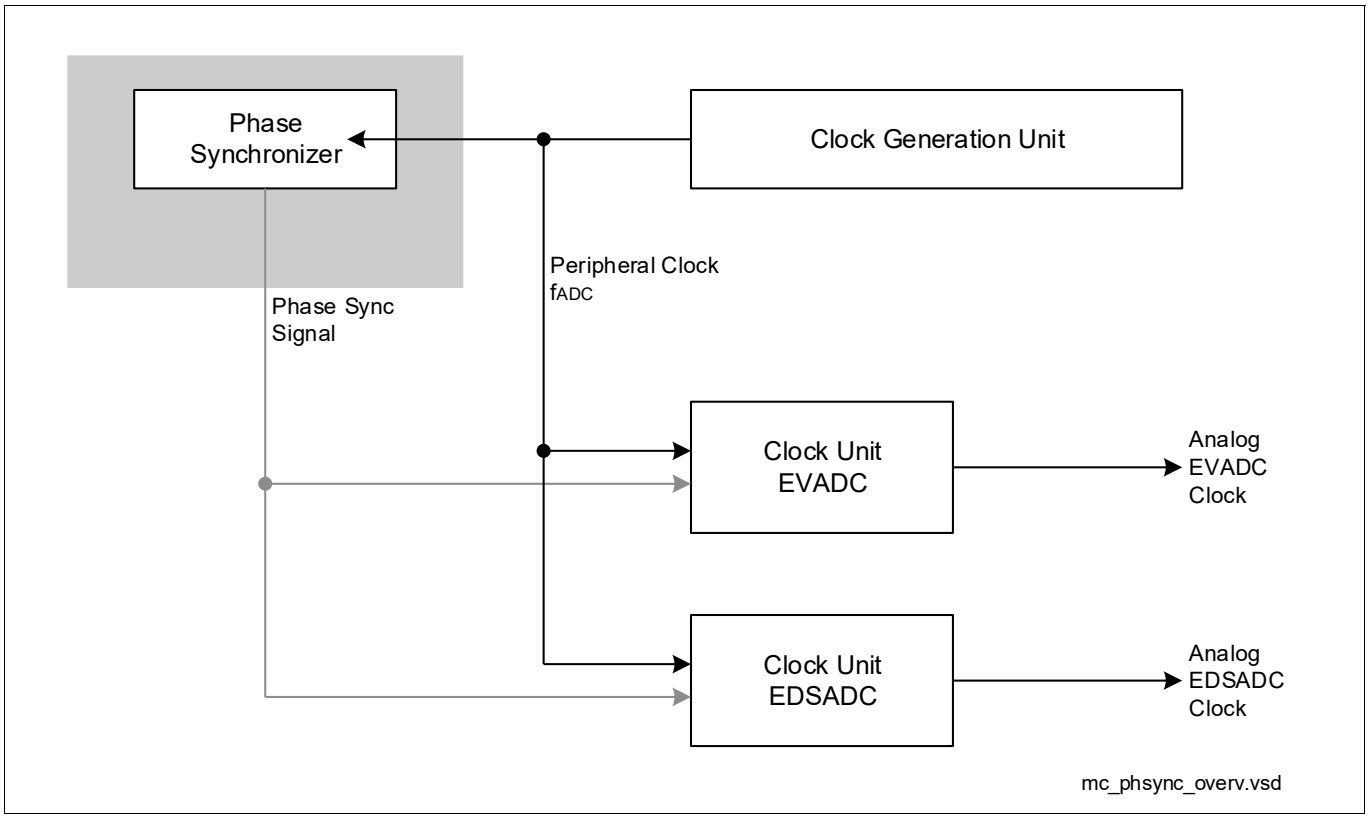


Figure 230 CONVCTRL Module Overview

Features

The following features describe the functionality of the PhSync:

- Synchronization of all analog blocks
  - Centrally generated clock enable signal
  - Local synchronization stages within each module
- Target frequency selectable to adapt to system requirements
- Based on the peripheral clock signal f_{PER} (160 MHz)

You will find the following major sections within this chapter:

- **“Introduction and Basic Structure” on Page 10**
- **“Operation of the PhSync” on Page 11**
- **“Generation of the Analog Clock Signal” on Page 12**
- **“Safety Measures” on Page 13**
- **“Configuration of the PhSync” on Page 16**

31.2.1 Introduction and Basic Structure

The Phase Synchronizer module (PhSync) provides a centralized clock enable signal for all analog blocks (EVADC, EDSADC). This synchronization ensures that the analog blocks do not disturb each other by drawing current peaks from the reference voltage lines at adjacent edges of the basic peripheral clock f_{PER}.

The frequency of the resulting analog clock enable signal is configurable within a certain range.

The PhSync broadcasts its clock enable signal, while each connected module locally generates its own analog clock signal based on the PhSync signal. With this approach no additional clock balancing is required.

The basic module clock is the peripheral clock signal f_{PER}.

The figures below show the analog clock generation mechanisms used in the connected modules.

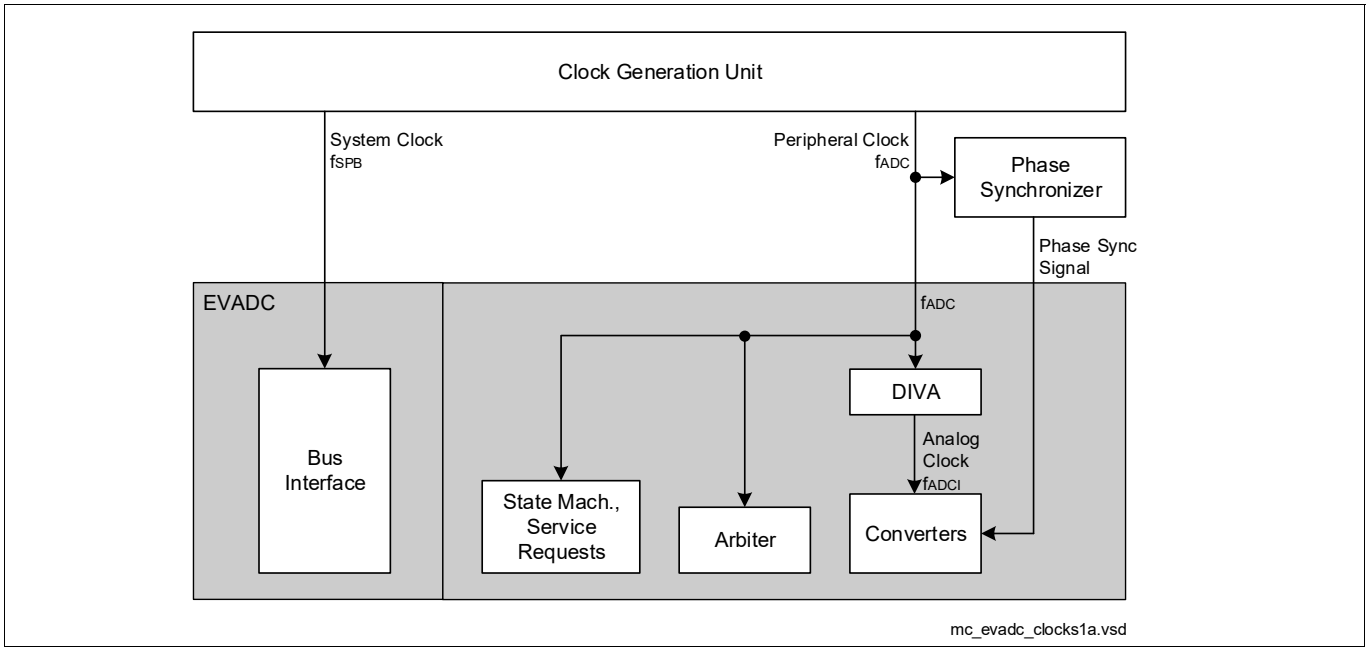


Figure 231 Analog Clock Generation in the EVADC

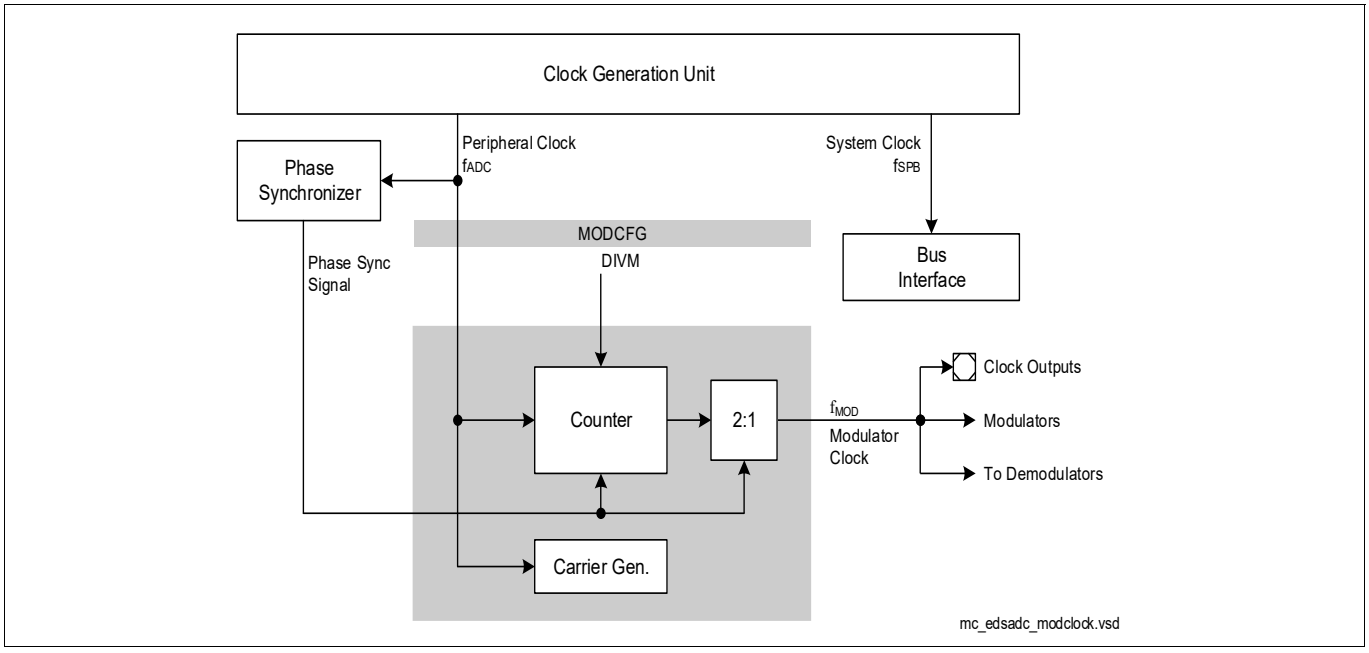


Figure 232 Analog Clock Generation in the EDSADC

31.2.2 Operation of the PhSync

The Phase Synchronizer counts peripheral clock (f_{PER}) cycles and generates a clock enable signal after a configurable number of clock cycles. The prescaling factor can be selected from 2 to 16, which equals an analog clock frequency of 80 MHz to 10 MHz (assuming f_{PER} = 160 MHz).

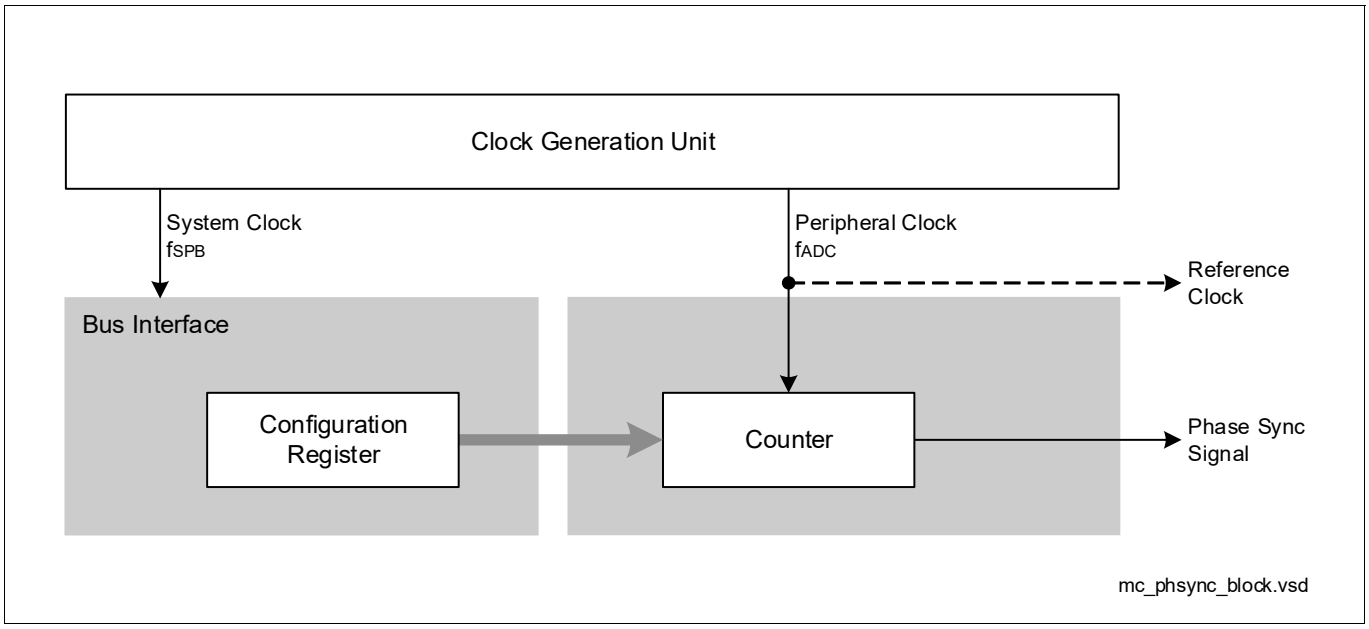


Figure 233 Basic PhSync Block Diagram

The phase synchronization signal triggers all local clock generation blocks to generate a clock edge with a subsequent[1)] rising edge of the common peripheral clock. The figure below shows the resulting timing for prescaler factors of 2 and 8.

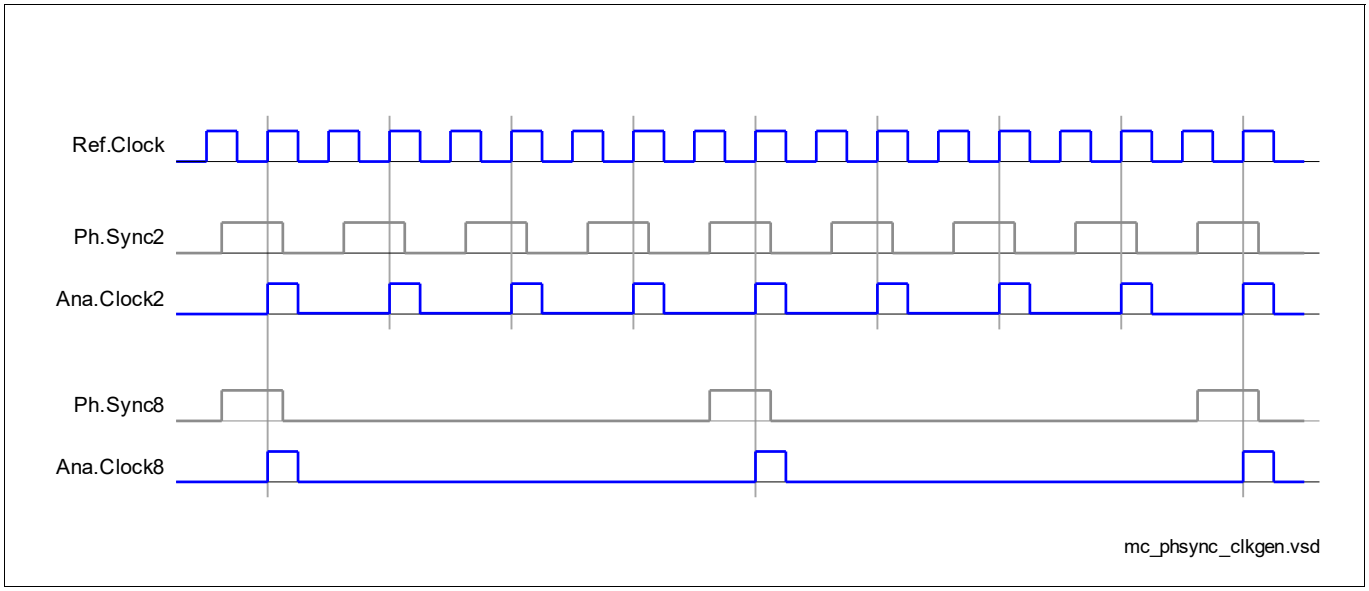


Figure 234 Clock Generation with the Phase Synchronization Signal (Pulse Swallowing)

Note: The PhSync signal is generated centrally by the phase synchronizer, while the analog clock signals (e.g. Ana.Clock2 or Ana.Clock8) are generated locally within the respective module.

1) See "**Generation of the Analog Clock Signal**" for more details.

31.2.3 Generation of the Analog Clock Signal

The local clock generation units generate the respective analog clock signal triggered by the phase synchronization signal. The corresponding analog clock edge or trigger signal is generated upon a subsequent rising edge of the reference clock f_{PER}.

Note: The EVADC evaluates the level of the phase synchronizer signal to start a conversion.
The EDSADC evaluates the rising edge of the phase synchronizer signal to generate clock pulses.
For proper operation, only change the configuration of the phase synchronizer while the converters are idle.

Each unit can configure a certain phase shift for the synchronized analog clock or trigger. This way, all analog modules can switch on exactly the same clock edge, or they can switch in a certain sequence that can be configured by the user.

The figure below shows the effect of the phase shift.

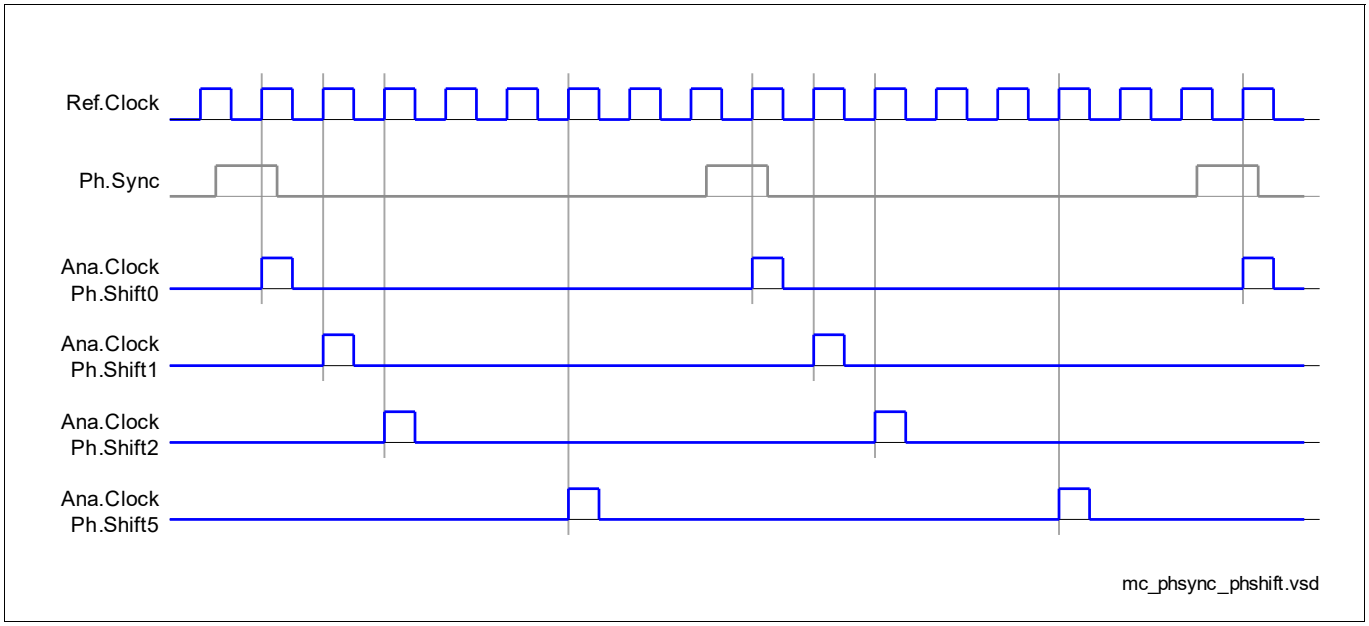


Figure 235 Analog Clock Phase Shift

If pulse swallowing is not feasible, the analog clock can also be generated with defined duty cycles. Still, the rising clock edges are triggered by the phase synchronizer signal.

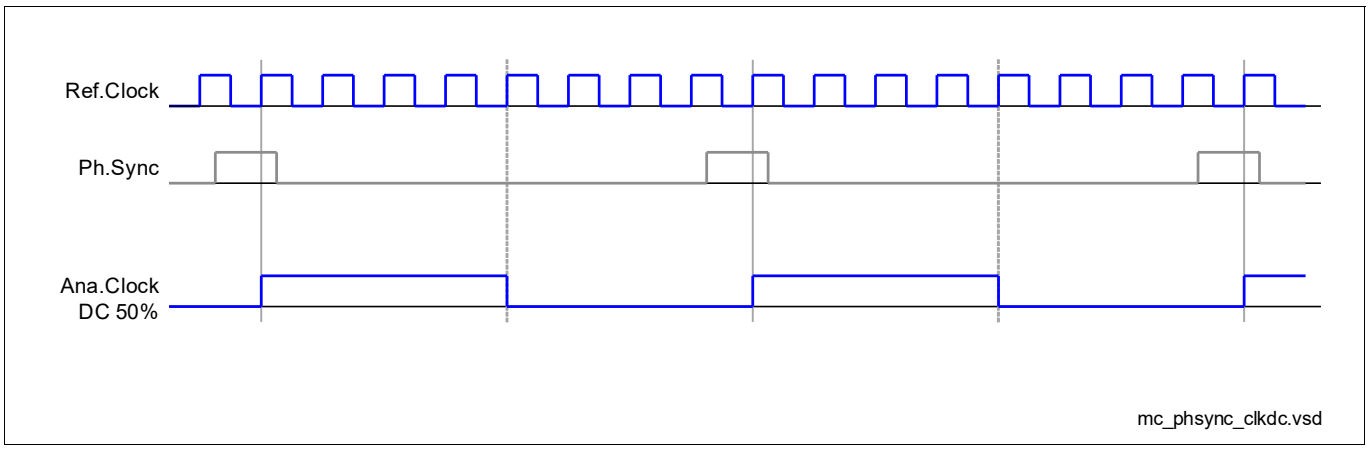


Figure 236 Analog Clock Generation without Pulse Swallowing

31.2.4 Safety Measures

Since the Phase Synchronizer influences all analog-digital-converters, safety measures are implemented to supervise its operation and, as a consequence, the operability of the converters.

Parity-Protection of the Prescaler Value

The additional parity bit helps to detect a corrupted prescaler value, which would result in the wrong synchronization frequency for the converters. The parity bit is generated automatically when writing to bitfield PHSDIV. Parity is then checked constantly.

A parity error caused by the corruption of a prescaler value bit triggers an alarm event.

Run-Time Supervision of the Counter

Two counters are counting in parallel. Their states are constantly compared and a difference triggers an alarm event.

Alarm-Event Signaling

If an alarm event is triggered it sets the sticky alarm flag and activates the alarm output signal.

The output signal is connected to the SMU.

The alarm flag can be cleared by writing a 1 to bit ALFCLR. This also deactivates the alarm output signal.

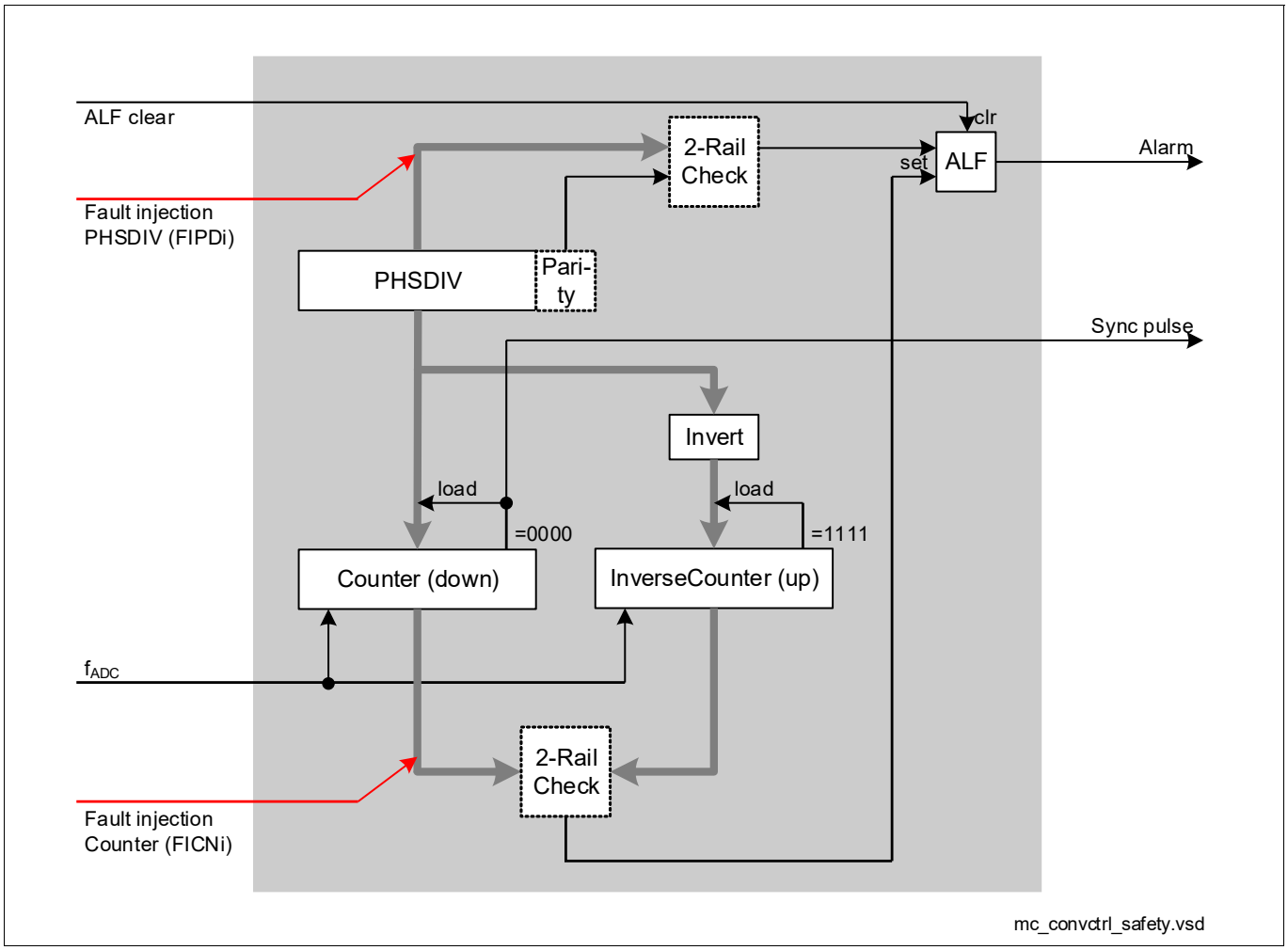


Figure 237 Phase Synchronizer Safety Features

Fault Injection

To test the supervision mechanism, the application software can deliberately inject a fault condition into both mechanisms. This is accomplished by writing a 0 to bit FIPD0/FICN0 and writing a 1 to bit FIPD1/FICN1 at the same time. Bits FIPDi trigger a fault condition in the supervision logic of bitfield PHSDIV, bits FICNi trigger a fault condition in the supervision logic of the phase sync counter.

The alarm flag can be cleared by writing a 1 to bit ALFCLR. This also deactivates the alarm output signal.

Phase Synchronizer Safety Control Register

PHSSFTY

Phase Synchronizer Safety Control Register (084$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										FICN1	FIPD1	0			ALFCLR
r										w	w	r			w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										FICN0	FIPD0	0			ALF
r										w	w	r			rh

Field	Bits	Type	Description
ALF	0	rh	**Alarm Flag for Safety Features** This sticky flag is set when a described error is detected (see "Safety Measures"), it is cleared by writing a "1" to bit ALFCLR. 0_B No error indicated 1_B Safety problem has been detected
FIPD0	4	w	**Fault Injection Phase sync Divider** 0_B Inject fault if FIPD1 is written with 1 1_B No action
FICN0	5	w	**Fault Injection Counter** 0_B Inject fault if FICN1 is written with 1 1_B No action
ALFCLR	16	w	**Alarm Flag ALF Clear** 0_B No action 1_B Clear alarm flag ALF
FIPD1	20	w	**Fault Injection Phase sync Divider** 0_B No action 1_B Inject fault if FIPD0 is written with 0
FICN1	21	w	**Fault Injection Phase sync Divider** 0_B No action 1_B Inject fault if FICN0 is written with 0

Field	Bits	Type	Description
0	3:1, 15:6, 19:17, 31:22	r	**Reserved, write 0, read as 0**

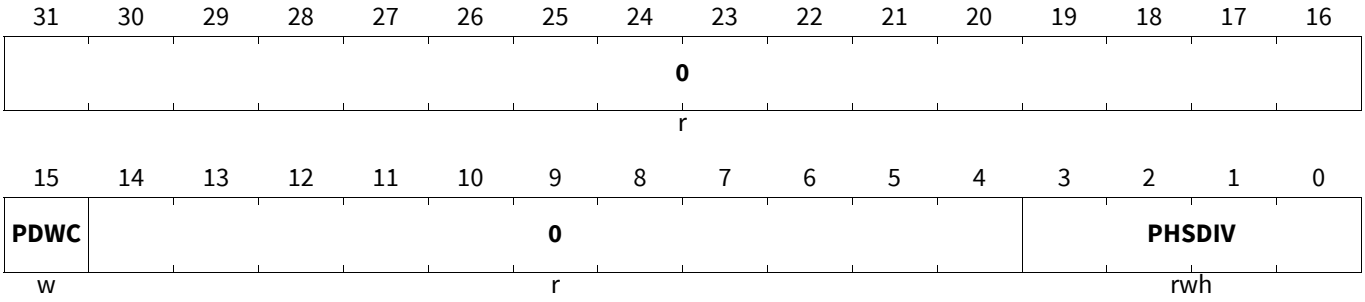
31.2.5 Configuration of the PhSync

The prescaling factor of the phase synchronization signal can be configured to adapt the functionality of the CONVCTRL to the requirements of the actual application.

Phase Synchronizer Configuration Register

PHSCFG

Phase Synchronizer Configuration Register (080$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
PHSDIV	3:0	rwh	**Phase Synchronizer Divider** Selects the prescaling factor between the peripheral clock and the phase synchronization signal. 0_H Off, the phase synchronization signal is constantly active 1_H The phase synchronization signal is generated at fADC / 2 … F_H The phase synchronization signal is generated at fADC / 16
PDWC	15	w	**Write Control for Phase Sync. Divider** 0_B No write access to divider factor 1_B Bitfield PHSDIV can be written
0	14:4, 31:16	r	**Reserved, write 0, read as 0**

Table 241 Access Mode Restrictions of PHSCFG sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **PDWC**	rwh	PHSDIV	Set PDWC during write access
(default)	rh	PHSDIV	

Note: Writing to bitfield PHSDIV automatically generates the corresponding parity bit (indicated by rw"h").

31.3 Application Considerations

The operation of the CONVCTRL and, hence, its behavior is programmable. This makes it suitable for different applications while requiring a minimum of handling.

31.3.1 Clock Synchronization

To eliminate the interference of concurrently operating ADC channels, the converters can operate in a synchronized way so each of them can reach its optimum performance. The Phase Synchronizer distributes a clock control signal which is used by the converters to generate a local clock signal.

After reset, the clock synchronization is active within the converters. The Phase Synchronizer, therefore, must be configured to generate the clock synchronization signal. This synchronization signal is used within the EDSADC and the EVADC. The following conditions must be fulfilled for proper operation of the converters:

- **EDSADC**:
  The EDSADC uses the synchronization signal directly to generate its modulator clock. Therefore, the phase synchronizer must be programmed according to the required EDSADC modulator frequency,
  i.e. $f_{PHSYNC} = f_{MOD}$.
  The modulator frequency must be configured accordingly within the EDSADC (DIVM).
- **EVADC**:
  The EVADC uses the synchronization signal to start a conversion. Therefore, the analog clock of the EDVADC f_{ADCI} must be selected as an integer multiple of the phase synchronizer frequency,
  i.e. $f_{ADCI} = f_{PHSYNC}$ or $f_{ADCI} = f_{PHSYNC} \times 2$.
  The analog clock frequency must be configured accordingly within the EVADC (DIVA).

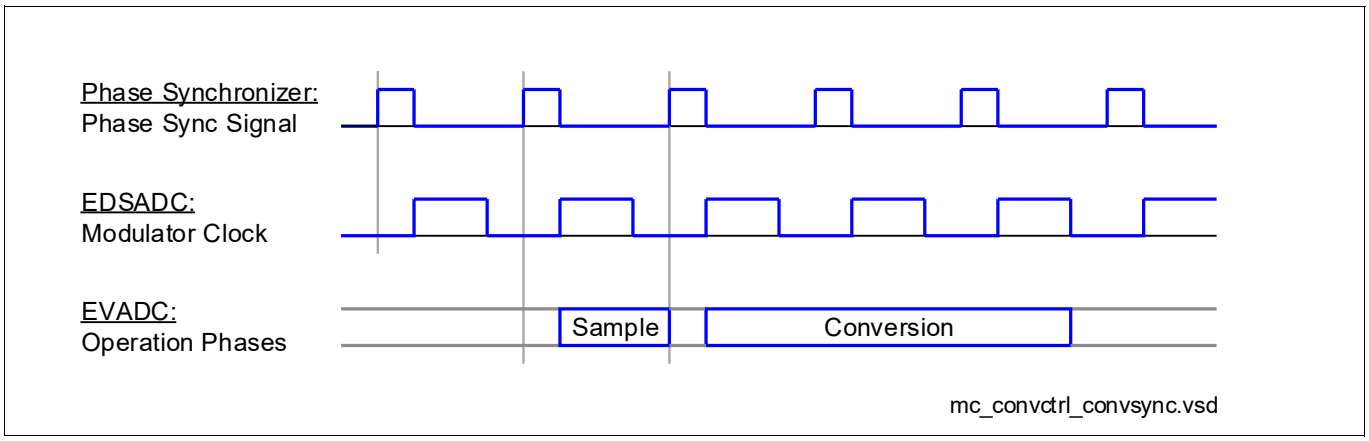


Figure 238 Inter-Module Synchronization

31.3.2 Example Settings for Operation

According to the rules given above the following settings can be derived:

Table 242 Synchronized Converter Clock Scenarios

Phase Synchronizer		EDSADC		EVADC	
Synchronization Frequency f_{PHSYNC}	**PHSDIV**	**Modulator Frequency f_{MOD}**	**DIVM**	**Analog Clock Frequency f_{ADCI}**	**DIVA**
16 MHz	1001_B	16 MHz	011_B	16 MHz	01001_B
				32 MHz	00100_B
20 MHz	0111_B	20 MHz	010_B	20 MHz	00111_B
				40 MHz	00011_B
26.67 MHz	0101_B	26.67 MHz	001_B	26.7 MHz	00101_B
				53.3 MHz	00010_B
40 MHz	0011_B	40 MHz	000_B	40 MHz	00011_B

31.3.3 Basic Initialization Sequence

After reset, the CONVCTRL is disabled to minimize the initial power consumption. By executing the following steps the CONVCTRL can be prepared for operation and be started:

```
CONVCTRL_CLC          = 0x00000000 ;Enable module CONVCTRL
CONVCTRL_CCCTRL       = 0xB0000000 ;Unlock converter control registers
CONVCTRL_PHSCFG       = 0x00008007 ;Synchronizer frequency = 160 MHz / 8 = 20 MHz
CONVCTRL_CCCTRL       = 0x00000000 ;Lock converter control registers
```

*Note: The synchronizer frequency is determined by the overall system setup (see **Table 242**).*

31.3.4 Module Handling in Sleep Mode

The CONVCTRL does not change its operating mode in sleep mode. While sleep mode is evaluated (CLC.EDIS = 0, default after reset), the module clocks are stopped upon a sleep mode request. To achieve the power reduction that is usually intended during sleep mode, the phase synchronizer is off during sleep mode, i.e. the phase synchronization signal is constantly active.

Note: If any activity of connected modules is intended during sleep mode make sure that sleep mode requests are disregarded (CLC.EDIS = 1).
If the CONVCTRL shall enter sleep mode, make sure also the connected modules EVADC and EDSADC are prepared for sleep mode.

31.4 Summary of Registers and Locations

The CONVCTRL is a very small module which is controlled by very few registers.

Registers with write access mode "...,M" are additionally protected from unintended write access by bitfield **CCCTRL**.TC. Set bitfield TC to $1011_B/B_H$ to enable write access to the module registers.

Table 243 Register Overview - CONVERTER (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	000_H	U,SV	SV,E,P	Application Reset	2
ID	Module Identification Register	008_H	U,SV	BE	PowerOn Reset	2
OCS	OCDS Control and Status Register	028_H	U,SV	SV,P,OEN	See page 3	3
KRSTCLR	Kernel Reset Status Clear Register	$02C_H$	U,SV	SV,E,P	Application Reset	7
KRST1	Kernel Reset Register 1	030_H	U,SV	SV,E,P	Application Reset	6
KRST0	Kernel Reset Register 0	034_H	U,SV	SV,E,P	Application Reset	6
ACCEN0	Access Enable Register 0	$03C_H$	U,SV	SV,SE	Application Reset	4
CCCTRL	Converter Control Block Control Register	$07C_H$	U,SV	SV,E,P	Application Reset	7
PHSCFG	Phase Synchronizer Configuration Register	080_H	U,SV	SV,P,M	Application Reset	16
PHSSFTY	Phase Synchronizer Safety Control Register	084_H	U,SV	SV,P,M	Application Reset	14

31.5 Revision History

This is a summary of the modifications that have been applied to this chapter.

Table 244 Revision History

Reference	Change to Previous Version	Comment
V3.0.0		
Page 5	Describe details of the module kernel reset	
Page 2	Add info about suspend state to register CLC	
Page 3	Add note about suspend mode to register OCS	
V3.0.1		
Page 2	Self-reference removed from register CLC.	
Page 16	Replace fPER with fADC in register PHSCFG.	

32 Enhanced Versatile Analog-to-Digital Converter (EVADC)

The EVADC provides a series of analog input channels connected to several clusters of Analog/Digital Converters using the Successive Approximation Register (SAR) principle to convert analog input values (voltages) to discrete digital values.

The EVADC is based on SAR converters, each comprising a sample&hold unit and a converter block. An analog multiplexer selects one of several input channels and a dedicated control logic with several request sources defines the sequence of consecutive conversions. This altogether builds a conversion group.
The number of analog input channels and conversion groups depends on the chosen product type. This is described in the product-specific appendix.

Each converter of the ADC clusters can operate independent of the others. The results of each channel can be stored in dedicated channel-specific result registers or in a group-specific result register.

Three clusters with different functionality are available: (section summary on **Page 3**)

- **Primary converter cluster**: Equipped with 8:1 multiplexers and 8-stage queues, conversion time down to below 0.5 µs.[1)]
- **Secondary converter cluster**: Equipped with 16:1 multiplexers and 16-stage queues, conversion time down to below 1 µs.[1)]
- **Fast compare cluster**: Single channels, update rate down to below 0.2 µs.[1)]

In addition to the EVADC clusters also the EDSADC performs analog-digital conversions.

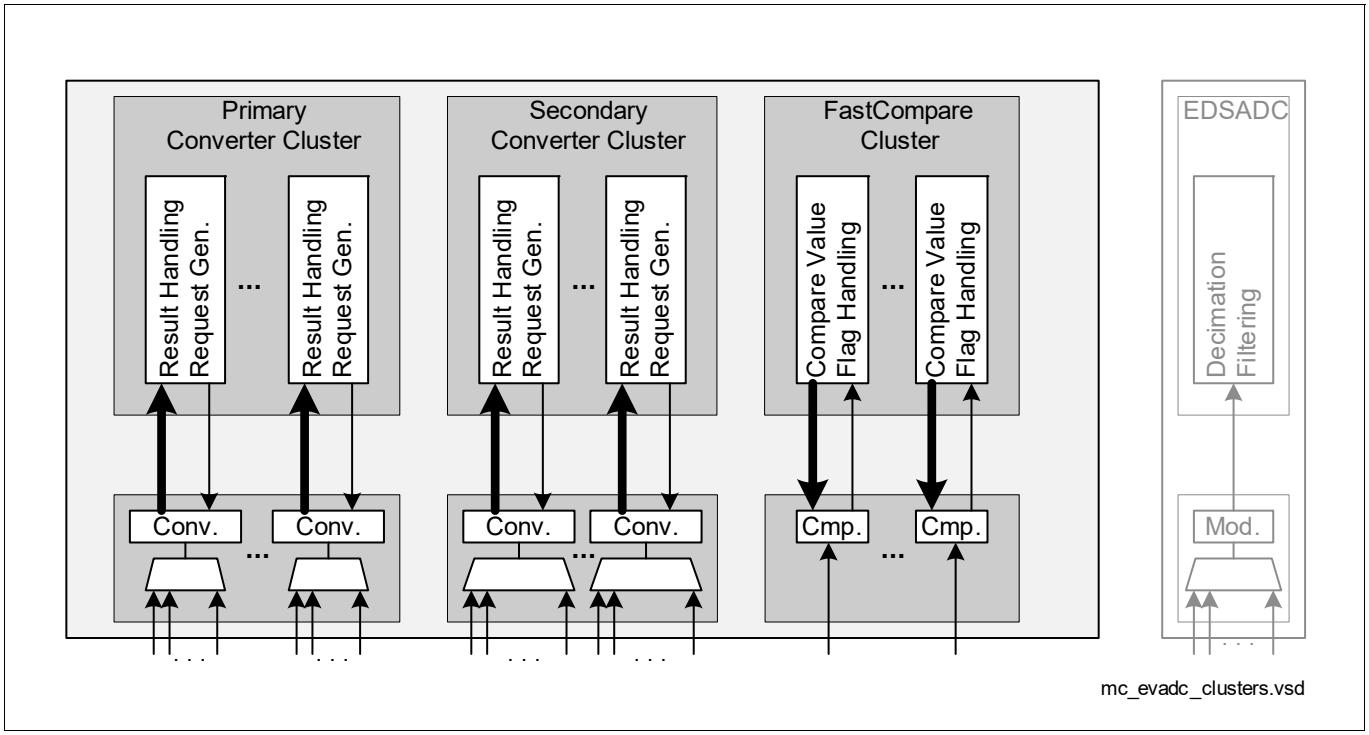


Figure 239 Summary of Clusters

1) Refer to **"Conversion Timing" on Page 86**.

32.1 Feature List

The following features describe the functionality of the ADC clusters:

- Nominal analog supply voltage 5.0 V or 3.3 V
- Input voltage range from 0 V up to analog supply voltage
- Standard (V_{AREF}) and alternate (CH0) reference voltage source selectable for each channel to support ratiometric measurements and different signal scales
- Several independent converters with up to 8/16 analog input channels each
- External analog multiplexer control, including adjusted sample time and scan support
- Conversion speed and sample time adjustable to adapt to sensors and reference
- Conversion time below 0.5 µs for primary channels (depending on sample time, refer to **Section 32.9**)
- Flexible source selection and arbitration
  - Programmable arbitrary conversion sequence (single or repeated)
  - Request source chaining to generate extended conversion sequences
  - Conversions triggered by software, timer events, or external events
  - Cancel-inject-restart mode for reduced conversion delay on priority channels
- Powerful result handling
  - Independent result registers
  - Configurable limit checking against programmable border values
  - Storage of maximum/minimum value
  - Data rate reduction through adding a selectable number of conversion results
  - FIR/IIR filter with selectable coefficients
- Fast Compare Channels with above 5 Msamples/s (refer to **Section 32.9**)
  - Automatic handling of flags and output signals
  - Compare value adjustable via software, via conversion channel or via ramp
- Flexible service request generation based on selectable events
- Built-in safety features
  - Configurable register access protection supports safety applications
  - Broken wire detection
  - Multiplexer test mode to verify signal path integrity
  - Automatic execution of test sequences
- Support of suspend and power saving modes

32.2 Overview

The Enhanced Versatile Analog to Digital Converter module (EVADC) comprises a set of converter blocks that can be operated either independent from each other, sequentially chained for longer conversion sequences, or synchronized for parallel conversion of up to 4 channels.

You will find the following major sections within this chapter:

Attention: This chapter describes the EVADC of the TC3XX family, encompassing the features and functions of all family members.
The specific characteristics of a product are described in the product-specific appendix, which specifies deviations (e.g. downgrades, etc.) from this family documentation.

Table 245 Abbreviations used in EVADC

Abbreviation	Meaning
ADC	Analog to Digital Converter
DMA	Direct Memory Access (controller)
DNL	Differential Non-Linearity (error)
EDSADC	Enhanced Delta-Sigma (conversion principle) Analog to Digital Converter
HDI	Hardware Data Interface
INL	Integral Non-Linearity (error)
LSB_n	Least Significant Bit: finest granularity of the analog value in digital format, represented by one least significant bit of the conversion result with n bits resolution (measurement range divided in 2^n equally distributed steps)
SCU	System Control Unit of the device
TUE	Total Unadjusted Error

The EVADC is connected to other on-chip modules to support functions on system level (see **Figure 242**).

Each converter block is equipped with a dedicated input multiplexer and dedicated request sources (except for fast compare channels), which together build separate groups, each assigned to a kernel (see **Figure 239**).

Primary groupsprovide 8:1 input multiplexers and deliver the minimum conversion time of approx. 0.5 µs (refer to **Section 32.9**)

Secondary groupsprovide 16:1 input multiplexers and require a higher sample time leading to an increased conversion time.

Fast compare channelsprovide one dedicated input channel each and deliver a compare time of approx. 200 ns.

This basic structure supports application-oriented programming and operating while still providing general access to all resources. The almost identical converter groups allow a flexible assignment of functions to channels.

A set of functional units (described further down in this section) can be configured according to the requirements of a given application. These units build a path from the input signals to the digital results.

Each kernel provides a dedicated Sample&Hold unit connected to the input multiplexer and to the converter itself.

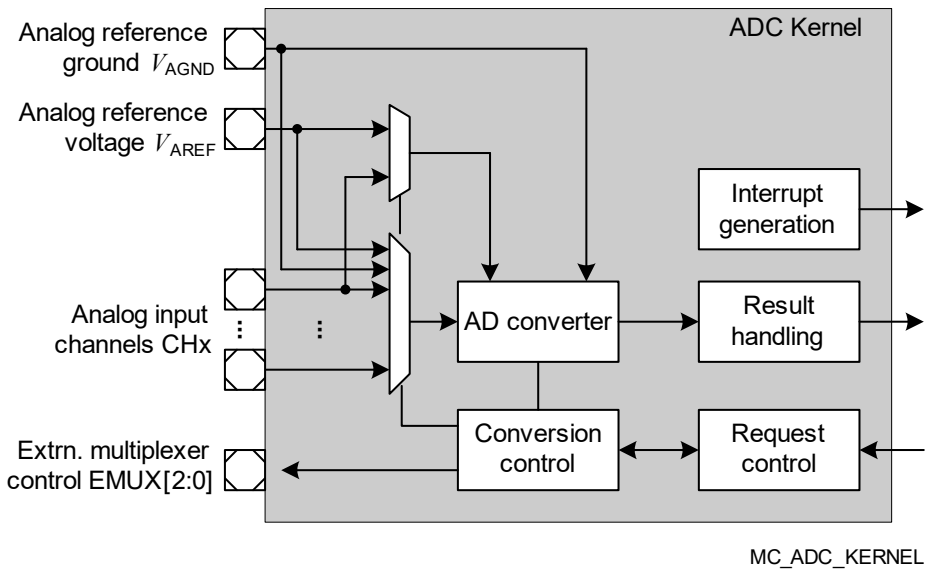


Figure 240 ADC Kernel Block Diagram

The basic module clock f_{ADC} is connected to the peripheral clock signal (see also **"General Clocking Scheme and Control" on Page 18**).

Conversion Modes and Request Sources

Analog/Digital conversions can be requested by three request sources and can be executed in several conversion modes. The request sources can be enabled concurrently with configurable priorities.

- **Fixed Channel Conversion (single or continuous)**A request source requests conversions of one selectable channel (once or repeatedly)
- **Channel Sequence Conversion (single or continuous)**A request source requests a sequence of conversions of up to 8/16 (primary/secondary groups) arbitrarily selectable channels (once or repeatedly)

Note: Extended sequences (more than 8/16 conversions) can be executed by linking several request sources together.

The conversion modes can be used concurrently by the available request sources, i.e. conversions in different modes can be enabled at the same time. Each source can be enabled separately and can be triggered by external events, such as edges of PWM or timer signals, or pin transitions.

Request Source Control

Because all request sources can be enabled at the same time, an arbiter resolves concurrent conversion requests from different sources. Each source can be triggered by external signals, by on-chip signals, or by software. The internal request source timer can request the individual conversions of a configured sequence with a programmable delay.

Requests with higher priority can either cancel a running lower-priority conversion (cancel-inject-repeat mode) or be converted immediately after the currently running conversion (wait-for-start mode). If the target result register has not been read, a conversion can be deferred (wait-for-read mode).

Certain channels can also be synchronized with other ADC kernels, so several signals can be converted in parallel.

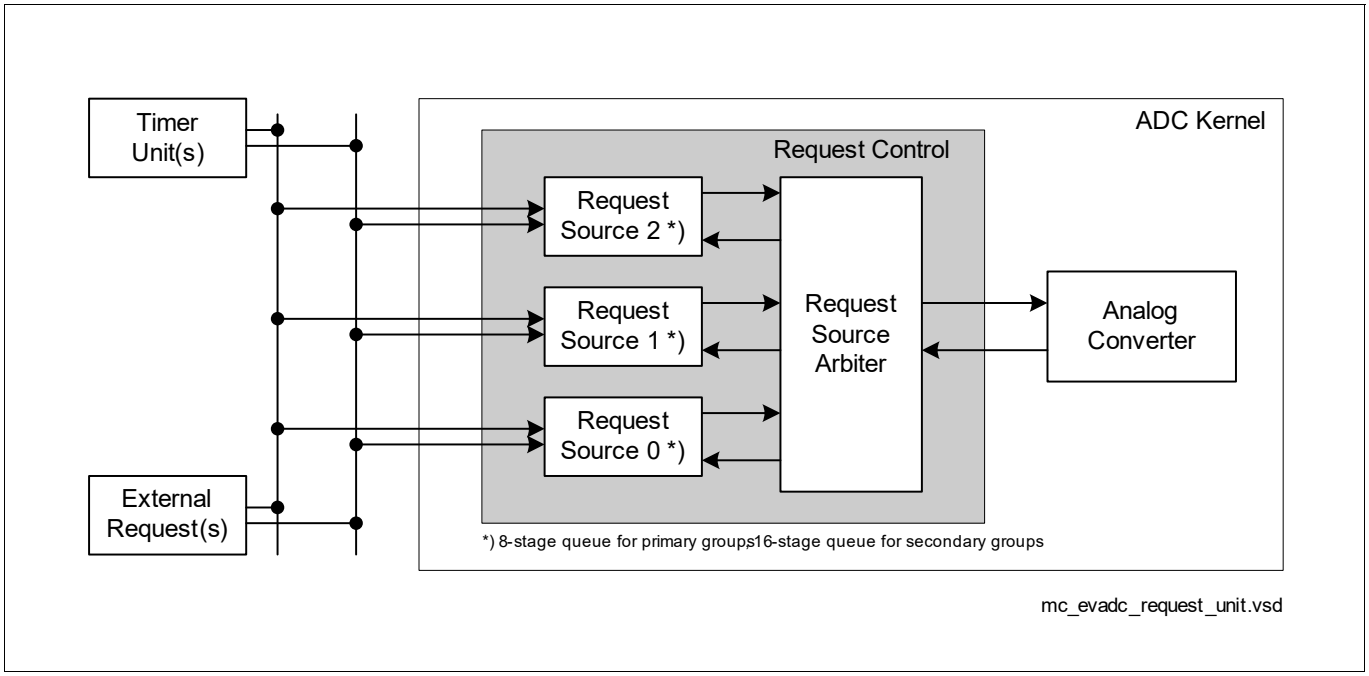


Figure 241 Conversion Request Unit

Input Channel Selection

The analog input multiplexer selects one of the available analog inputs (CH0 - CHx[1)]) to be converted. Three sources can select a linear sequence, an arbitrary sequence, or a specific channel. The priorities of these sources can be configured.

Additional external analog multiplexers can be controlled automatically, if more separate input channels are required than are built in (see **Section 32.13**).

Note: Not all analog input channels are necessarily available in all packages, due to pin limitations. Refer to the implementation description in the product-specific appendix.

Conversion Control

Conversion parameters, such as sample phase duration or conversion mode (noise reduction) can be configured for 4 input classes (2 group-specific classes, 2 global classes). Each channel can be individually assigned to one of these input classes.

Each channel can select either the standard reference voltage or the alternate reference voltage (restrictions see product-specific appendix).

The input channels can, thus, be adjusted to the type of sensor (or other analog sources) connected to the ADC.

This unit also controls the built-in multiplexer and external analog multiplexers, if selected.

Analog/Digital Converter

The selected input channel is converted to a digital value by first sampling the voltage on the selected input and then generating the result bits.

The sample&hold unit of a group is connected to an SAR converter. This converter generates the digital result values for the sampled signal.

Safety features (see **Section 32.12**) help to ensure the plausibility and correctness of the generated result values.

Result Handling

The conversion results of each analog input channel can be directed to one of 16 group-specific result registers and one global result register to be stored there. A result register can be used by a group of channels or by a single channel.

The wait-for-read mode avoids data loss due to result overwrite by blocking a conversion until the previous result has been read.

Data reduction (e.g. for digital anti-aliasing filtering) can automatically add up to 16 conversion results before issuing a service request.

Alternatively, an FIR or IIR filter can be enabled that preprocesses the conversion results before sending them to the result register (available for 2 registers).

Also, result registers can be concatenated to build FIFO structures that store a number of conversion results without overwriting previous data. This increases the allowed CPU latency for retrieving conversion data from the ADC.

The minimum value or maximum value of a sequence of conversions can be determined and stored automatically while the results are being generated.

1) The availablity of input channels depends on the package of the used product type. A summary can be found in the product-specific appendix.

Service Request Generation

Several ADC events can issue service requests to CPU or DMA:

- **Source events** indicate the completion of a conversion sequence in the corresponding request source. This event can be used to trigger the setup of a new sequence.
- **Channel events** indicate the completion of a conversion for a certain channel. This can be combined with limit checking, so interrupts are generated only if the result is within a defined range of values.
- **Result events** indicate the availability of new result data in the corresponding result register. If data reduction mode is active, events are generated only after a complete accumulation sequence.

Each event can be assigned to one of eight service request nodes. This allows grouping the requests according to the requirements of the application.

Safety Features

Safety-aware applications are supported with mechanisms that help to ensure the integrity of a signal path.

Broken-wire-detection (BWD) discharges the converter network before sampling the input channel. The result will then reflect a reduced value if the input signal is no more connected. If buffer capacitors are used, a certain number of conversions may be required to reach the failure indication level.

Pull Down Diagnostics (PDD) connects an additional strong pull-down device to an input channel. A subsequent conversion can then confirm the expected modified signal level. This allows to check the proper connection of a signal source (sensor) to the multiplexer.

Multiplexer Diagnostics (MD) connects a weak pull-up or pull-down device to an input channel. A subsequent conversion can then confirm the expected modified signal level. This allows to check the proper operation of the multiplexer.

Converter Diagnostics (CD) connects an alternate signal to the converter. A subsequent conversion can then confirm the proper operation of the converter.

Fast Compare Channels

Applications that do not require the exact conversion result but just an indication if the input signal is above/below a given threshold (e.g. peak-and-hold operation) are supported by dedicated fast compare channels. A comparison takes approx. 200 ns. The results can control output signal via boundary flags.

The compare reference value can be written by software, can be generated by a ramp generator, or can be provided by another analog input channel.

System Integration and Communication

Besides the connection to the peripheral bus, the EVADC is directly connected to other hardware modules of the AURIX™ TC3xx Platform. **Figure 242** shows an overview of these interconnections. More detailed information can be found within the family documentation of the EVADC as well as of the connected modules.

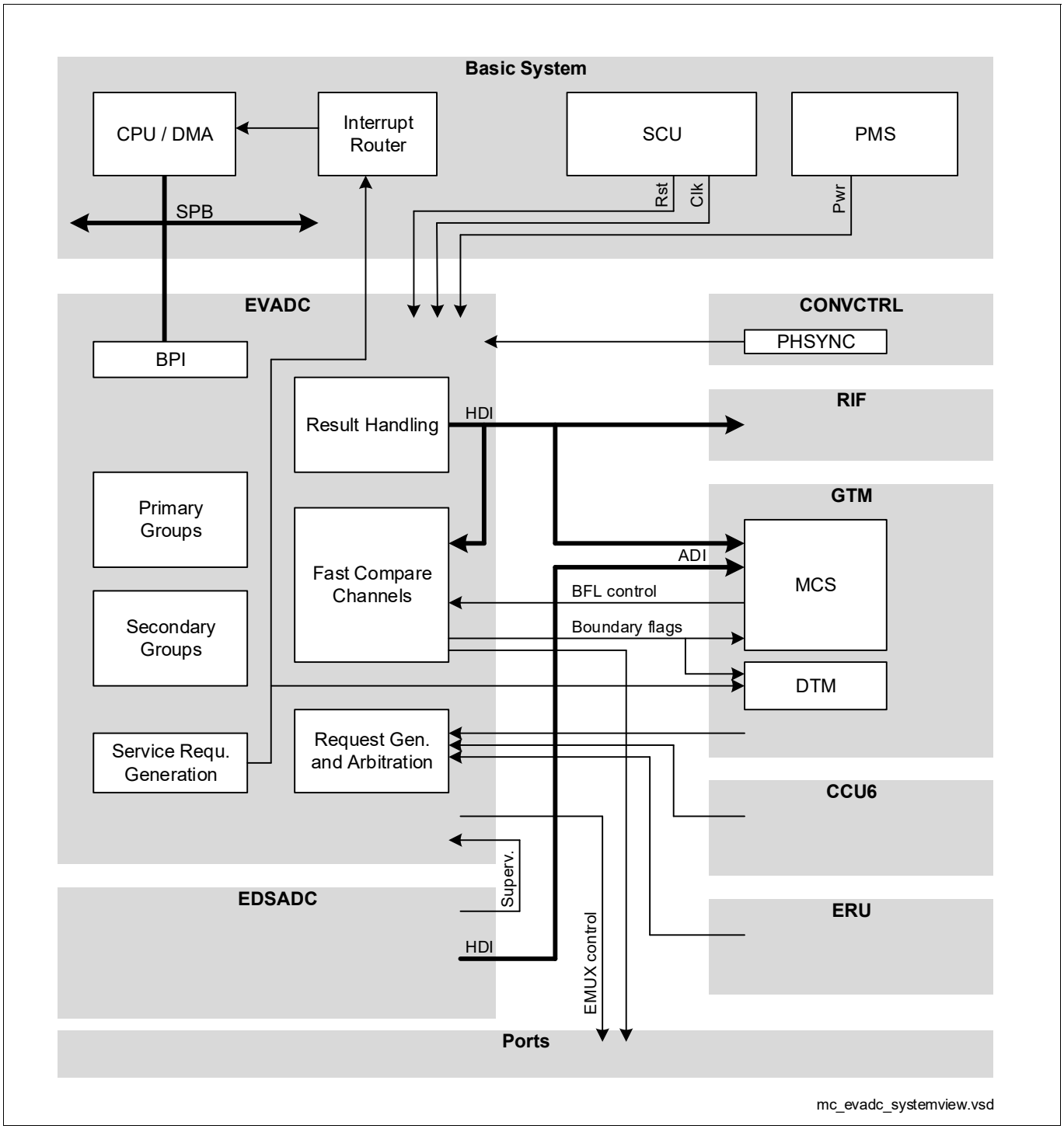


Figure 242 Direct System Interconnections

32.3 Configuration of General Functions

Several parameters can be configured individually for each channel, source, or group, or are valid for all ADC clusters. These parameters adapt the functionality of the EVADC to the requirements of the actual application (see **Section 32.3.5**).

32.3.1 Changing the Configuration

The configuration bitfields control the operation of the functional blocks of the EVADC. To ensure proper operation and interaction of these functional blocks, it is recommended to change the configuration parameters of a converter group only while this group's converter is inactive.
This means: ANONS = 00_B for primary/secondary groups, ANON = 0_B for fast compare channels.

After reset, this is the default state. When changing the configuration during run-time, be aware of the respective implications listed in the table below.

Table 246 Implications When Changing the Configuration

Configuration	Implication
Channel properties (GxCHCTRy, GxICLASSi, GLOBICLASSi, GxALIAS)	These properties are used for the next conversion to be started, so the setup may be inconsistent. Only change the configuration when no conversions for the respective channel are scheduled.
Service request node pointers (GxSEVNP, GxCEVNPi, GxREVNPi, GLOBEVNP)	The association of an event to a service request output may be unclear. Make sure no events are generated while switching pointers.
Boundary values (GxBOUND, GLOBBOUND)	The association of a limit band to a conversion result may be unclear. Only change limits while the corresponding result is not being generated.
Fast compare channels (FCzFCM, FCzFCHYST, FCzFCCTRL, FCzFCBFL except for BFS)	Since compare operations in most cases are triggered constantly, changing the configuration on the fly may lead to unintended intermediate states, e.g. wrong compare levels or missed output signals. It is recommended to stop the channel during reconfiguration.
Analog control (GxANCFG)	May corrupt the current conversion leading to wrong results. Only change while no conversions are running.
Basic configuration (GLOBCFG)	Changing basic configuration parameters during operation may lead to malfunctions or deadlocks. Only change the configuration while all groups are switched off.
Arbiter configuration (GxARBPR)	Changing priorities etc. during operation may lead to unintended conversion sequences. Only change the configuration of inactive request queues.
Request source control (GxQCTRLi, GxQMRi, GxREQTMi GxTRCTR except for COV)	Modifying source and mode of trigger signals may generate unintended trigger events. Only change the configuration while the request queue is inactive.
Result generation (GxRCRy, GLOBRCR)	The result generation properties control how new results are evaluated. A changed configuration may need some time to become effective. Only change the configuration of registers that are currently not targeted by the application.

Table 246 Implications When Changing the Configuration (cont'd)

Configuration	Implication
External multiplexer control (GxEMUXCTR, GxEMUXCS, EMUXSEL)	Modifications during operation may lead to inconsistent conversion sequences or intermediate transitions in the control of external multiplexers. Only change while the respective group is inactive. Exception: subchannel update in steady mode. Only change EMUXSEL while all groups are inactive.
Synchronous conversions GxSYNCTR	Changing the configuration of a synchronization group may lead to inconsistent settings within such a group. Only change the setup of a synchronization group while all involved converter groups are inactive.

32.3.2 Register Access Notes

Tricore atomic instructions (LDMST, ST.T, SWAP.W, SWAPMASK.W, CMPSWAP.W) only write back bits that are changing their level. This leads to the fact that bits that are already set cannot be written with a 1 when using RMW instructions.

No problem exists when using direct write instructions (e.g. ST.W).

This affects the bits in register GxVFR which are cleared by writing 1s.

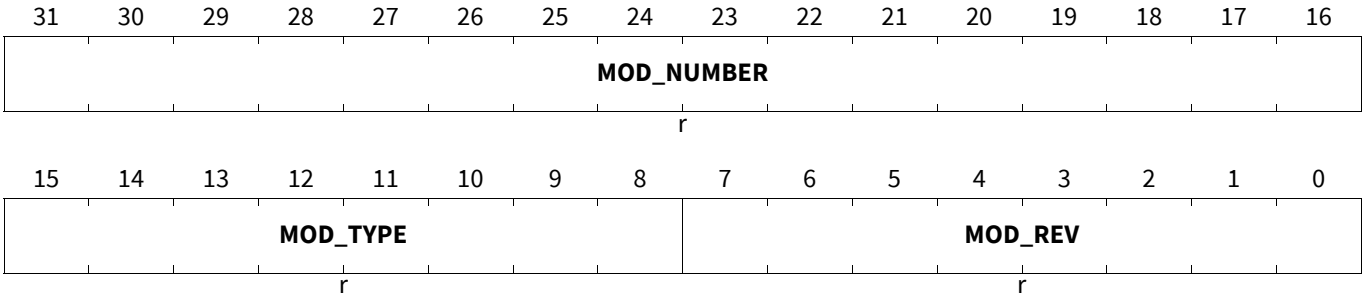
Also the events flags are affected (registers GxSEFLAG, GxCEFLAG, GxREFLAG, GLOBEFLAG) when trying to trigger an event via software while the corresponding bit is set.

32.3.3 Module Identification

The module identification register indicates the version of the EVADC module that is used in this product.

Module Identification Register

ID
Module Identification Register (0008$_H$) PowerOn Reset Value: 00C5 C013$_H$



Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision** Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	15:8	r	**Module Type** This internal marker is fixed to C0$_H$.
MOD_NUMBER	31:16	r	**Module Number** Indicates the module identification number (00C5$_H$ = SARADC).

32.3.4 System Registers

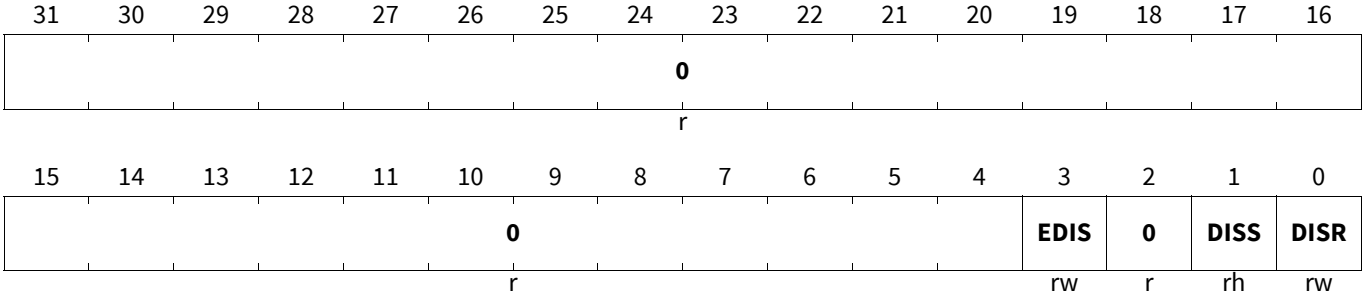
A set of standardized registers provides general access to the module and controls basic system functions.

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application.

Register CLC controls the module clock signal and the reactivity to the sleep signal.

CLC
Clock Control Register **(0000_H)** **Application Reset Value: 0000 0003_H**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. Also the analog section is disabled by clearing ANONS. 0_B On request: enable the module clock 1_B Off request: stop the module clock
DISS	1	rh	**Module Disable Status Bit** 0_B Module clock is enabled 1_B Off: module is not clocked f_{SPB} and f_{ADC} are disabled
EDIS	3	rw	**Sleep Mode Enable Control** Used to control the module's reaction to sleep mode. 0_B Sleep mode request is enabled and functional 1_B Module disregards the sleep mode control signal
0	2, 31:4	r	**Reserved, write 0**

It is recommended not to write to or read from module registers (except CLC) while the module is disabled. Write operations will generate a bus error.

When the module is disabled (DISR = 1_B) while in suspend state, the corresponding status bit (DISS = 1_B) will only be set after several clock cycles. To generate them, writes to CLC need to be repeated. See OCS note on **Page 14**.

To avoid unnecessary current consumption, it is recommended to disable the analog parts before disabling the module (GxARBCFG.ANONC = 00_B, GxANCFG.BE = 0_B or FCyFCM.ANON = 0_B).

OCDS Control and Status Register

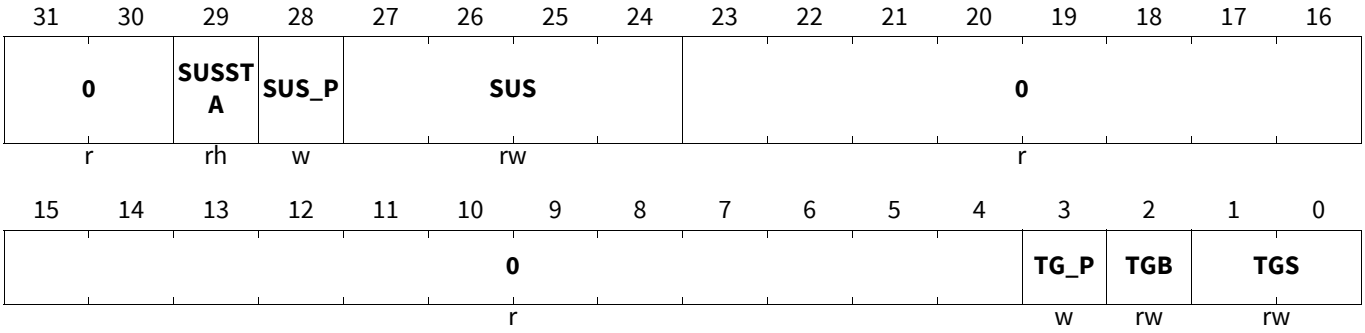
The OCDS Control and Status register OCS controls the module's behavior in suspend mode (used for debugging).

Register OCS is cleared by Debug Reset. It can only be written when OCDS is enabled.

If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective.

OCS

OCDS Control and Status Register **(0028_H)** **Reset Value: Table 248**



Field	Bits	Type	Description
TGS	1:0	rw	**Trigger Set for OTGB0/1** 00_B No Trigger Set output 01_B Trigger Set 1 TS16_SSIGC, input sample signals of primary/secondary groups 10_B Trigger Set 2 TS16_SSIGF, input sample signals of fast compare channels 11_B Reserved
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1
TG_P	3	w	**TGS, TGB Write Protection** TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) Not listed combinations are reserved. 0_H Will not suspend 1_H Hard suspend: Clocks f_{SPB} and f_{ADC} are switched off immediately. 2_H Soft suspend mode Stop conversions after the currently running one is completed and its result has been stored. The arbiter does not request a subsequent conversion.
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:4, 31:30	r	**Reserved, write 0, read as 0**

Table 247 Access Mode Restrictions of OCS sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **TG_P**	rw	TGB, TGS	Set TG_P during write access
write 1 to **SUS_P**	rw	SUS	Set SUS_P during write access
(default)	r	SUS, TGB, TGS	

Table 248 Reset Values of OCS

Reset Type	Reset Value	Note
PowerOn Reset	0000 0000_H	
Debug Reset	0000 0000_H	

Table 249 TS16_SSIGC Trigger Set EVADC

Bits	Name	Description
[11:0]	GxSAMPLE	Input signal sample phase of converter group x (x = 11-0)
[15:12]	0	Reserved

Table 250 TS16_SSIGF Trigger Set EVADC

Bits	Name	Description
[7:0]	FCxSAMPLE	Input signal sample phase of fast compare channel x (x = 7-0)
[15:8]	0	Reserved

*Note: During write access in hard suspend state the clocks are activated for a few cycles to accomplish the access. To prevent state machines from advancing due to this, it is recommended to stop the respective converter by clearing bitfield ANONC = 00_B in register **GxARBCFG (x=0-11)**.*

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions with the on-chip bus master TAG IDs 00 0000_B to 01 1111_B (see On-Chip Bus chapter for the mapping of product TAG ID <-> master peripheral). Register ACCEN0 provides one enable bit for each possible TAG ID encoding. Register ACCEN0 itself is protected by the Safety Endinit feature.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 00 0000_B, EN1 -> TAG ID 00 0001_B , ... , EN31 -> TAG ID 01 1111_B (TAG IDs 1X $XXXX_B$ are not used).

1) The Access Enable functionality controls only write transactions to registers CLC, OCS, KRSTx, and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

ACCEN0
Access Enable Register 0 **($003C_H$)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENx (x=0-31)	x	rw	**Access Enable for Master TAG ID x** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID x 0_B No Write access 1_B Write access will be executed

Individual Module Reset

The Kernel Reset Registers **KRST0**/**KRST1** are used to reset the related module kernel. Kernel Reset Registers 0 and 1 each include bit RST. To reset a module kernel it is necessary to set the RST bits in both Kernel Reset Registers. They will be cleared automatically with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set in the same clock cycle the RST bit is cleared. This bit indicates that a kernel reset was processed. Bit RSTSTAT can be cleared by setting bit CLR in register **KRSTCLR**.

A module kernel reset has the following effects:

Table 251 Effects of a Module Kernel Reset

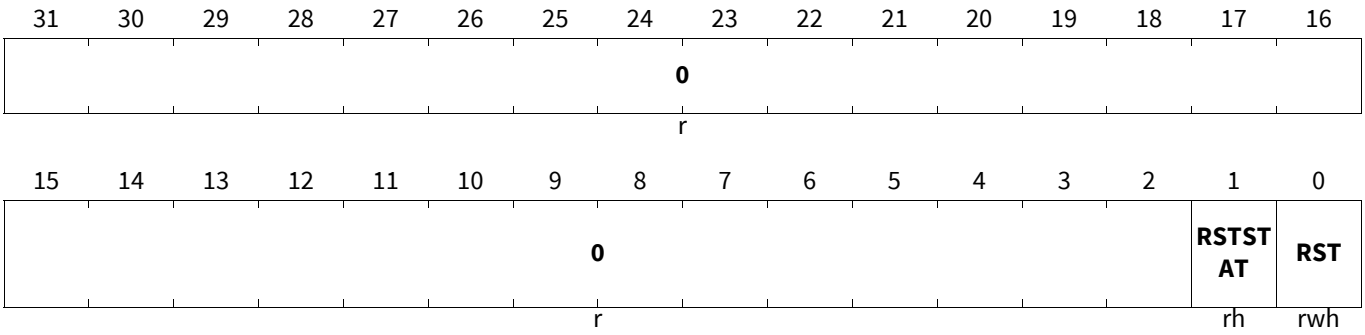
Register	Executed Action
CLC, OCS, ACCEN0	No influence
ID	No influence
KRST0	Bit RST is cleared automatically after reset execution, RSTSTAT indicates a module kernel reset, cleared via KRSTCLR
KRST1	Bit RST is cleared automatically after reset execution
KRSTCLR	No influence
Other registers	Reset to their defined reset values

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

Kernel Reset Register 0

KRST0

Kernel Reset Register 0 (0034_H) Application Reset Value: $0000\ 0000_H$

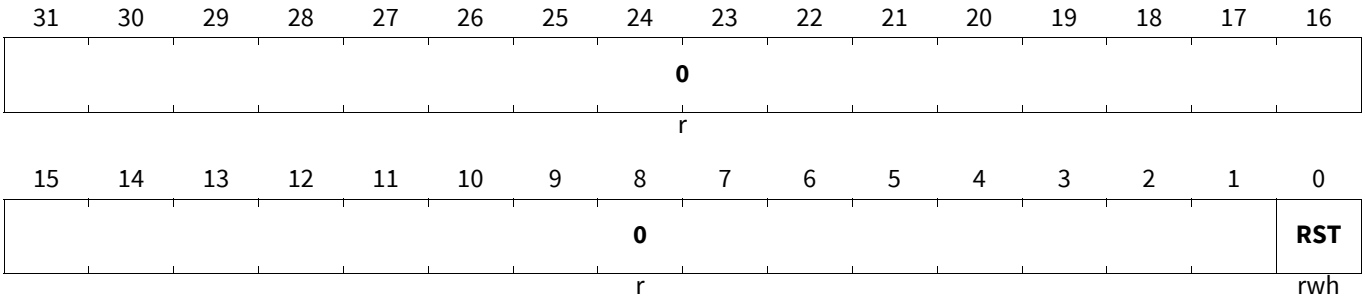


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** Request a kernel reset. The reset is executed if the reset bits of both kernel reset registers are set. RST is cleared after the kernel reset was executed. 0_B No action 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** Indicates an executed kernel reset. RSTSTAT is set after the execution of a kernel reset in the same clock cycle in which the reset bits are cleared. Clear RSTSTAT by setting bit CLR in register KRSTCLR. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved, write 0, read as 0**

Kernel Reset Register 1

KRST1

Kernel Reset Register 1 (0030_H) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** Request a kernel reset. The reset is executed if the reset bits of both kernel reset registers are set. RST is cleared after the kernel reset was executed. 0_B No action 1_B A kernel reset was requested
0	31:1	r	**Reserved, write 0, read as 0**

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **($002C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**CLR**
r															w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved, write 0, read as 0**

32.3.5 General Clocking Scheme and Control

The EVADC is operated with the peripheral clock signal (f_{ADC}). This clock signal controls all functions of all logic blocks and determines the overall timing. The converters are operated with the analog clock f_{ADCI} (see figure below). The analog clock is based on the analog phase synchronizer. The phase synchronizer defines the initial clock pulse for a conversion. This minimizes disturbances in the analog system. The started conversion is then executed without referring to the phase synchronizer. This ensures deterministic behavior of converters that shall operate in parallel.

*Note: The EVADC evaluates the level of the phase synchronizer signal. In synchronized mode, it will operate while the phase synchronizer signal is high. In unsynchronized mode (**GLOBCFG**.USC = 1), conversions are started independent of the phase synchronizer signal.*
For proper operation, only change the configuration of the phase synchronizer while the converters are idle.

The bus interface is clocked with the system clock f_{SPB}.

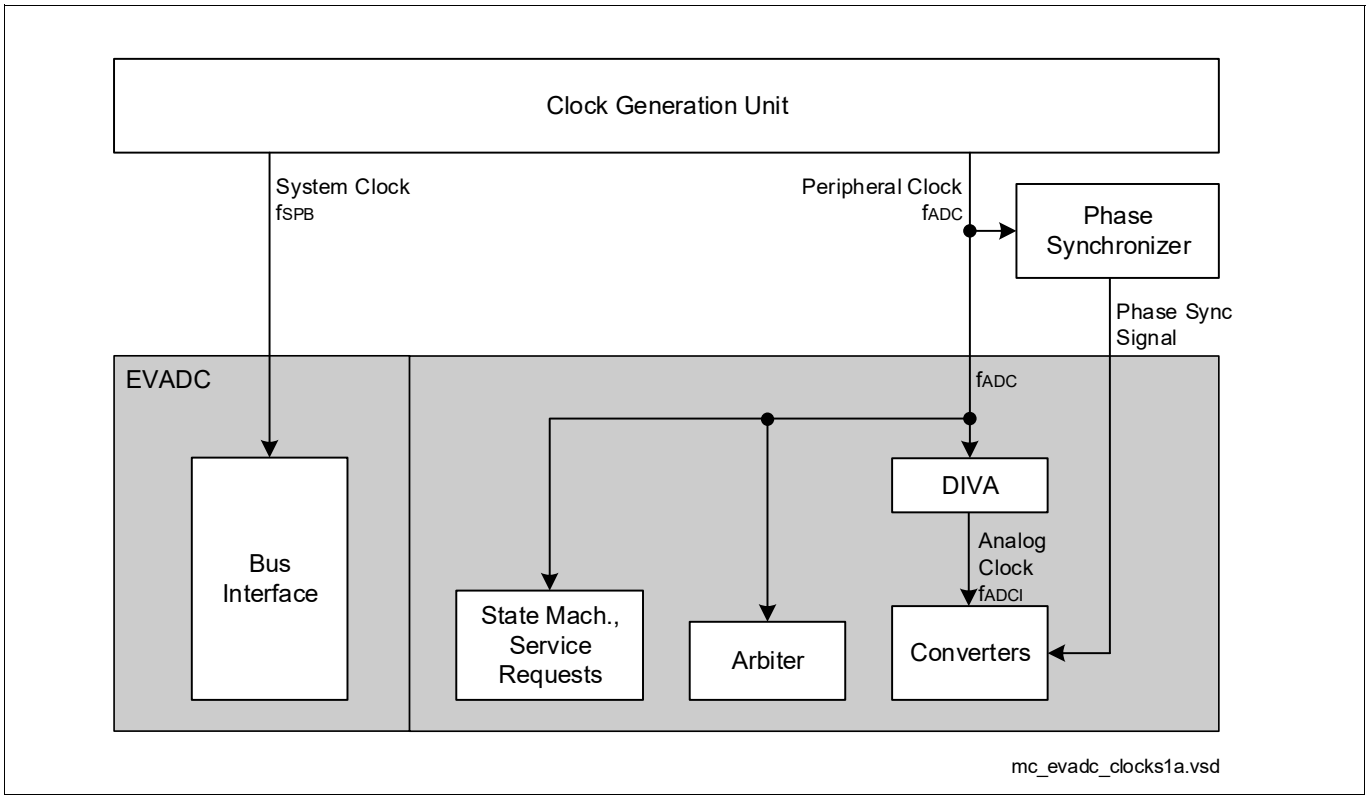


Figure 243 Clock Signal Summary

Attention: To ensure proper synchronization between the clock domains of the EVADC, the peripheral clock must never be slower than the bus clock, i.e. $f_{ADC} >= f_{SPB}$.

Global Configuration

The global configuration register GLOBCFG provides global control and configuration options that are valid for all converters.

Global Configuration Register

GLOBCFG
Global Configuration Register **(0080_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUCAL								0							
w								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPWC	SUPLEV		USC	0											
w	rw		rw	r											

Field	Bits	Type	Description
USC	12	rw	**Unsynchronized Clock Generation** Defines the way the analog clock is generated. 0_B Synchronized mode Initial clock pulse is defined by the phase synchronizer 1_B Unsynchronized mode The analog clock is generated independently.
SUPLEV	14:13	rw	**Supply Voltage Level** Adjusts the analog circuitry to the supply voltage used in the application system. Make sure to keep SUPLEV = 00_B or 01_B in the case of a 5 V supply. 00_B Automatic control: voltage range is controlled by the power supply 01_B Upper voltage range: assume a 5 V power supply is connected 10_B Lower voltage range: assume a 3.3 V power supply is connected 11_B Reserved
CPWC	15	w	**Write Control for Control Parameters** 0_B No write access to control parameters 1_B Bitfields SUPLEV, USC can be written
SUCAL	31	w	**Start-Up Calibration** Writing 1 to bit SUCAL initiates the start-up calibration phase of all enabled analog converters (except for the fast compare channels). *Note: Before and during start-up calibration, all converters must be inactive. After reset this is the case anyway.* 0_B No action 1_B Initiate the start-up calibration phase (indication in bit GxARBCFG.CAL)
0	11:0, 30:16	r	**Reserved, write 0, read as 0**

Table 252 Access Mode Restrictions of GLOBCFG sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **CPWC**	rw	SUPLEV, USC	Set CPWC during write access
(default)	r	SUPLEV, USC	

32.3.6 Register Access Control

Several protection schemes are provided to prevent unintended write access to control bitfields of the EVADC.

- The registers of the EVADC are protected by the general access control mechanism that is configured by register **ACCEN0**.
- A specific register access control scheme provides a versatile protection scheme against unintended corruption of register contents. Registers ACCPROT0/1/2 allow the restriction of write accesses for several groups of registers. The registers to be protected can be selected by the user. **Table 253** lists the registers that belong to each register group.
  Registers ACCPROT0/1/2 themselves are protected by the Safety Endinit feature.
- Groups of bitfields within a register may also be protected by an associated write control bit. This write control bit (xxWC) must be written with 1 along with the write access to the intended bitfield(s).

Access Protection Register 0

Controls write access to initialization registers and channel control registers.

The register below shows the maximum configuration. Other products of the family may have less groups and, consequently, less valid APxx control bits.

ACCPROT0
Access Protection Register 0 **(0088_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				APIS				APIP							
r				rw				rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				APCS				APCP							
r				rw				rw							

Field	Bits	Type	Description
APCP	7:0	rw	**Access Protection Channel Control, Primary Groups** Each bit of this bitfield is associated with the corresponding group. 0: Full access to registers 1: Write access to channel control registers is blocked
APCS	11:8	rw	**Access Protection Channel Control, Secondary Groups** Each bit of this bitfield is associated with the corresponding group. 0: Full access to registers 1: Write access to channel control registers is blocked
APIP	23:16	rw	**Access Protection Initialization, Primary Groups** Each bit of this bitfield is associated with the corresponding group. 0: Full access to registers 1: Write access to initialization registers is blocked
APIS	27:24	rw	**Access Protection Initialization, Secondary Groups** Each bit of this bitfield is associated with the corresponding group. 0: Full access to registers 1: Write access to initialization registers is blocked

Field	Bits	Type	Description
0	15:12, 31:28	r	**Reserved, write 0, read as 0**

Access Protection Register 1

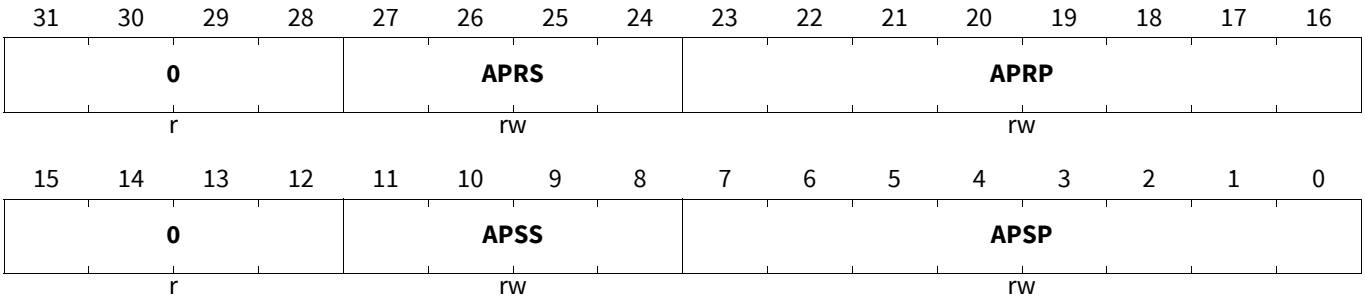
Controls write access to result registers and service request control registers.

Each bit of bitfield APR or APS controls the associated converter group.

The number of control bits in APR or APS, therefore, may be different in other products of the family.

ACCPROT1

Access Protection Register 1 **($008C_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
APSP	7:0	rw	**Access Protection Service Request, Primary Groups** Each bit of this bitfield is associated with the corresponding group. 0: Full access to registers 1: Write access to service request registers is blocked
APSS	11:8	rw	**Access Protection Service Request, Secondary Groups** Each bit of this bitfield is associated with the corresponding group. 0: Full access to registers 1: Write access to service request registers is blocked
APRP	23:16	rw	**Access Protection Result Registers, Primary Groups** Each bit of this bitfield is associated with the corresponding group. 0: Full access to registers 1: Write access to result registers is blocked
APRS	27:24	rw	**Access Protection Result Registers, Secondary Groups** Each bit of this bitfield is associated with the corresponding group. 0: Full access to registers 1: Write access to result registers is blocked
0	15:12, 31:28	r	**Reserved, write 0, read as 0**

Access Protection Register 2

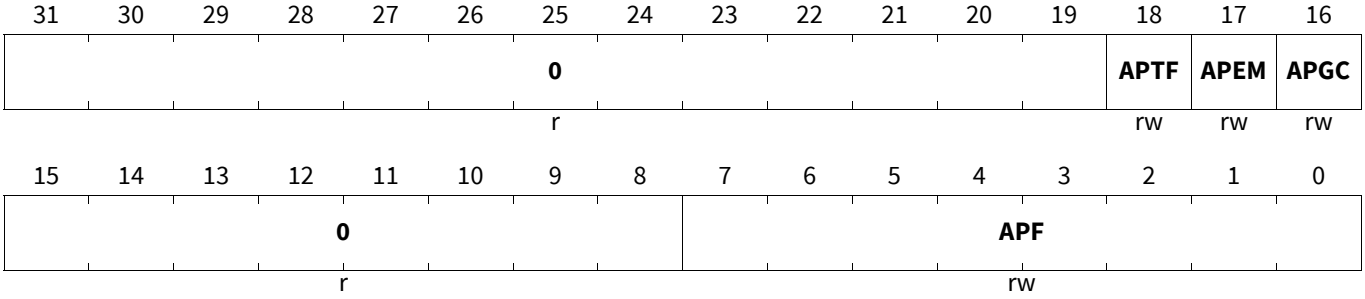
Controls write access to fast compare channel registers and global control registers.

Each bit of bitfield APF controls the associated fast compare channel.

The number of control bits in APF, therefore, may be different in other products of the family.

ACCPROT2

Access Protection Register 2 **(0090_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
APF	7:0	rw	**Access Protection Fast Compare Channels** Each bit of this bitfield is associated with the corresponding channel. 0: Full access to registers 1: Write access to fast compare channel registers is blocked
APGC	16	rw	**Access Protection Global Configuration** 0_B Full access to register 1_B Write access to global configuration register is blocked
APEM	17	rw	**Access Protection External Multiplexer** 0_B Full access to registers 1_B Write access to external multiplexer registers is blocked
APTF	18	rw	**Access Protection Test Function** 0_B Full access to register 1_B Write access to test function register is blocked
0	15:8, 31:19	r	**Reserved, write 0, read as 0**

Table 253 Register Protection Groups

Control Bits	Registers	Notes
APCx	GxCHCTR0 … GxCHCTRy	Channel control
APIx	GxARBCFG, GxARBPR, GxICLASS0/1, GxSYNCTR	Initialization
APSx	GxSEFLAG, GxSEVNP, GxCEFLAG, GxCEVNP0/1, GxREFLAG, GxREVNP0/1, GxSRACT	Service request control
APRx	GxRCR0 … GxRCR15, GxBOUND, GxRES0 … GxRES15	Result control
APFy	FCyFCBFL, FCyFCHYST, FCyFCCTRL, FCyFCM, FCyFCRAMP0/1	Fast compare channels
APGC	GLOBCFG	Gobal configuration
APEM	EMUXSEL, GxEMUXCTR, GxEMUXCS	External multiplexer control
APTF	GLOBTF, GLOBTE	Test functions

32.4 Analog Module Activation and Control

The analog converter of the EVADC is the functional block that generates the digital result values from the selected input voltage. It draws a permanent current during its operation and can be deactivated between conversions to reduce the consumed overall energy.

Note: After reset the analog converters are off. They must be enabled before triggering any action involving a converter.

The accuracy of the conversions is established by executing the start-up calibration (**"Calibration" on Page 25**).

32.4.1 Analog Converter Control

If an application does not require permanent activity of a converter, it can disable this converter intermittently to save energy. This can be done under software control or automatically.

The operating mode is determined by bitfield **GxARBCFG (x=0-11)**.ANONS:

- ANONS = 11_B: **Normal Operation**The converter is active, conversions are started immediately.
  Requires no wakeup time after initial settling (see note below).
- ANONS = 10_B: **Fast Standby mode**The converter enters a power reduction mode while no activity is required. It automatically returns to operation mode while conversions are requested. Fast standby mode reduces the overall power consumption for the ADC supply.
  Requires the standard wakeup time (see below).
- ANONS = 01_B: **Slow Standby mode**The converter enters a power save mode while no activity is required. It automatically returns to operation mode while conversions are requested. Slow standby mode enables the lowest overall power consumption for the ADC supply.
  Requires the extended wakeup time (see below).
- ANONS = 00_B: **Converter switched Off** (default after reset)
  The converter itself is switched off. Furthermore, digital logic blocks are set to their initial state.
  Before starting a conversion, select the active mode for ANONS.
  Requires the extended wakeup time (see below).

Wakeup Time from Analog Powerdown

When the converter is activated, it needs a certain wakeup time (depending on the operating mode) to settle before a conversion can be properly executed. This wakeup time can be established by adding it to the intended sample time.

The standard wakeup time is approximately 1 µs,
the extended wakeup time is approximately 5 µs.
Exact numbers can be found in the respective Data Sheets.

Note: The extended wakeup time is also required after initially enabling the converter.

32.4.2 Analog Signal Buffering

During operation, the converter capacitance must be charged and discharged several times. During the sampling phase it is charged from the selected signal input, during the conversion phase it is charged from the selected reference.

Both phases are supported by unloading the respective analog input by drawing a part of the required charge from the supply voltage.

Buffer for the Analog Input

The analog input buffer boosts the selected analog input signal for a certain time, when enabled. The time during which the input buffer is active can be adapted to the configured sample time by bitfields AIPS/AIPE in register **GxICLASSi (i=0-1;x=0-11)** / **GLOBICLASSi (i=0-1)**, or by bitfield AIPF in register **FCxFCCTRL (x=0-7)**. The input precharge time can be configured to 8, 16, or 32 clocks of f_{ADC}.

After the programmed buffer time the sampling is continued directly from the selected input. The remaining direct sampling time must cover the specified minimum sample time (Data Sheet), i.e. the programmed sample time must cover both phases, buffered sampling and direct sampling. Sample times specified in the data sheet consider a buffered sample time of 200 ns that means the input precharge time has to be configured to 32 clocks of f_{ADC}. For input precharge times lower than 200 ns, the charge consumption from the analog input is increased accordingly.

When the analog input buffer is activated (BE = 1 in register **GxANCFG (x=0-11)**), it needs a certain setup time to settle before a conversion can be properly executed.

The input buffer setup time is approximately 1 µs.
Exact numbers can be found in the respective Data Sheets.

Precharge for the Reference Input

The precharge feature charges the capacitor during conversion for a certain time, when enabled. After that, the capacitor is connected to the selected reference voltage to achieve maximum precision. This reduces the charge that is drawn from the reference (values see Data Sheet).

Reference precharging is controlled by bit RPE = 1 in register **GxANCFG (x=0-11)** or **FCxFCCTRL (x=0-7)**. Clearing bit RPE disables reference precharging. After reset, precharging is enabled.

32.4.3 Alternate Reference Selection

The EVADC control features allow selecting an alternate reference for each channel. This reference signal is taken from channel 0 of a group. The reference source is selected by bit REFSEL in the channel control registers and, therefore, can be configured for each input separately.

32.4.4 Calibration

Calibration automatically compensates deviations caused by process variation and aging. This ensures precise results throughout the operation time.

An initial start-up calibration is required once after a reset for all enabled converters (except for fast compare channels). All converters must be enabled (ANONS = 11_B). The start-up calibration is initiated globally by writing 1 to bit SUCAL in register GLOBCFG.

The duration of the start-up calibration phase depends on the programming, refer to **“Conversion Timing” on Page 86**.

Attention: During the start-up calibration, all converters must be inactive. After reset this is the case anyway. Only start conversions after the calibration has completed (indicated by bit CAL = 0).

After that, calibration cycles can compensate the effects of drifting external parameters, if not disabled. The converter itself requires no further calibration.

Postcalibration steps can be appended to each conversion. Since variation caused by aging is very slow and postcalibration leaves the capacitor discharged, for the current version of the converter postcalibration shall be disabled (most applications are restarted periodically and, therefore, also repeat the start-up calibration).

This also provides the highest conversion rate (refer to **Table 259 “Conversion Timing Overview” on Page 88**).

32.4.5 Noise Reduction Methods

Analog input signals almost always contain a certain amount of noise. This may be generated within the sensor itself or may be added along the signal path to the converter input.

Because the analog-digital converters are integrated into a microcontroller, also some noise generated by the digital blocks of the microcontroller may influence the operation of the converters.

The EVADC provides several means to attenuate the noise components that influence its operation and performance.

Noise-Reduction Conversions

Conversions can be extended by a selectable number of additional steps (1, 3, 7) to refine the generated result value. The noise reduction level is configured by bitfields CMS/CME in register **GxICLASSi (i=0-1;x=0-11)** etc. The influence on the conversion timing is described in **“Standard Converter Channels Timing” on Page 87**.

Spread Early Sample Point

Other than statistical noise, synchronous noise cannot be eliminated by oversampling. The spread early sample point feature attenuates synchronous noise by moving the end of the sample phase in pseudo random steps. The spread early sample point feature is enabled by bits SESPS/SESPE in register **GxICLASSi (i=0-1;x=0-11)** etc. When this feature is enabled, the sample phase is randomly shortened by up to 100 ns. This must be respected when configuring the duration of the sample time.

Accumulated Conversions

Accumulation adds 2 … 16 conversion results to support the calculation of average values. The predefined number of values is automatically accumulated and service requests are generated only after a completed accumulation.

More details are described in **“Data Modification” on Page 101**.

Mutual Interference of Converters

Since all converters use the common reference voltage V_{AREF} (except for the alternate reference), the converters can disturb each other. This can be avoided by synchronizing their operation. The phase synchronizer does this while bit USC = 0 in register **GLOBCFG** (default after reset). In special cases a dedicated phase shift can be selected via bitfield ACSD in register **GxANCFG (x=0-11)**.

Refer also to **“Synchronous Sampling” on Page 113**.

32.4.6 Analog Module Functions

Certain operating modes that control the analog behavior are configurable for each converter separately.

Analog Fct. Config. Register, Group x

GxANCFG (x=0-11)

Analog Fct. Config. Register, Group x (0488$_H$+x*400$_H$) Application Reset Value: 0030 0004$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						DCMSB	DIVA					SSE	ACSD		
r						rw	rw					rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									DPCAL	CALSTC		RPC	RPE	BE	IPE
r									rw	rw		rw	rw	rw	rw

Field	Bits	Type	Description
IPE	0	rw	**Idle Precharge Enable** 0_B No precharge, the sampling capacitor keeps the current charge 1_B The sampling capacitor is precharged to approx. half the reference when idle
BE	1	rw	**Input Buffer Enable** 0_B Input buffer off, input buffering is not possible Make sure AIPS/AIPE = 00_B. 1_B Input buffer enabled, select buffering time via bitfields AIPS/AIPE in register GxICLASS0 etc.
RPE	2	rw	**Reference Precharge Enable** *Note: Enabled after reset.* 0_B No reference precharge Only use V_{AREF}/V_{AGND} for the conversion. 1_B Precharge enabled Use V_{DDM}/V_{SSM} for precharging and V_{AREF}/V_{AGND} for the final adjustment during a conversion.
RPC	3	rw	**Reference Precharge Control** 0_B Precharge the reference input for 1 clock phase 1_B Precharge the reference input for 1 clock period (2 phases)
CALSTC	5:4	rw	**Calibration Sample Time Control** 00_B $2 \times t_{ADCI}$ 01_B $4 \times t_{ADCI}$ 10_B $6 \times t_{ADCI}$ 11_B $8 \times t_{ADCI}$
DPCAL	6	rw	**Disable Post-Calibration** 0_B Automatic post-calibration after each conversion of group x 1_B No post-calibration

Field	Bits	Type	Description
ACSD	18:16	rw	**Analog Clock Synchronization Delay** Defines the delay of the analog clock in clocks after the sync signal. *Note: Do not exceed the current DIVA setting for the clock delay. Valid only if the phase synchronizer is selected (USC = 0_B)* 000_B 0, no delay 001_B 1 clock cycle delay 010_B 2 clock cycles delay … 111_B 7 clock cycles delay
SSE	19	rw	**Sample Synchronization Enable** *Note: See section **"Synchronous Sampling" on Page 113**.* 0_B No synchronization 1_B Sample timing is synchronized Recommended for operation of several ADC groups.
DIVA	24:20	rw	**Divider Factor for the Analog Internal Clock** Defines the frequency of the analog converter clock f_{ADCI} (base clock for conversion steps), derived from the peripheral clock: $f_{ADCI} = f_{ADC}$ / CP. 00_H CP = 2 (max. frequency) 01_H CP = 2 02_H CP = 3 … $1F_H$ CP = 32
DCMSB	25	rw	**Double Clock for the MSB Conversion** Selects an additional clock cycle for the conversion step of the MSB. 0_B 1 clock cycle for the MSB (standard) 1_B 2 clock cycles for the MSB (test only)[1)]
0	15:7, 31:26	r	**Reserved, write 0, read as 0**

1) Not used for standard applications. Keep DCMSB = 0.

32.5 Conversion Request Generation

The conversion request unit of a group autonomously handles the generation of conversion requests. Three request sources can generate requests for the conversion of an analog channel. The arbiter resolves concurrent requests and selects the channel to be converted next.

Upon a trigger event, the request source requests the conversion of a certain analog input channel or a sequence of channels.

- **Software triggers**directly activate the respective request source.
- **Self-timed triggers**are generated by the request timer of the respective source.
- **External triggers**synchronize the request source activation with external events, such as a trigger pulse from a timer generating a PWM signal or from a port pin.

Application software selects the trigger type and source, the channel(s) to be converted, and the request source priority. A request source can also be activated directly by software without requiring an external trigger.

The arbiter regularly scans the request sources for pending conversion requests and selects the conversion request with the highest priority. This conversion request is then forwarded to the converter to start the sampling and conversion of the requested channel.

Each request source can operate in single-shot or in continuous mode:

- **In single-shot mode**,
  the programmed conversion (sequence) is requested once after being triggered. A subsequent conversion (sequence) must be triggered again.
- **In continuous mode**,
  the programmed conversion (sequence) is automatically requested repeatedly after being triggered once.

For each request source of a group, external triggers are generated from one of 15 selectable trigger inputs (REQTRx[O:A]) and from one of 16 selectable gating inputs (REQGTx[P:A])[1]. The available trigger signals are listed in the product-specific appendix.

Note: ***Figure 241 "Conversion Request Unit" on Page 5*** *summarizes the request sources.*

Properties of the Queued Request Source

A queued request source provides several buffer stages building a queue and can handle application-specific arbitrary conversion sequences up to the queue size.

Primary groups are equipped with 8-stage queues, secondary channels are equipped with 16-stage queues. Each queue can, therefore, handle all channels associated with the corresponding group.

The channel numbers for this sequence can be freely programmed[2]. Also, multiple conversions of the same channel within a sequence are supported. The programmed sequence is stored in a queue buffer (based on a FIFO mechanism). The requested channel numbers are entered via the queue input, while queue stage 0 defines the channel to be converted next. Each entry can be executed once or can be automatically reloaded to form continuously repeated sequences.

A conversion request is only issued to the request source arbiter if a valid entry is stored in queue stage 0.

If the arbiter aborts a conversion triggered by a queued request source due to higher priority requests, the corresponding conversion parameters are automatically saved in the backup stage. This ensures that an aborted conversion is not lost but takes part in the next arbitration step (before stage 0).

1) The selected gating input can be used as a trigger signal (REQTRxP).

2) The availablity of input channels depends on the package of the used product type. A summary can be found in the product-specific appendix.

Note: Conversion requests that are saved in the backup stage are executed as soon as their priority allows, even if the original conversion request was waiting for a trigger event.

The trigger and gating unit generates trigger events from the selected internal or external (outside the ADC) trigger and gating signals. For example, a timer unit can issue a request signal to synchronize conversions to PWM events.

Trigger events start a queued sequence and can be generated either via software or via the selected hardware triggers. The occurrence of a trigger event is indicated by bit QSRx.EV. This flag is cleared when the corresponding conversion is started or by writing to bit QMRx.CEV.

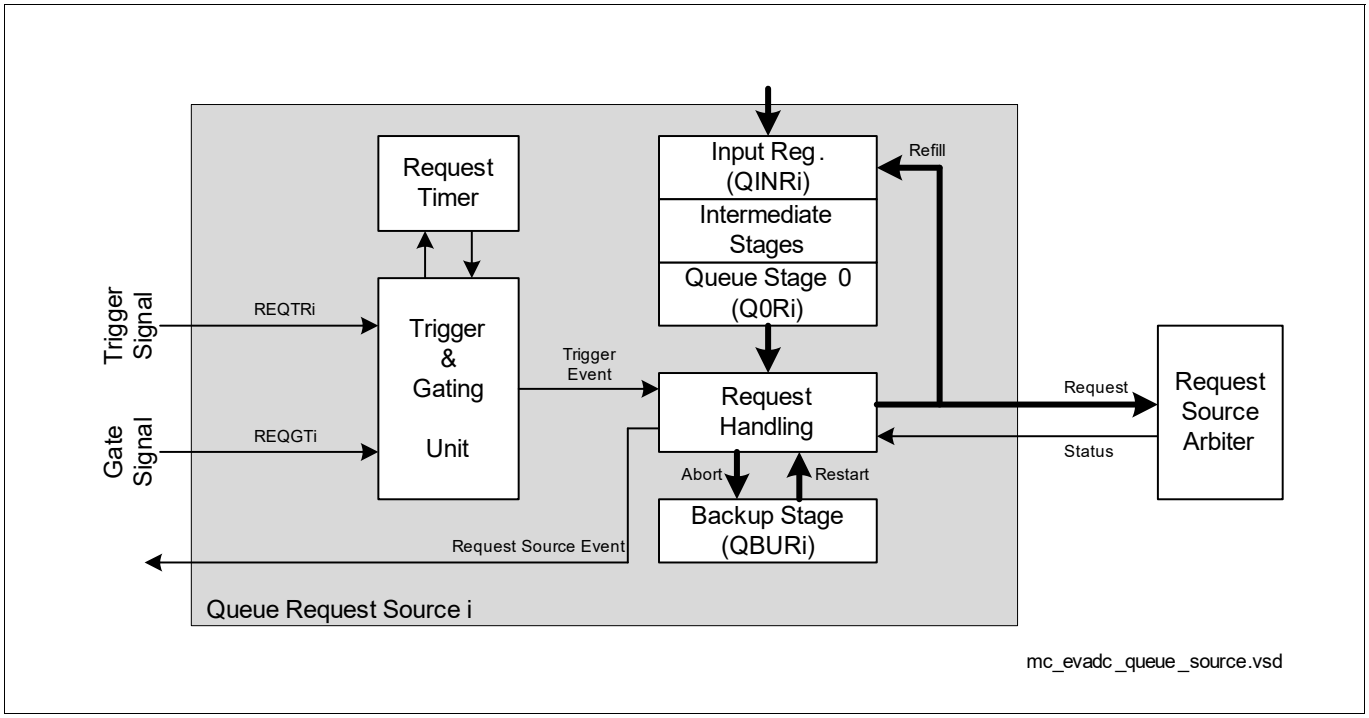


Figure 244 Queued Request Source

A sequence is defined by entering conversion requests into the queue input register (**GxQINRi (i=0-2;x=0-11)**). Each entry selects the channel to be converted and can enable an external trigger, generation of an interrupt, and an automatic refill (i.e. copy this entry to the top of the queue after conversion). The entries are stored in the queue buffer stages.

The content of stage 0 (**GxQ0Ri (i=0-2;x=0-11)**) selects the channel to be converted next. When the requested conversion is started, the contents of this queue stage is invalidated and copied to the backup stage. Then the next queue entry can be handled (if available).

Note: The contents of the queue stages cannot be modified directly, but only by writing to the queue input or by flushing the queue.
*The current status of the queue is shown in register **GxQSRi (i=0-2;x=0-11)**.*
If all queue entries have automatic refill selected, the defined conversion sequence can be repeated without re-programming.

32.5.1 Queued Source Operation

Configure the queued request source by executing the following actions:

- Define the sequence by writing the entries to the queue input **GxQINRi (i=0-2;x=0-11)**. Initialize the complete sequence before enabling the request source, because with enabled refill feature, software writes to QINRx are not allowed.
- If hardware trigger or gating is desired, select the appropriate trigger and gating inputs and the proper transitions by programming **GxQCTRLi (i=0-2;x=0-11)**.
  Enable the trigger and select the gating mode by programming bitfield ENGT in register **GxQMRi (i=0-2;x=0-11)**.[1)]
- Enable the corresponding arbiter input to accept conversion requests from the queued source (see register **GxARBPR (x=0-11)**).

Start a queued sequence by generating a trigger event:

- If a hardware trigger is selected and enabled, generate the configured transition at the selected input signal, e.g. from a timer or an input pin.
- Generate a software trigger event by setting GxQMRi.TREV = 1.
- Write a new entry to the queue input of an empty queue. This leads to a (new) valid queue entry that is forwarded to queue stage 0 and starts a conversion request (if enabled by GxQMRi.ENGT and without waiting for an external trigger).

Note: If the refill mechanism is activated, a processed entry is automatically reloaded into the queue. This permanently repeats the respective sequence (autoscan).
In this case, do not write to the queue input while the queued source is running.
Write operations to a completely filled queue are ignored.

Stop or abort an ongoing queued sequence by executing one of the following actions:

- If external gating is enabled, switch the gating signal to the defined inactive level. This does not modify the queue entries, but only prevents issuing conversion requests to the arbiter.
- Disable the corresponding arbiter input in the arbiter. This does not modify the queue entries, but only prevents the arbiter from accepting requests from the request handling block.
- Disable the queued source:
  - Prevent conversion requests to the arbiter by clearing bitfield ENGT = 00_B.
  - Invalidate the next pending queue entry by setting bit GxQMRi.CLRV = 1.
    If the backup stage contains a valid entry, this one is invalidated, otherwise stage 0 is invalidated.
  - Remove all entries from the queue by setting bit GxQMRi.FLUSH = 1.

Queue Request Source Events and Service Requests

A request source event of a queued source occurs when a conversion is finished and the request timer has expired. A source event service request can be generated based on a request source event according to the structure shown in **Figure 245**. If a request source event is detected, it sets the corresponding indication flag in register **GxSEFLAG (x=0-11)**. These flags can also be set by writing a 1 to the corresponding bit position, whereas writing 0 has no effect. The indication flags can be cleared by SW by writing a 1 to the corresponding bit position in register **GxSEFCLR (x=0-11)**.
The interrupt enable bit is taken from stage 0 for a normal sequential conversion, or from the backup stage for a repeated conversion after an abort.

1) If PDOUT signals from the ERU are used, initialize the ERU accordingly before enabling the gate inputs to avoid unexpected signal transitions.

The service request output line SRx that is selected by the request source event interrupt node pointer bitfields in register **GxSEVNP (x=0-11)** becomes activated each time the related request source event is detected (and enabled by GxQ0R0.ENSI, or GxQBUR0.ENSI respectively) or the related bit position in register **GxSEFLAG (x=0-11)** is written with a 1 (this write action simulates a request source event).

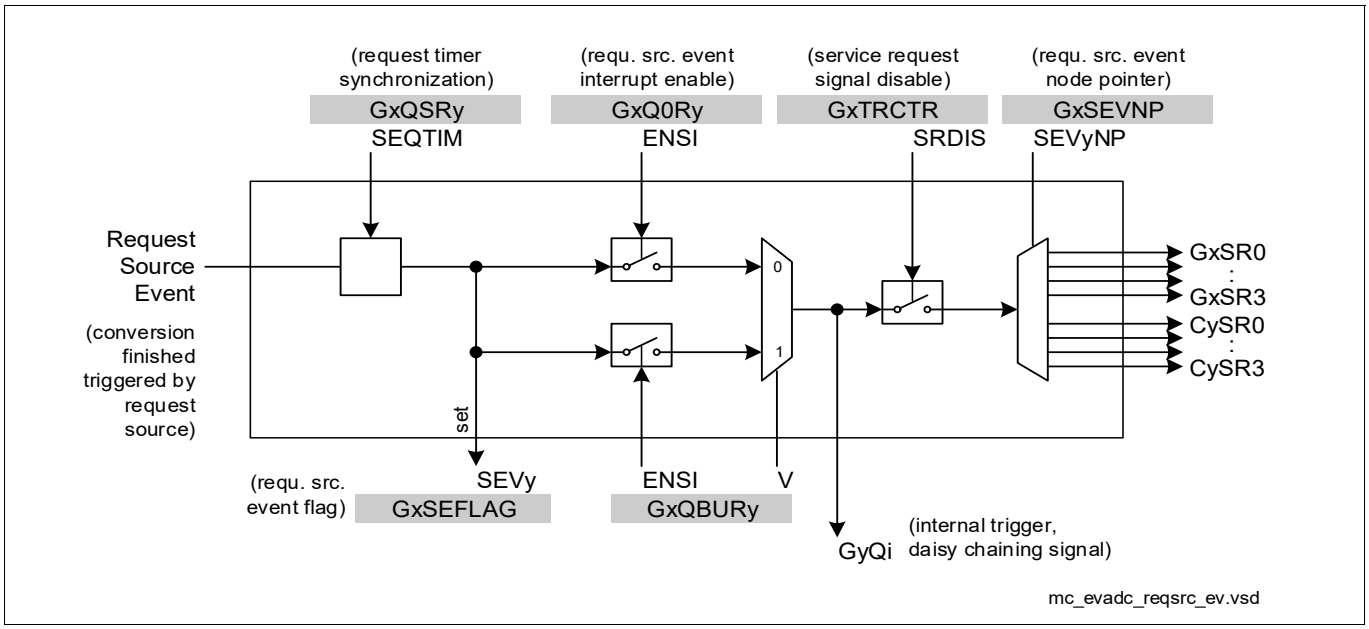


Figure 245 Interrupt Generation of a Queued Request Source

32.5.2 Triggers and Gate Signals

Each source can be started by selectable triggers and/or its operation can be externally controlled by gate signals. These trigger/gate signals can be individually selected from a set of inputs.

The actual signal sources are described in the product-specific appendix.

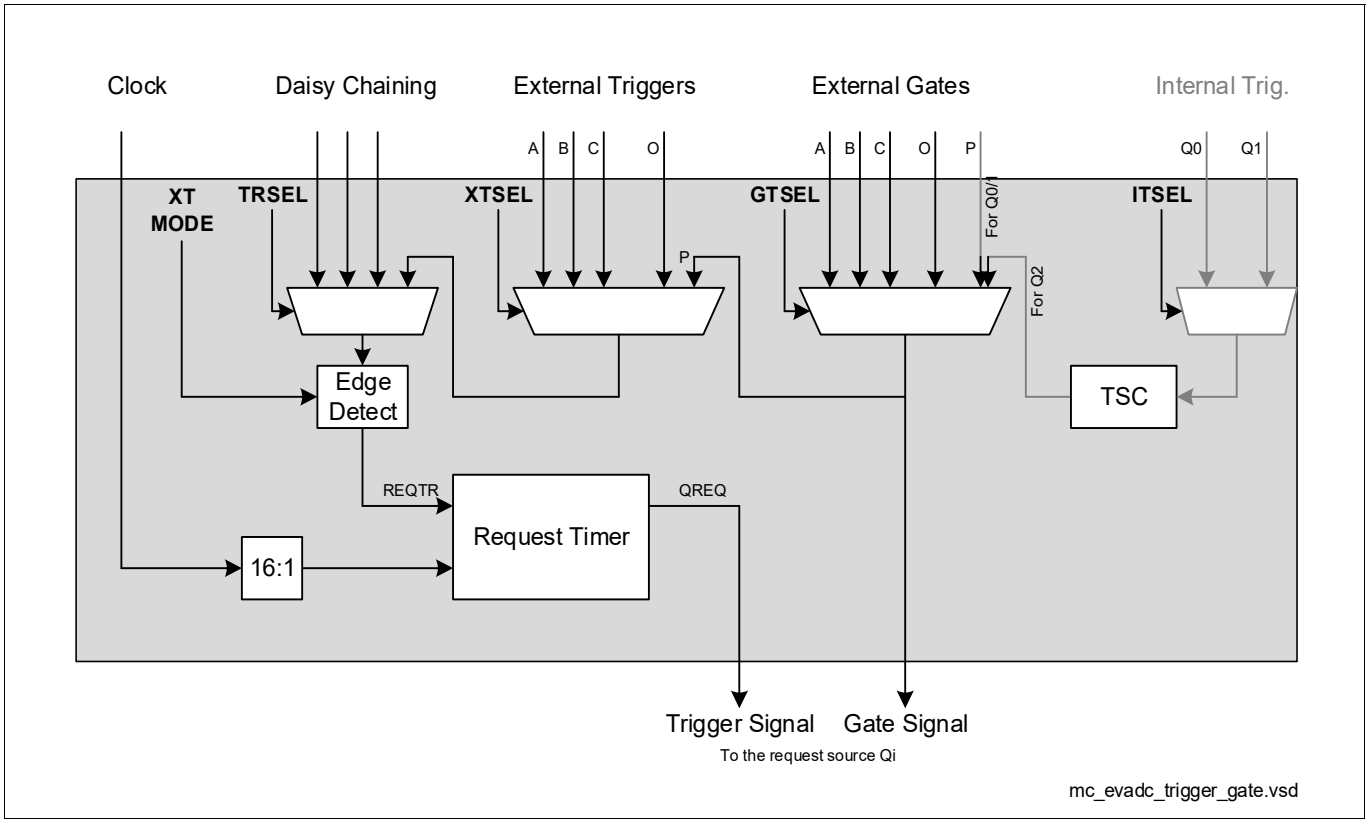


Figure 246 Trigger and Gate Inputs

Enhanced flexibility is provided by using also external gate inputs for the generation of trigger signals. An internal connection of the selected gate input to the uppermost trigger input (GxREQTRyP, XTSEL = 1111_B) provides access to the external gate inputs.

In addition, queue source Q2 can also be triggered by the other request sources of the same group. The selected internal trigger signal is internally connected to the uppermost of the Q2 external gate inputs (GxREQGT2P, GTSEL = 1111_B).

Daisy chaining connects adjacent groups for sequences comprising more than one multiplexer.
This way, extended sequences can be triggered by a single external trigger event.
See also **“Extended Conversion Sequences through Concatenation” on Page 36**.

Self-Timed Execution of Conversion Sequences

The built-in request timer can request conversions of a programmed sequence in configurable intervals. The 16:1 prescaler generates a timebase of 0.1 µs (for f_{ADC} = 160 MHz). The 10-bit timer creates intervals up to 102.4 µs.

The timer is started by a trigger event (generated by hardware or by software), it is not restarted when the queue has run empty. While the request timer is running, additional trigger inputs are ignored.

Several operating modes can be selected:

- **Pause after each conversion**After a trigger event the first conversion is immediately requested. Each subsequent conversion is requested after the request timer has counted the configured number of clock cycles.
- **Wait before first conversion**After a trigger event the request timer is started. The conversion sequence is requested after the request timer has counted the configured number of clock cycles. No further delays are generated then by the timer.
- **Wait before each conversion**After a trigger event the request timer is started. Each conversion of the sequence is requested after the request timer has counted the configured number of clock cycles.

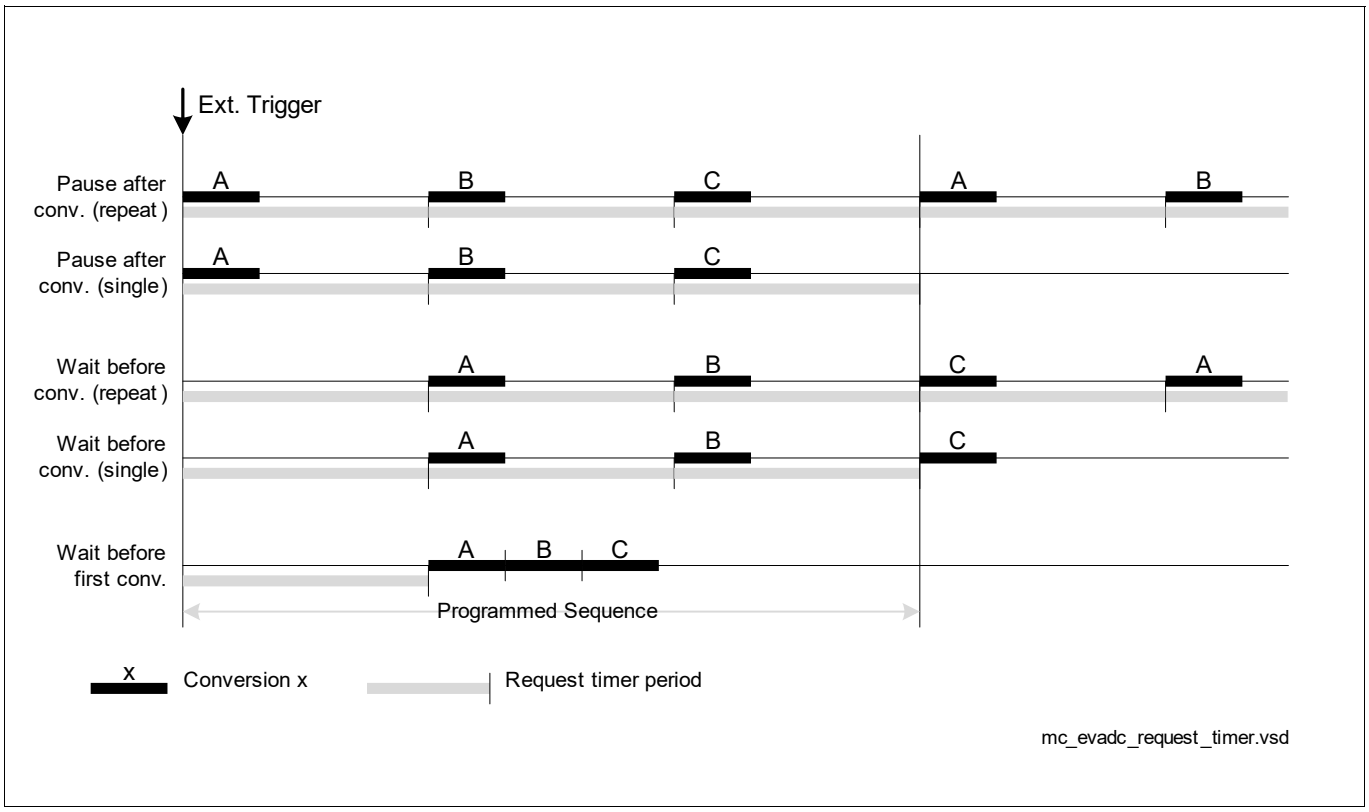


Figure 247 Request Timer Functions

Note: The request timer generates trigger signals for the request source. The conversion commands in the request source determine whether or not a conversion waits for this trigger.
*The first examples in **Figure 247** assume all commands (A, B, C) waiting for a trigger, the last example assumes waiting for a trigger for the first command (A) only.*

When the request timer is active (i.e. GxREQTMi.SEQMOD ≠ 00_B), service request signals are synchronized to the timer. A source event is only signalled at the end of the configured timer period. This ensures equal conversion steps in case of concatenation of sequences using multiple groups. For this functionality the timer must be enabled and the configured period must be long enough to end after the respective conversion.

Controlling Internal and External Triggers

Trigger events can be generated by a transition on the selected hardware trigger signal or by software. In the latter case the external trigger can be excluded from generating trigger events.

When the request timer is enabled, the selected external trigger signal controls the timer itself, while the trigger signal for the corresponding request queue is generated locally by the request timer.

The figure below summarizes the control paths of the engaged blocks.

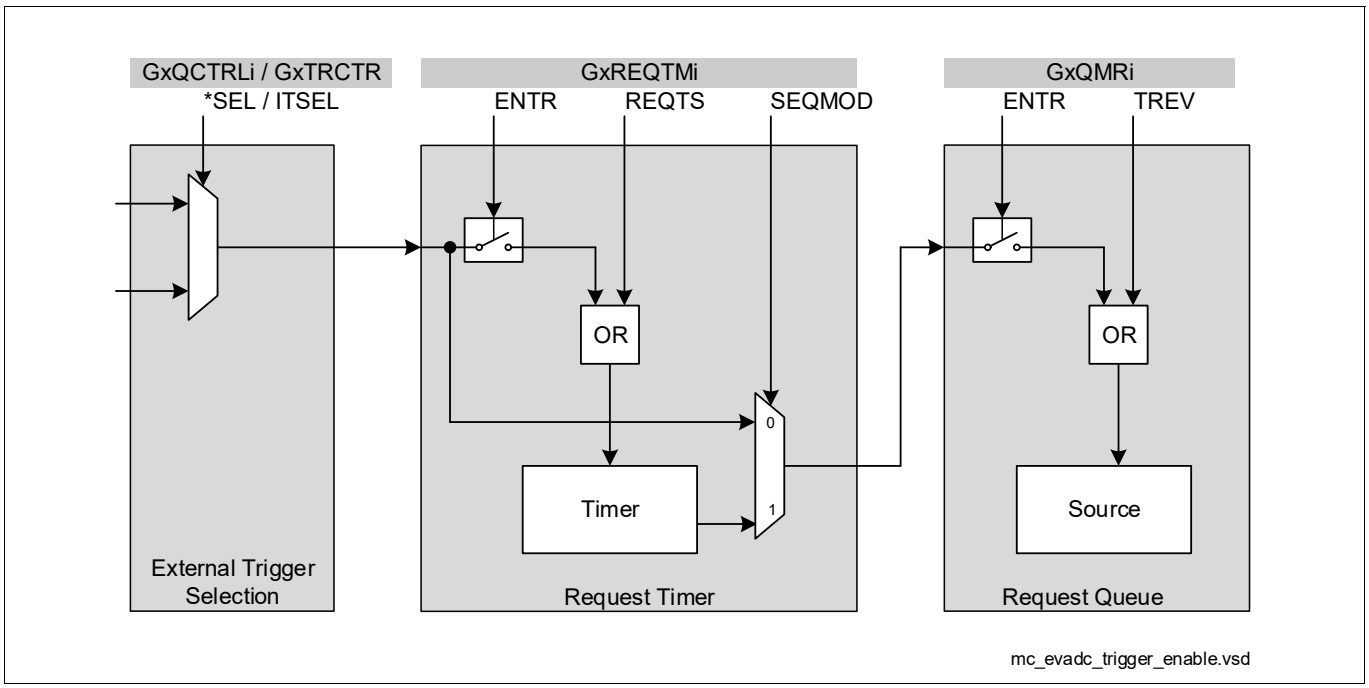


Figure 248 Enabling and Generating Triggers

Controlling the Trigger Output Signal

While a trigger signal usually is a single pulse, its shape can be controlled by writing an appropriate value to bitfield SEQTIMOFF. This is useful to generate the preface time for timer mode, see **“Equidistant Sampling” on Page 112**.

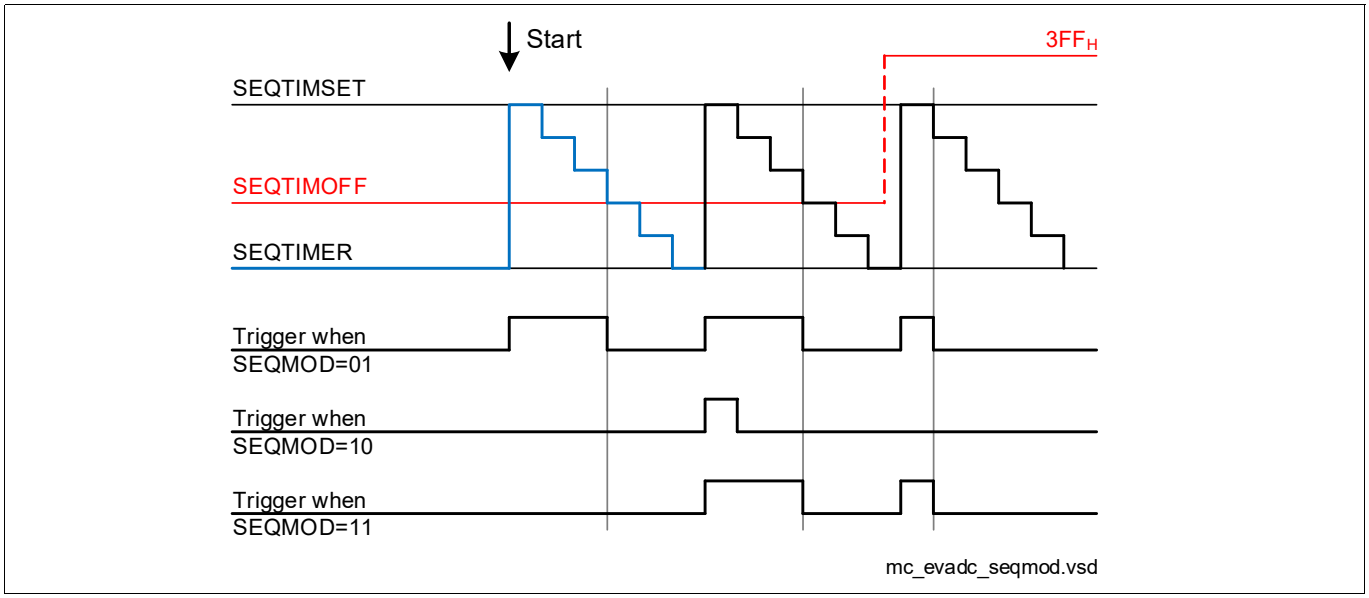


Figure 249 Trigger Output Configuration

32.5.3 Extended Conversion Sequences through Concatenation

A request source can automatically handle sequences of up to 8 or 16 conversions. If longer sequences shall be started with a single trigger event, several request sources can be concatenated, so one source triggers the other.

Three modes of concatenation are possible:

- **Within a group**Request source 2 (Q2) can be internally triggered by another request source of the same group. After the configured sequence has been converted by Q0 or Q1, Q2 is triggered directly (see “Internal Trigger” in **Figure 246**).
  This is useful either to extend the sequence a single queue can handle or to execute test conversions (see additional features of Q2) automatically after a sequence of application conversions.
- **Daisy chaining**If the extended sequence shall cover more channels than available from the respective multiplexer of a given group, a request source can also be triggered by source events of an adjacent group. Daisy chains can be established within a cluster or spanning all groups, sequencing can be done in both directions, wrap-around is possible. These connections are within the module and do not impose any synchronization delays.
  This is useful to configure extended sequences with many channels.
- **Multiple groups**Triggers between groups can also be established by using the service request signal of another group.
  This is useful to configure arbitrary sequences with channels from different groups.

Establishing Daisy-Chain Concatenation

Groups can be concatenated in a configurable daisy-chain. Each group selects a source event of its predecessor group as trigger input.

Example: Concatenation of secondary groups 9-10-11-8:

- Trigger input (rising edge) G10 from G9:
  G10QCTRLi.TRSEL = 01_B, G10QCTRLi.XTMODE = 10_B, G10QMRi.ENTR = 1, G10QINRi.EXTR = 1
- Trigger input (rising edge) G11 from G10:
  G11QCTRLi.TRSEL = 01_B, G11QCTRLi.XTMODE = 10_B, G11QMRi.ENTR = 1, G11QINRi.EXTR = 1
- Trigger input (rising edge) G8 from G11:
  G8QCTRLi.TRSEL = 10_B, G8QCTRLi.XTMODE = 10_B, G8QMRi.ENTR = 1, G8QINRi.EXTR = 1
- Trigger output G9 (no service request):
  G9QINRi.ENSI = 1, G9SEVNP.SEViNP = 1111_B
- Trigger output G10 (no service request):
  G10QINRi.ENSI = 1, G10SEVNP.SEViNP = 1111_B
- Trigger output G11 (no service request):
  G11QINRi.ENSI = 1, G11SEVNP.SEViNP = 1111_B

Establishing Multi-Group Concatenation

Groups can be concatenated via the sevice request lines (see product-specific appendix). The predecessor group routes a source event to a specific service request line, the successor group selects this service request line as trigger input.

Example: Concatenation of queue 0 of group 1 to queue 0 of group 3:

- Generate source event: G1QINR0.ENSI = 1
- Route source event to service request output: G1SEVNP.SEV0NP = 0101_B (common service request C1SR1)
- Select service request as trigger input: G3QCTRL0.XTSEL = 1110_B, G3QCTRL0.XTMODE = 10_B
- Activate trigger for successor group: G3QMR0.ENTR = 1, G3QINR0.EXTR = 1

Note: Do not route other events to the selected service request line.

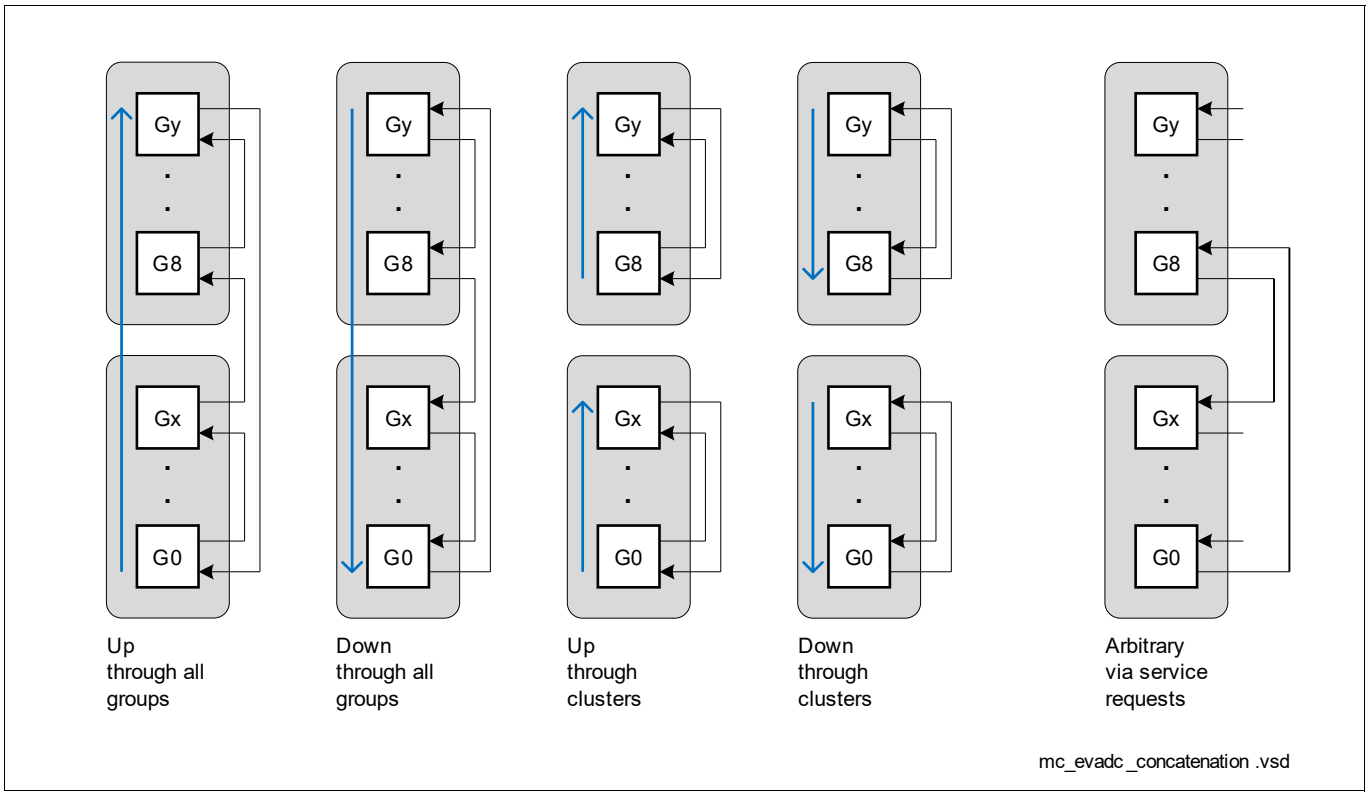


Figure 250 Possibilities for Concatenation

32.5.4 Queue Request Source Control Registers

The following set of registers provides initialization and control of the operation of the queue request sources.

*Note: Requests for channel numbers above the maximum available channel of a group are treated like requests for channel 0. Use the ALIAS feature for higher channel numbers (see **Section 32.7.2**).*

Queue request source Q2 provides two enhancements compared to the other sources Q0 and Q1:

- **Internal Triggers**Q2 can also be internally triggered by request sources Q0/Q1 of the same group (see **GxTRCTR (x=0-11)**).
- **Testfunctions**Q2 can be used to automatically control safety-oriented test conversion sequences. It, therefore, provides the additional control bitfields CDSEL, CDEN, MDPU, MDPD, PDD, which are not available in Q0/Q1.

Queue i Source Contr. Register, Group x

The control register of the queue source selects the external gate and/or trigger signals. Also triggers from other groups can be selected to allow the concatenation of groups to execute extended conversion sequences. Write control bits allow separate control of each function with a simple write access.

GxQCTRLi (i=0-2;x=0-11)

Queue i Source Contr. Register, Group x (0500_H+x*400_H+i*20_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TMWC	0		TMEN	0				GTWC	0		GTLVL	GTSEL			
w	r		rw	r				w	r		rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XTWC	XTMODE		XTLVL	XTSEL				TRSEL		0		SRCRESREG			
w	rw		rh	rw				rw		r		rw			

Field	Bits	Type	Description
SRCRESREG	3:0	rw	**Source-specific Result Register** 0_H Use GxCHCTRy.RESREG/RESTGT to select a result register 1_H Store result in group result register GxRES1 … F_H Store result in group result register GxRES15
TRSEL	7:6	rw	**Trigger Source Selection** *Note: Daisy chaining via source events from the corresponding adjacent request source Qi. Use XTMODE = 10_B.* 00_B External trigger, as selected by XTSEL 01_B Source event of next lower group, for G0: highest available group 10_B Source event of lowest group within cluster (primary or secondary), for lowest group: highest group within cluster 11_B Source event of next higher group, for highest available group: G0

Field	Bits	Type	Description
XTSEL	11:8	rw	**External Trigger Input Selection** The connected trigger input signals are listed in the product-specific appendix. *Note: XTSEL = 1111_B uses the selected gate input as trigger source (ENGT must be $0X_B$).*
XTLVL	12	rh	**External Trigger Level** Current level of the selected trigger input
XTMODE	14:13	rw	**Trigger Operating Mode** 00_B No external trigger 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B Trigger event upon any edge
XTWC	15	w	**Write Control for Trigger Configuration** 0_B No write access to trigger configuration 1_B Bitfields XTMODE, XTSEL, TRSEL can be written
GTSEL	19:16	rw	**Gate Input Selection** The connected gate input signals are listed in the product-specific appendix. *Note: GTSEL = 1111_B uses the selected internal trigger source for queue 2.*
GTLVL	20	rh	**Gate Input Level** Current level of the selected gate input
GTWC	23	w	**Write Control for Gate Configuration** 0_B No write access to gate configuration 1_B Bitfield GTSEL can be written
TMEN	28	rw	**Timer Mode Enable** 0_B No timer mode: standard gating mechanism can be used 1_B Timer mode for equidistant sampling enabled: standard gating mechanism must be disabled
TMWC	31	w	**Write Control for Timer Mode** 0_B No write access to timer mode 1_B Bitfield TMEN can be written
0	5:4, 22:21, 27:24, 30:29	r	**Reserved, write 0, read as 0**

Table 254 Access Mode Restrictions of GxQCTRLi (i=0-2;x=0-11) sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **XTWC**	rw	TRSEL, XTMODE, XTSEL	Set XTWC during write access
write 1 to **GTWC**	rw	GTSEL	Set GTWC during write access

Table 254 Access Mode Restrictions of GxQCTRLi (i=0-2;x=0-11) sorted by descending priority (cont'd)

Mode Name	Access Mode		Description
write 1 to **TMWC**	rw	TMEN	Set TMWC during write access
(default)	r	GTSEL, TMEN, TRSEL, XTMODE, XTSEL	

Queue i Mode Register, Group x

The Queue Mode Register configures the operating mode of a queued request source.

GxQMRi (i=0-2;x=0-11)
Queue i Mode Register, Group x **(0504_H+x*400_H+i*20_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															RPTDIS
r															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				CEV	FLUSH	TREV	CLRV	0					ENTR	ENGT	
r				w	w	w	w	r					rw	rw	

Field	Bits	Type	Description
ENGT	1:0	rw	**Enable Gate** Selects the gating functionality for the request source. *Note: REQGTx is the selected gating signal.* 00_B No conversion requests are issued 01_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register 10_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 1 11_B Conversion requests are issued if a valid conversion request is pending in the queue 0 register or in the backup register and REQGTx = 0
ENTR	2	rw	**Enable External Trigger** 0_B External trigger disabled 1_B The selected edge at the selected trigger input signal REQTR generates the trigger event
CLRV	8	w	**Clear Valid Bit** 0_B No action 1_B The next pending valid queue entry in the sequence and the event flag EV are cleared. If there is a valid entry in the queue backup register (QBUR.V = 1), this entry is cleared, otherwise the entry in queue register 0 is cleared.

Field	Bits	Type	Description
TREV	9	w	**Trigger Event** Generates a software trigger for the request source.[1)] 0_B No action 1_B Generate a trigger event by software
FLUSH	10	w	**Flush Queue** 0_B No action 1_B Clear all queue entries (incl. backup stage) and the event flag EV. The queue contains no more valid entry.
CEV	11	w	**Clear Event Flag** 0_B No action 1_B Clear bit EV
RPTDIS	16	rw	**Repeat Disable** 0_B A cancelled conversion is repeated 1_B A cancelled conversion is discarded
0	7:3, 15:12, 31:17	r	**Reserved, write 0, read as 0**

1) To trigger the request timer instead, write 1 to bit GxREQTIMi.REQTS.

*Note: For the function of bit RPTDIS see also **Section 32.6.2**.*

Queue i Status Register, Group x

The Queue Status Register indicates the current status of the queued source. The filling level and the empty information refer to the queue intermediate stages (if available) and to the queue register 0. An aborted conversion stored in the backup stage is not indicated by these bits (therefore, see QBURx.V).

GxQSRi (i=0-2;x=0-11)
Queue i Status Register, Group x **(0508_H+x*400_H+i*20_H)** **Application Reset Value: 0000 0020_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							EV	REQGT	0	EMPTY	0	FILL			
r							rh	rh	r	rh	r	rh			

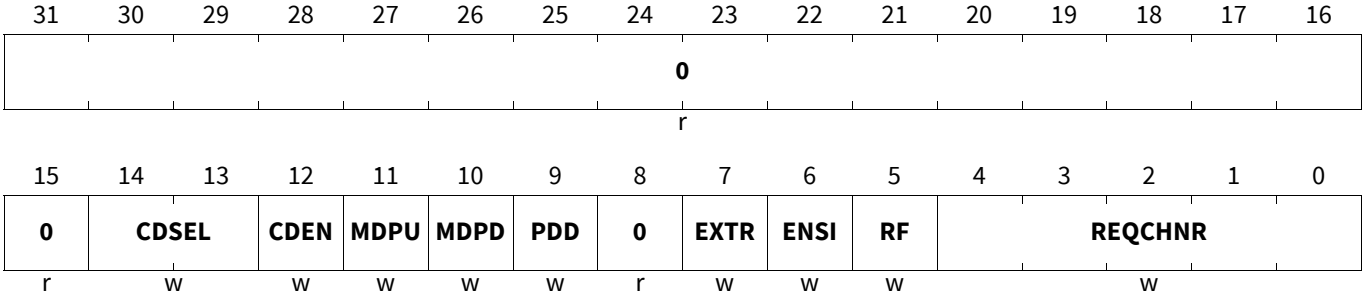
Field	Bits	Type	Description
FILL	3:0	rh	**Filling Level for Queue** Indicates the number of valid queue entries. It is incremented each time a new entry is written to QINRx or by an enabled refill mechanism. It is decremented each time a requested conversion has been started. A new entry is ignored if the filling level has reached its maximum value. Maximum fill level for primary groups: 8 entries Maximum fill level for secondary groups: 16 entries 0_H There is 1 (if EMPTY = 0) or no (if EMPTY = 1) valid entry in the queue 1_H There are 2 valid entries in the queue … 7_H There are 8 valid entries in the queue 8_H There are 9 valid entries in the queue … F_H There are 16 valid entries in the queue
EMPTY	5	rh	**Queue Empty** 0_B There are valid entries in the queue (see FILL) 1_B No valid entries (queue is empty)
REQGT	7	rh	**Request Gate Level** Monitors the level at the selected REQGT input. 0_B The gate input is low 1_B The gate input is high
EV	8	rh	**Event Detected** Indicates that an event has been detected while at least one valid entry has been in the queue (queue register 0 or backup stage). Once set, this bit is cleared automatically when the requested conversion is started. 0_B No trigger event 1_B A trigger event has been detected
0	4, 6, 31:9	r	**Reserved, write 0, read as 0**

Queue i Input Register, Group x

The Queue Input Register is the entry point for conversion requests of a queued request source.[1)]

1) Bitfields CDSEL, CDEN, MDPU, MDPD, PDD are only available in Q2, not in Q0/Q1.

GxQINRi (i=0-2;x=0-11)
Queue i Input Register, Group x **(0510_H+x*400_H+i*20_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
REQCHNR	4:0	w	**Request Channel Number** Defines the channel number to be converted. Not available channel numbers are treated as channel 0.
RF	5	w	**Refill** 0_B No refill: this queue entry is converted once and then invalidated 1_B Automatic refill: this queue entry is automatically reloaded into QINRx when the related conversion is started
ENSI	6	w	**Enable Source Interrupt** 0_B No request source interrupt 1_B A request source event interrupt is generated upon a request source event (related conversion is finished)
EXTR	7	w	**External Trigger** Enables the external trigger functionality. *Note: To use external triggers, enable them by setting bit GxQMRy.ENTR.* 0_B A valid queue entry immediately leads to a conversion request. 1_B A valid queue entry waits for a trigger event to occur before issuing a conversion request.
PDD	9	w	**Pull-Down Diagnostics Enable** *Note: Channels with pull-down diagnostics device are marked in the product-specific appendix.* 0_B Disconnected 1_B The pull-down diagnostics device is active

Field	Bits	Type	Description
MDPD	10	w	**Multiplexer Diagnostics Pull-Devices Enable** Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with multiplexer diagnostics pull devices are marked in the product-specific appendix.* 0_B Disconnected 1_B The respective device is active
MDPU	11	w	**Multiplexer Diagnostics Pull-Devices Enable - MDPD,MDPU** Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with multiplexer diagnostics pull devices are marked in the product-specific appendix.* 0_B Disconnected 1_B The respective device is active
CDEN	12	w	**Converter Diagnostics Enable** 0_B All diagnostic pull devices are disconnected 1_B Diagnostic pull devices connected as selected by bitfield CDSEL
CDSEL	14:13	w	**Converter Diagnostics Pull-Devices Select** 00_B Connected to VDDM 01_B Connected to VSSM 10_B Connected to 1/2 VDDM 11_B Connected to 2/3rd VDDM
0	8, 31:15	r	**Reserved, write 0, read as 0**

Queue i Register 0, Group x

The queue register 0 monitors the status of the pending request (queue stage 0).[1)]

GxQ0Ri (i=0-2;x=0-11)

Queue i Register 0, Group x **($050C_H+x*400_H+i*20_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**CDSEL**		**CDEN**	**MDPU**	**MDPD**	**PDD**	**V**	**EXTR**	**ENSI**	**RF**	**REQCHNR**				
r	rh		rh	rh	rh	rh	rh	rh	rh	rh	rh				

1) Bitfields CDSEL, CDEN, MDPU, MDPD, PDD are only available in Q2, not in Q0/Q1.

Field	Bits	Type	Description
REQCHNR	4:0	rh	**Request Channel Number** Indicates the channel number to be converted.
RF	5	rh	**Refill** Indicates the handling of handled requests. 0_B The request is discarded after the conversion start. 1_B The request is automatically refilled into the queue after the conversion start.
ENSI	6	rh	**Enable Source Interrupt** 0_B No request source interrupt 1_B A request source event interrupt is generated upon a request source event (related conversion is finished)
EXTR	7	rh	**External Trigger** Enables external trigger events. 0_B A valid queue entry immediately leads to a conversion request 1_B The request handler waits for a trigger event
V	8	rh	**Request Channel Number Valid** Indicates a valid queue entry in queue register 0. 0_B No valid queue entry 1_B The queue entry is valid and leads to a conversion request
PDD	9	rh	**Pull-Down Diagnostics Enable** *Note: Channels with pull-down diagnostics device are marked in the product-specific appendix.* 0_B Disconnected 1_B The pull-down diagnostics device is active
MDPD	10	rh	**Multiplexer Diagnostics Pull-Devices Enable** Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with multiplexer diagnostics pull devices are marked in the product-specific appendix.* 0_B Disconnected 1_B The respective device is active
MDPU	11	rh	**Multiplexer Diagnostics Pull-Devices Enable - MDPD,MDPU** Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with multiplexer diagnostics pull devices are marked in the product-specific appendix.* 0_B Disconnected 1_B The respective device is active

Field	Bits	Type	Description
CDEN	12	rh	**Converter Diagnostics Enable** 0_B All diagnostic pull devices are disconnected 1_B Diagnostic pull devices connected as selected by bitfield CDSEL
CDSEL	14:13	rh	**Converter Diagnostics Pull-Devices Select** 00_B Connected to VDDM 01_B Connected to VSSM 10_B Connected to 1/2 VDDM 11_B Connected to 2/3rd VDDM
0	31:15	r	**Reserved, write 0, read as 0**

Queue i Backup Register, Group x

The Queue Backup Registers monitor the status of an aborted queued request.[1)]

GxQBURi (i=0-2;x=0-11)
Queue i Backup Register, Group x **(0514_H+x*400_H+i*20_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CDSEL		CDEN	MDPU	MDPD	PDD	V	EXTR	ENSI	RF	REQCHNR				
r	rh		rh	rh	rh	rh	rh	rh	rh	rh	rh				

Field	Bits	Type	Description
REQCHNR	4:0	rh	**Request Channel Number** The channel number of the aborted conversion that has been requested by this request source
RF	5	rh	**Refill** The refill control bit of the aborted conversion
ENSI	6	rh	**Enable Source Interrupt** The enable source interrupt control bit of the aborted conversion
EXTR	7	rh	**External Trigger** The external trigger control bit of the aborted conversion
V	8	rh	**Request Channel Number Valid** Indicates if the entry (REQCHNR, RF, TR, ENSI) in the queue backup register is valid. Bit V is set when a running conversion (that has been requested by this request source) is aborted, it is cleared when the aborted conversion is restarted. 0_B Backup register not valid 1_B Backup register contains a valid entry. This will be requested before a valid entry in queue register 0 (stage 0) will be requested.

1) Bitfields CDSEL, CDEN, MDPU, MDPD, PDD are only available in Q2, not in Q0/Q1.

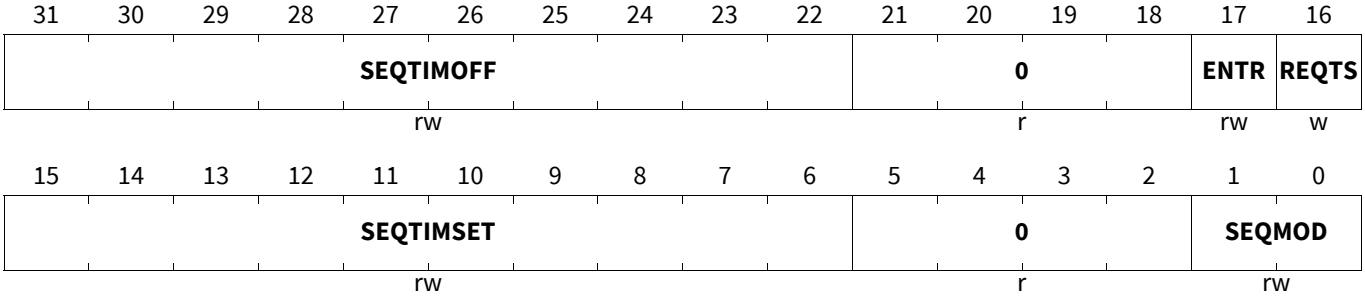
Field	Bits	Type	Description
PDD	9	rh	**Pull-Down Diagnostics Enable** *Note: Channels with pull-down diagnostics device are marked in the product-specific appendix.* 0_B Disconnected 1_B The pull-down diagnostics device is active
MDPD	10	rh	**Multiplexer Diagnostics Pull-Down Devices Enable** Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with multiplexer diagnostics pull devices are marked in the product-specific appendix.* 0_B Disconnected 1_B The respective device is active
MDPU	11	rh	**Multiplexer Diagnostics Pull-Up Devices Enable** Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with multiplexer diagnostics pull devices are marked in the product-specific appendix.* 0_B Disconnected 1_B The respective device is active
CDEN	12	rh	**Converter Diagnostics Enable** 0_B All diagnostic pull devices are disconnected 1_B Diagnostic pull devices connected as selected by bitfield CDSEL
CDSEL	14:13	rh	**Converter Diagnostics Pull-Devices Select** 00_B Connected to VDDM 01_B Connected to VSSM 10_B Connected to 1/2 VDDM 11_B Connected to 2/3rd VDDM
0	31:15	r	**Reserved, write 0, read as 0**

Queue i Requ. Timer Mode Reg., Group x

The Request Timer Mode Register configures the operating mode of a source-specific request timer.

GxREQTMi (i=0-2;x=0-11)

Queue i Requ. Timer Mode Reg., Group x (0518_H+x*400_H+i*20_H) **Application Reset Value: FFC0 0000_H**



Field	Bits	Type	Description
SEQMOD	1:0	rw	**Sequence Mode** Selects how the request timer controls the conversion sequence. Before changing the operating mode, stop the sequence timer, i.e. SEQMOD = 00_B. 00_B Request timer off Trigger events become effective immediately. 01_B Pause after conversion Conversions are requested after a trigger event and each time the request timer has expired. 10_B Wait before first conversion After a trigger event the request timer is started. The conversion sequence is executed after the request timer has expired. 11_B Wait before each conversion After a trigger event the request timer is started. Conversions are requested each time the request timer has expired.
SEQTIMSET	15:6	rw	**Sequence Timer, Set Value** Initial value for the sequence timer in steps of 16 × t_{PER}. This value is loaded into SEQTIMER when a new request timer period is started.
REQTS	16	w	**Request Timer Start Trigger** 0_B No action 1_B Start the request timer immediately via software[1)]
ENTR	17	rw	**Enable External Trigger** 0_B External trigger disabled 1_B The selected edge at the selected trigger input signal starts the request timer
SEQTIMOFF	31:22	rw	**Sequence Timer, Switch Off Value** The generated trigger signal is disabled when the timer value is equal to or below this threshold.
0	5:2, 21:18	r	**Reserved, write 0, read as 0**

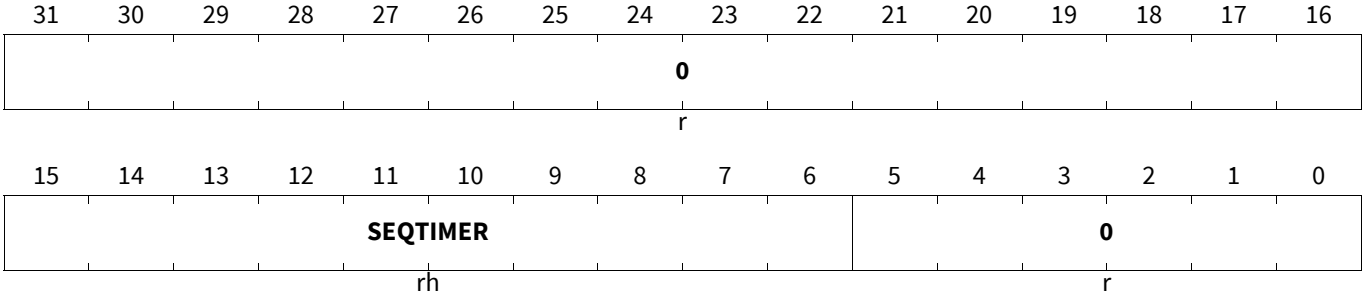
1) Mode and start/stop values must be configured before setting the trigger bit REQTS.

Queue i Requ. Timer Status Reg., Group x

The Request Timer Status Register returns the current value of source-specific request timer.

GxREQTSi (i=0-2;x=0-11)

Queue i Requ. Timer Status Reg., Group x($051C_H$+x*400_H+i*20_H) **Application Reset Value: 0000 0000_H**



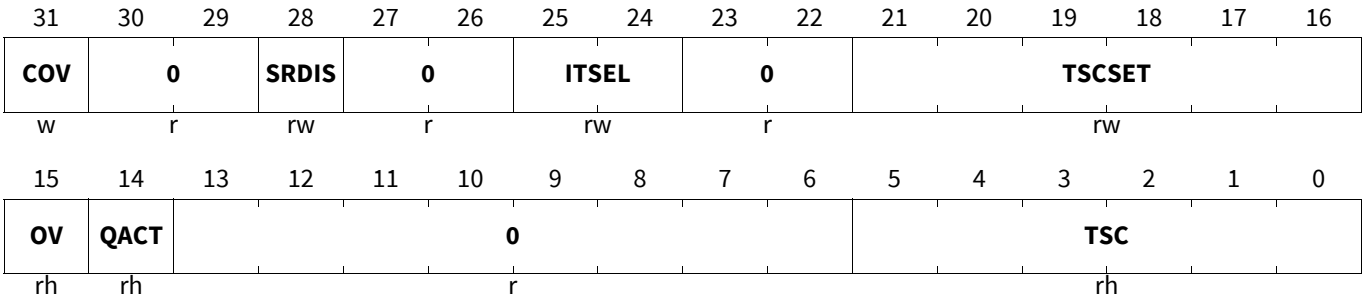
Field	Bits	Type	Description
SEQTIMER	15:6	rh	**Sequence Timer** Counts the request timer periods in steps of 16 × t_{ADC}. This timer is loaded from bitfield SEQTIMSET at the beginning of a period.
0	5:0, 31:16	r	**Reserved, write 0, read as 0**

Trigger Control Register, Group x

For queued request source Q2, the Trigger Control Register configures the trigger sequence counter and the internal trigger sources.

GxTRCTR (x=0-11)

Trigger Control Register, Group x **(0410_H+x*400_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TSC	5:0	rh	**Trigger Sequence Counter** Controls the effect of an incoming internal trigger: If TSC > 00_H: decrement TSC by one. 00_H Issue conversion requests immediately[1)]
QACT	14	rh	**Queue Active** Indicates that request source Q2 is currently executing a sequence. Cleared by writing 1 to bit COV. 0_B No activity 1_B Queue currently active

Field	Bits	Type	Description
OV	15	rh	**Overflow Detected** Indicates that a trigger has been activated while the queue was still active (QACT = 1). Cleared by writing 1 to bit COV. 0_B No trigger event 1_B An irregular trigger event has been detected
TSCSET	21:16	rw	**Trigger Sequence Counter Start Value** Defines the initial value of the trigger sequence counter TSC. TSC is reloaded with the value in TSCSET, when a trigger occurs while TSC = 00_H. TSCSET is automatically copied to TSC when being written.[1)]
ITSEL	25:24	rw	**Internal Trigger Input Selection** Internal triggers are generated by the respective source events. Enable a source event in the selected request source to generate an internal trigger signal. *Note: The selected trigger signal is internally connected to gate input GxREQGTP of this request source. It is selected when XTSEL = GTSEL = 1111_B.* 00_B Select queued request source Q0 01_B Select queued request source Q1 10_B Reserved 11_B Reserved
SRDIS	28	rw	**Service Request Disable** Controls if the source event of the selected trigger source also activates a service request. 0_B Source event generates service request 1_B No service request, only internal trigger generated
COV	31	w	**Clear Overflow Flag** 0_B No action 1_B Clear bits OV and QACT
0	13:6, 23:22, 27:26, 30:29	r	**Reserved, write 0, read as 0**

1) Write zero to bitfield TSCSET, to disable the trigger sequence counter and start the sequence with each incoming trigger.

32.6 Request Source Arbitration

The request source arbiter scans the request sources for pending conversion requests. Each request source is connected to a certain input of the arbiter.

The priority of each request source is user-configurable via register **GxARBPR (x=0-11)**, so the arbiter can select the next channel to be converted, in the case of concurrent requests from multiple sources, according to the application requirements. The synchronization source has always higher priority than all other sources.

After reset, all inputs are disabled and must be enabled (register **GxARBPR (x=0-11)**) to take part in the arbitration process.

The arbiter determines the highest priority conversion request. If this request has sufficient priority the corresponding conversion is started.

The following request sources are available:

- Input 0: **Queued request source Q0**, 8/16-stage sequences in arbitrary order
- Input 1: **Queued request source Q1**, 8/16-stage sequences in arbitrary order
- Input 2: **Queued request source Q2**, 8/16-stage sequences in arbitrary order, intra-group concatenation, test
- Input 3: **Synchronization source**, synchronized conversion requests from another ADC master kernel (always handled with the highest priority in a synchronization slave kernel).

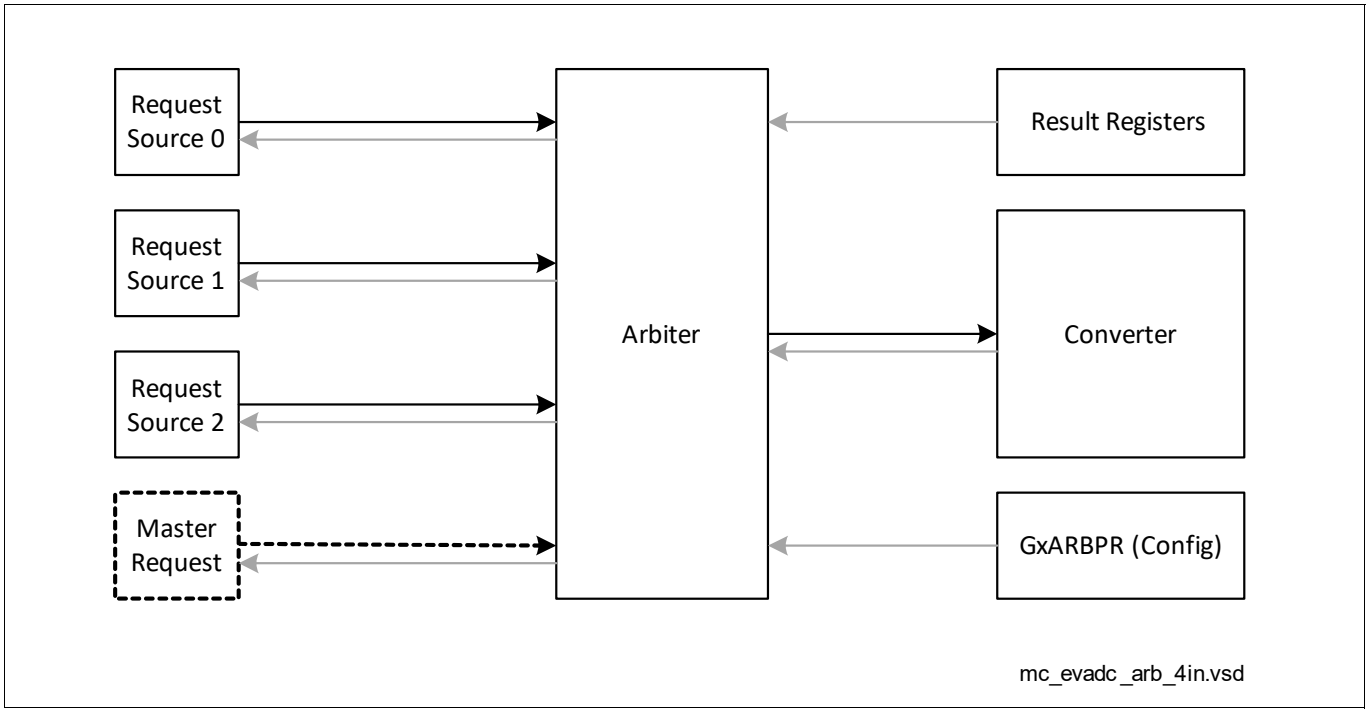


Figure 251 Arbiter with 4 Arbitration Inputs

32.6.1 Arbiter Operation and Configuration

Each request source has a configurable priority (0 = lowest, 3 = highest), so the arbiter can resolve concurrent conversion requests from different sources. The request with the highest priority is selected for conversion. These priorities can be adapted to the requirements of a given application (see register **GxARBPR (x=0-11)**).

The arbiter is controlled by bits in registers GxARBCFG and GxARBPR. A decision is made within one clock cycle, if any conversion request is active. While no conversion request is active, the arbiter remains idle.

The **Conversion Start Mode** determines the handling of the conversion request that has won the arbitration.

To achieve a reproducible reaction time (constant delay without jitter) between the trigger event of a conversion request (e.g. by a timer unit or due to an external event) and the start of the related conversion, the converter has to be idle. This is supported in particular by the timer mode (see **“Equidistant Sampling” on Page 112**).

Arbitration Config. Register, Group x

The Arbitration Configuration Register selects the timing and the behavior of the arbiter. Also some group-specific configurations can be selected here.

GxARBCFG (x=0-11)

Arbitration Config. Register, Group x **(0480_H+x*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAMPLE	BUSY	0	CAL	0		SYNRUN	CHNR					CSRC		ANONS	
rh	rh	r	rh	r		rh	rh					rh		rh	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														ANONC	
r														rw	

Field	Bits	Type	Description
ANONC	1:0	rw	**Analog Converter Control** Defines the value of bitfield ANONS in a stand-alone converter or a converter in master mode. Coding see ANONS or **Section 32.4**.
ANONS	17:16	rh	**Analog Converter Control Status** Defined by bitfield ANONC in a stand-alone kernel or a kernel in master mode. In slave mode, this bitfield is defined by bitfield ANONC of the respective master kernel. See also **Section 32.4**. 00_B Analog converter off 01_B Slow standby mode 10_B Fast standby mode 11_B Normal operation (permanently on)

Field	Bits	Type	Description
CSRC	19:18	rh	**Currently Converted Request Source** Indicates the arbitration slot number of the current (BUSY = 1) or of the last (BUSY = 0) conversion. This bitfield is updated when a conversion is started. 00_B Current/last conversion for request source 0 01_B Current/last conversion for request source 1 10_B Current/last conversion for request source 2 11_B Current/last conversion for synchronization request (slave converter)
CHNR	24:20	rh	**Channel Number** Indicates the current or last converted analog input channel. This bitfield is updated when a conversion is started.
SYNRUN	25	rh	**Synchronous Conversion Running** Indicates that a synchronized (= parallel) conversion is currently running. 0_B Normal conversion or no conversion running 1_B A synchronized conversion is running (cannot be cancelled by higher priority requests!)
CAL	28	rh	**Start-Up Calibration Active Indication** Indicates the start-up calibration phase of the corresponding analog converter. *Note: Start conversions only after the start-up calibration phase is complete, because a start-up calibration will abort a running conversion.* 0_B Completed or not yet started 1_B Start-up calibration phase is active (set one clock cycle after setting bit SUCAL)
BUSY	30	rh	**Converter Busy Flag** 0_B Not busy 1_B Converter is busy with a conversion (set one clock cycle after conversion start)
SAMPLE	31	rh	**Sample Phase Flag** 0_B Converting or idle 1_B Input signal is currently sampled
0	15:2, 27:26, 29	r	**Reserved, write 0, read as 0**

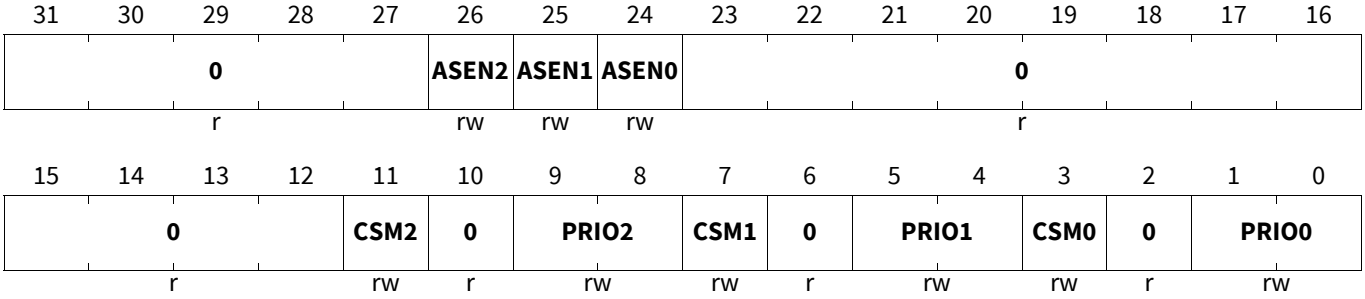
Arbitration Priority Register, Group x

The Arbitration Priority Register defines the request source priority and the conversion start mode for each request source.

Note: Only change priority and conversion start mode settings of a request source while this request source is disabled, and a currently running conversion requested by this source is finished.

GxARBPR (x=0-11)

Arbitration Priority Register, Group x **(0484_H+x*400_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PRIOi (i=0-2)	4*i+1:4*i	rw	**Priority of Request Source i** Arbitration priority of request source i (at input i) 00_B Select priority level 0 … 11_B Select priority level 3
CSMi (i=0-2)	4*i+3	rw	**Conversion Start Mode of Request Source i** 0_B Wait-for-start mode 1_B Cancel-inject-repeat mode i.e. this source can cancel conversions of other sources.
ASENi (i=0-2)	i+24	rw	**Arbitration Source Input i Enable** Enables the associated arbitration source input of the arbiter. The request source bits are not modified by write actions to ASENi. 0_B The arbitration source input is disabled. Pending conversion requests from the associated request source are disregarded. 1_B The arbitration source input is enabled. Pending conversion requests from the associated request source are arbitrated.
0	10, 6, 2, 23:12, 31:27	r	**Reserved, write 0, read as 0**

32.6.2 Conversion Start Mode

When the arbiter has selected the request to be converted next, the handling of this channel depends on the current activity of the converter:

- Converter is currently idle:
  the conversion of the arbitration winner is started immediately.
- Current conversion has same or higher priority:
  the current conversion is completed, the conversion of the arbitration winner is started after that.
- Current conversion has lower priority:
  the action is user-configurable:
  - **Wait-for-start mode**: the current conversion is completed, the conversion of the arbitration winner is started after that. This mode provides maximum throughput, but can produce a jitter for the higher priority conversion.
    Example in **Figure 252**:
    Conversion A is requested (t1) and started (t2). Conversion B is then requested (t3), but started only after completion of conversion A (t4).
  - **Cancel-inject-repeat mode**: the current conversion is aborted, the conversion of the arbitration winner is started after the abortion (3 f_{ADC} cycles).
    The aborted conversion request is restored in the corresponding request source and takes part again in the next arbitration step. This mode provides minimum jitter for the higher priority conversions, but reduces the overall throughput.
    Example in **Figure 252**:
    Conversion A is requested (t6) and started (t7). Conversion B is then requested (t8) and started (t9), while conversion A is aborted but requested again. When conversion B is complete (t10), conversion A is restarted.
    Exception: If both requests target the same result register with wait-for-read mode active (see **Section 32.10.3**), the current conversion cannot be aborted.

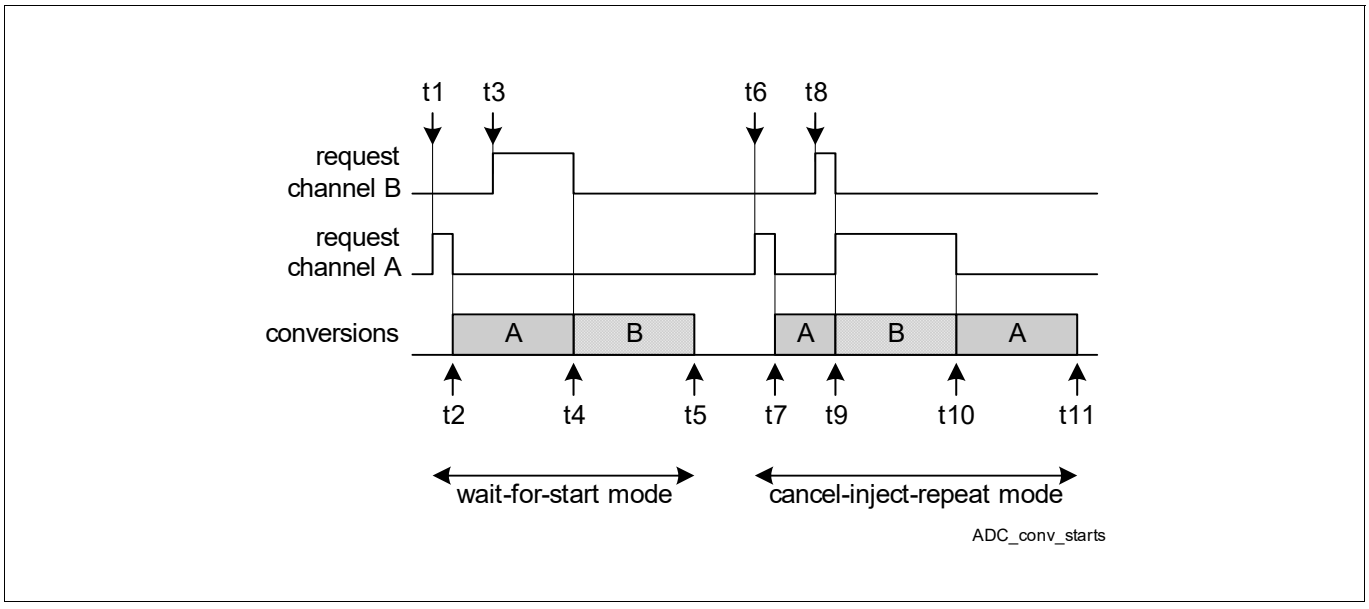


Figure 252 Conversion Start Modes

The conversion start mode can be individually programmed for each request source by bits in register **GxARBPR (x=0-11)** and is applied to all channels requested by the source. In this example, channel A is issued by a request source with a lower priority than the request source requesting the conversion of channel B.

A cancelled conversion can be repeated automatically in each case, or it can be discarded if it was cancelled. This is selected for each source by bit RPTDIS in the corresponding source's mode register.

Note: Discarding a cancelled conversion is not possible when an external multiplexer is operated in sequence mode. Bit RPTDIS must be 0 in this case.

32.7 Analog Input Channel Configuration

For each analog input channel a number of parameters can be configured that control the conversion of this channel. The channel control registers define the following parameters:

- **Channel Parameters**: The sample time for this channel and analog input precharge option are defined via input classes. Each channel can select one of two classes of its own group or one of two global classes.
- **Reference selection**: an alternate reference voltage can be selected for most channels (exceptions are marked in the product-specific appendix)
- **Result target**: The conversion result values are stored either in a group-specific result register or in the global result register[1]. The group-specific result registers are selected in several ways:
  - channel-specific, selected by bitfield RESREG in register **GxCHCTRy (x=0-7;y=0-7)** etc., with bitfield SRCRESREG = 0000_B
  - source-specific, selected by bitfield SRCRESREG in register **GxQCTRLi (i=0-2;x=0-11)** etc. with bitfield SRCRESREG ≠ 0000_B
  - The global result register is selected when SRCRESREG = 0000_B and RESTGT = 1.
- **Result position**: The result values of standard conversions can be stored left-aligned or right-aligned, other results are stored right-aligned. The global result register GLOBRES always stores results right-aligned. See also **Figure 263 “Result Storage Options” on Page 98**.
- **Compare with Standard Conversions (Limit Checking)**: Channel events can be generated whenever a new result value becomes available. Channel event generation can be restricted to values that lie inside or outside a user-configurable band.
- **Broken Wire Detection**: This safety feature can detect a missing connection to an analog signal source (sensor).
- **Synchronization of Conversions**: Synchronized conversions are executed at the same time on several converters.

The **Alias Feature** redirects conversion requests for channels CH0 and/or CH1 to other channels.

1) Conversion slaves cannot store results to the global result register.
Slaves must use local result registers in this case (GxCHCTRy.RESTGT = 0).

32.7.1 Channel Parameters

Each analog input channel is configured by its associated channel control register.

*Note: For the safety feature "Broken Wire Detection", refer to **Section 32.12.4**.*

Group 0, Channel 0 Control Register

The following features can be defined for each channel:

- The conversion class defines the conversion mode (noise reduction) and the sample time
- Generation of channel events and the result value band, if used
- Target of the result defining the target register and the position within the register

GxCHCTRy (x=0-7;y=0-7)
Group x, Channel y Control Register (0600_H+x*400_H+y*4) Application Reset Value: 0000 0000_H
GxCHCTRy (x=8-11;y=0-15)
Group x, Channel y Control Register (0600_H+x*400_H+y*4) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	BWDEN	BWDCH		0						RESPOS	RESTGT	RESREG			
r	rw	rw		r						rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDSELX				REFSEL	SYNC	CHEVMODE		BNDSELU		BNDSELL		0		ICLSEL	
rw				rw	rw	rw		rw		rw		r		rw	

Field	Bits	Type	Description
ICLSEL	1:0	rw	**Input Class Select** 00_B Use group-specific class 0 01_B Use group-specific class 1 10_B Use global class 0 11_B Use global class 1
BNDSELL	5:4	rw	**Lower Boundary Select** 00_B Use group-specific boundary 0 01_B Use group-specific boundary 1 10_B Use global boundary 0 11_B Use global boundary 1
BNDSELU	7:6	rw	**Upper Boundary Select** 00_B Use group-specific boundary 0 01_B Use group-specific boundary 1 10_B Use global boundary 0 11_B Use global boundary 1

Field	Bits	Type	Description
CHEVMODE	9:8	rw	**Channel Event Mode** Generate a channel event (with limit checking[1)]) 00_B Never 01_B If result is inside the boundary band 10_B If result is outside the boundary band 11_B Always (ignore band)
SYNC	10	rw	**Synchronization Request** 0_B No synchronization request, standalone operation 1_B Request a synchronized conversion of this channel (only taken into account for a master)
REFSEL	11	rw	**Reference Input Selection** Defines the reference voltage input to be used for conversions on this channel. [2)] 0_B Standard reference input VAREF 1_B Alternate reference input from CH0
BNDSELX	15:12	rw	**BoundaryExtension** While BNDSELX ≠ 0000_B, bitfields BNDSELU and BNDSELL are concatenated and select the corresponding result register as lower boundary. 0_H Standard mode select boundaries via BNDSELU/BNDSELL 1_H Use result reg. GxRES1 as upper boundary … F_H Use result reg. GxRES15 as upper boundary
RESREG	19:16	rw	**Result Register** 0_H Store result in group result register GxRES0 … F_H Store result in group result register GxRES15
RESTGT	20	rw	**Result Target** 0_B Store results in the selected group result register 1_B Store results in the global result register (not possible for synchronization slaves)
RESPOS	21	rw	**Result Position** 0_B Read results right-aligned (all modes) 1_B Read results left-aligned (std. conversions only)
BWDCH	29:28	rw	**Broken Wire Detection Channel** 00_B Select VAREF for precharging 01_B Select VAGND for precharging 10_B Reserved 11_B Reserved
BWDEN	30	rw	**Broken Wire Detection Enable** 0_B Normal operation 1_B Additional preparation phase is enabled
0	3:2, 27:22, 31	r	**Reserved, write 0, read as 0**

1) The boundary band is defined as the area where the result is less than or equal to the selected upper boundary and greater than or equal to the selected lower boundary, see **Section 32.7.3**.
2) Some channels cannot select an alternate reference.

Input Class Register i, Group x

The group-specific input class registers define the sample time and data conversion mode for each channel of any group that selects them via bitfield ICLSEL in its channel control register GxCHCTRy.

GxICLASSi (i=0-1;x=0-11)
Input Class Register i, Group x **($04A0_H$+x*400_H+i*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					SESPE	CME		AIPE		0	STCE				
r					rw	rw		rw		r	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					SESPS	CMS		AIPS		0	STCS				
r					rw	rw		rw		r	rw				

Field	Bits	Type	Description
STCS	4:0	rw	**Sample Time Control for Standard Conversions** Number of additional clock cycles to be added to the minimum sample phase of 2 sample clock cycles: Coding and resulting sample time see **Table 255**. For conversions of external channels, the value from bitfield STCE can be used.
AIPS	7:6	rw	**Analog Input Precharge Control for Standard Conversions** *Note: Buffer must be enabled by BE = 1 (see **GxANCFG**).* 00_B No precharge 01_B Precharge for 8 clock cycles 10_B Precharge for 16 clock cycles 11_B Precharge for 32 clock cycles
CMS	9:8	rw	**Conversion Mode for Standard Conversions** 00_B Standard conversion 01_B Noise reduction conversion level 1, 1 additional conversion step 10_B Noise reduction conversion level 2, 3 additional conversion steps 11_B Noise reduction conversion level 3, 7 additional conversion steps
SESPS	10	rw	**Spread Early Sample Point for Standard Conversions** 0_B Nominal sample timing 1_B Spread sample timing, end of sample phase is varied

Field	Bits	Type	Description
STCE	20:16	rw	**Sample Time Control for EMUX Conversions** Number of additional clock cycles to be added to the minimum sample phase of 2 sample clock cycles: Coding and resulting sample time see **Table 255**. For conversions of standard channels, the value from bitfield STCS is used.
AIPE	23:22	rw	**Analog Input Precharge Control for EMUX Conversions** *Note: Buffer must be enabled by BE = 1 (see **GxANCFG**).* 00_B No precharge 01_B Precharge for 8 clock cycles 10_B Precharge for 16 clock cycles 11_B Precharge for 32 clock cycles
CME	25:24	rw	**Conversion Mode for EMUX Conversions** 00_B Standard conversion 01_B Noise reduction conversion level 1, 1 additional conversion step 10_B Noise reduction conversion level 2, 3 additional conversion steps 11_B Noise reduction conversion level 3, 7 additional conversion steps
SESPE	26	rw	**Spread Early Sample Point for EMUX Conversions** 0_B Nominal sample timing 1_B Spread sample timing, end of sample phase is varied
0	5, 15:11, 21, 31:27	r	**Reserved, write 0, read as 0**

Input Class Register i, Global

The global input class registers define the sample time and data conversion mode for each channel of the respective group that selects them via bitfield ICLSEL in its channel control register GxCHCTRy.

GLOBICLASSi (i=0-1)

Input Class Register i, Global **($00A0_H$+i*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					**SESPE**	**CME**		**AIPE**		**0**	**STCE**				
r					rw	rw		rw		r	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					**SESPS**	**CMS**		**AIPS**		**0**	**STCS**				
r					rw	rw		rw		r	rw				

Field	Bits	Type	Description
STCS	4:0	rw	**Sample Time Control for Standard Conversions** Number of additional clock cycles to be added to the minimum sample phase of 2 sample clock cycles: Coding and resulting sample time see **Table 255**. For conversions of external channels, the value from bitfield STCE can be used.
AIPS	7:6	rw	**Analog Input Precharge Control for Standard Conversions** *Note: Buffer must be enabled by BE = 1 (see **GxANCFG**).* 00_B No precharge 01_B Precharge for 8 clock cycles 10_B Precharge for 16 clock cycles 11_B Precharge for 32 clock cycles
CMS	9:8	rw	**Conversion Mode for Standard Conversions** 00_B Standard conversion 01_B Noise reduction conversion level 1, 1 additional conversion step 10_B Noise reduction conversion level 2, 3 additional conversion steps 11_B Noise reduction conversion level 3, 7 additional conversion steps
SESPS	10	rw	**Spread Early Sample Point for Standard Conversions** 0_B Nominal sample timing 1_B Spread sample timing, end of sample phase is varied
STCE	20:16	rw	**Sample Time Control for EMUX Conversions** Number of additional clock cycles to be added to the minimum sample phase of 2 sample clock cycles: Coding and resulting sample time see **Table 255**. For conversions of standard channels, the value from bitfield STCS is used.
AIPE	23:22	rw	**Analog Input Precharge Control for EMUX Conversions** *Note: Buffer must be enabled by BE = 1 (see **GxANCFG**).* 00_B No precharge 01_B Precharge for 8 clock cycles 10_B Precharge for 16 clock cycles 11_B Precharge for 32 clock cycles
CME	25:24	rw	**Conversion Mode for EMUX Conversions** 00_B Standard conversion 01_B Noise reduction conversion level 1, 1 additional conversion step 10_B Noise reduction conversion level 2, 3 additional conversion steps 11_B Noise reduction conversion level 3, 7 additional conversion steps
SESPE	26	rw	**Spread Early Sample Point for EMUX Conversions** 0_B Nominal sample timing 1_B Spread sample timing, end of sample phase is varied

Field	Bits	Type	Description
0	5, 15:11, 21, 31:27	r	**Reserved, write 0, read as 0**

Sample Time Control

The sample time is defined by a configurable number of sample clock cycles. The sample clock is the analog clock f_{ADCI}. The sample time can, therefore, be configured in steps of t_{ADCI} = 50/37.5/25 ns (for f_{ADCI} = 20/26.7/40 MHz). The table below lists the possible settings.

Table 255 Sample Time Coding

STCS / STCE / STCF	Additional Clock Cycles	Resulting Sample Time	Time for f_{ADCI} = 20 MHz	Time for f_{ADCI} = 53.3 MHz	Clock Cycle Formula
$0\ 0000_B$	0	2 / f_{ADCI}	100 ns	37.5 ns	2 + STCi
$0\ 0001_B$	1	3 / f_{ADCI}	150 ns	56.25 ns	
…	…	…	…	…	
$0\ 0100_B$	4	6 / f_{ADCI}	300 ns	112.5 ns	
…	…	…	…	…	
$0\ 1111_B$	15	17 / f_{ADCI}	850 ns	318.75 ns	
$1\ 0000_B$	16	18 / f_{ADCI}	900 ns	337.5 ns	2 + (STCi - 15) × 16
$1\ 0001_B$	32	34 / f_{ADCI}	1.7 µs	637.5 ns	
…	…	…	…	…	
$1\ 1110_B$	240	242 / f_{ADCI}	12.1 µs	4.5 µs	
$1\ 1111_B$	256	258 / f_{ADCI}	12.9 µs	4.8 µs	

The configured sample time must include the input buffering time (if selected by bitfields AIPi) and the timespan subtracted by sample point spreading (if selected by bitfields SESPi).

Note: Find more information in section ***"Conversion Timing" on Page 86***.

32.7.2 Alias Feature

The Alias Feature redirects conversion requests for channels CH0 and/or CH1 to other channel numbers. This feature can be used to trigger conversions of the same input channel by independent events and to store the conversion results in different result registers.

- The same signal can be measured twice without the need to read out the conversion result to avoid data loss. This allows triggering both conversions quickly one after the other and being independent from CPU/DMA service request latency.
- The sensor signal is connected to only one analog input (instead of two analog inputs). This saves input pins in low-cost applications and only the leakage of one input has to be considered in the error calculation.
- Even if the analog input CH0 is used as alternative reference (see **Figure 253**), the internal trigger and data handling features for channel CH0 can be used.
- The channel settings for both conversions can be different (boundary values, service requests, etc.).

In typical low-cost AC-drive applications, only one common current sensor is used to determine the phase currents. Depending on the applied PWM pattern, the measured value has different meanings and the sample points have to be precisely located in the PWM period. **Figure 253** shows an example where the sensor signal is connected to one input channel (CHx) but two conversions are triggered for two different channels (CHx and CH0). With the alias feature, a conversion request for CH0 leads to a conversion of the analog input CHx instead of CH0, but taking into account the settings for CH0. Although the same analog input (CHx) has been measured, the conversion results can be stored and read out from the result registers RESx (conversion triggered for CHx) and RESy (conversion triggered for CH0). Additionally, different interrupts or limit boundaries can be selected, enabled or disabled.

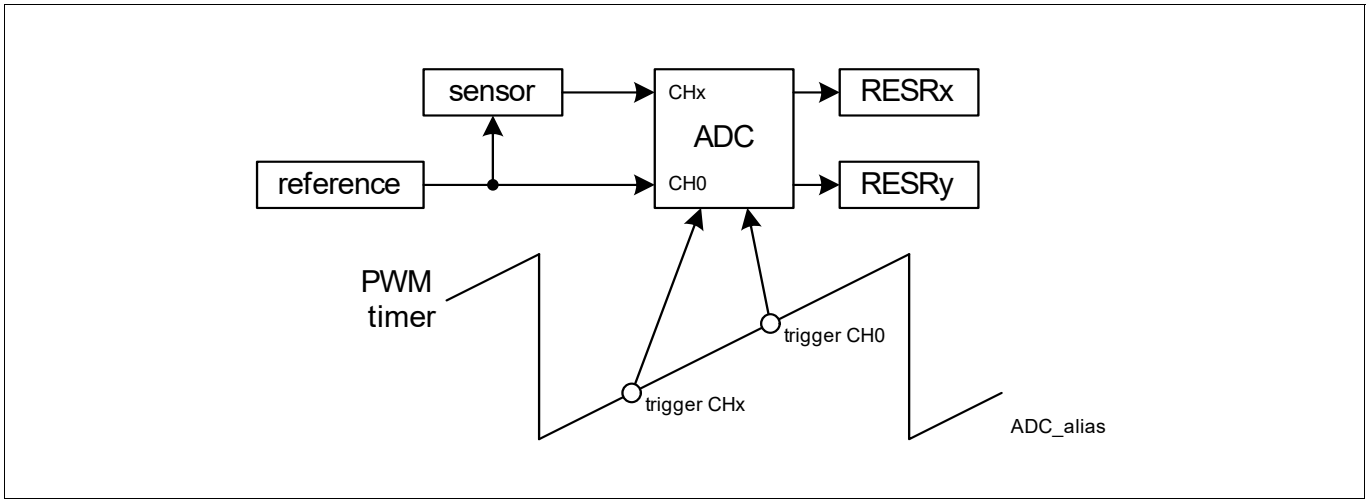


Figure 253 Alias Feature

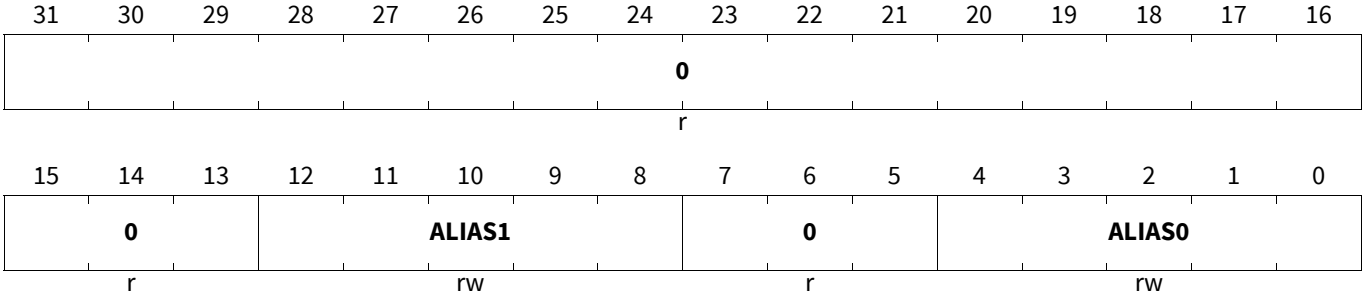
*Note: The redirection via ALIASi remains active until register **GxALIAS (x=0-11)** is reprogrammed.*

Alias Register, Group x

The alias register can replace the channel numbers of channels CH0 and CH1 with another channel number. The reset value disables this redirection.

GxALIAS (x=0-11)

Alias Register, Group x **($04B0_H$+x*400_H)** **Application Reset Value: 0000 0100_H**



Field	Bits	Type	Description
ALIAS0	4:0	rw	**Alias Value for CH0 Conversion Requests** Indicates the channel that is converted instead of channel CH0. The conversion is done with the settings defined for channel CH0.
ALIAS1	12:8	rw	**Alias Value for CH1 Conversion Requests** Indicates the channel that is converted instead of channel CH1. The conversion is done with the settings defined for channel CH1.
0	7:5, 31:13	r	**Reserved, write 0, read as 0**

32.7.3 Compare with Standard Conversions (Limit Checking)

The limit checking mechanism can automatically compare each digital conversion result to an upper and a lower boundary value. A channel event can then be generated when the result of a conversion[1)]/comparison is inside or outside a user-defined band (see bitfield CHEVMODE and **Figure 254**).

This feature supports automatic range monitoring and minimizes the CPU load by issuing service requests only under certain predefined conditions.

Note: Channel events can also be generated for each result value (ignoring the band) or they can be suppressed completely.

The boundary values to which results are compared can be selected from several sources (see register GxCHCTRy).

While bitfield BNDSELX = 0000_B, bitfields BNDSELU and BNDSELL select the valid upper/lower boundary value either from the group-specific boundary register **GxBOUND (x=0-11)** or from the global boundary register **GLOBBOUND**. The group boundary register can be selected for each channel of the respective group, the global boundary register can be selected by each available channel.

Otherwise, the compare values are taken from the lower 12 bits of result registers, where bitfield BNDSELX selects the upper boundary value (GxRES1 … GxRES15), the concatenated bitfields BNDSELU||BNDSELL select the lower boundary value (GxRES0 … GxRES15).

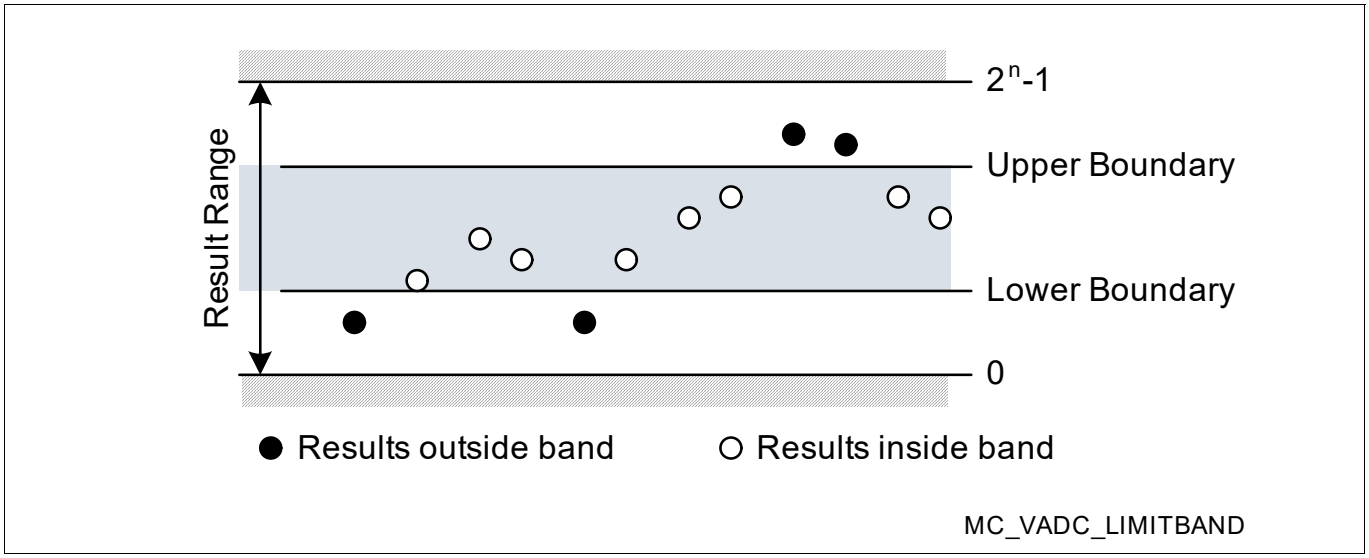


Figure 254 Result Monitoring through Limit Checking

A result value is considered inside the defined band when both of the following conditions are true:

- the value is less than or equal to the selected upper boundary
- the value is greater than or equal to the selected lower boundary

The result range can also be divided into two areas:
To select the lower part as valid band, set the lower boundary to the minimum value (000_H) and set the upper boundary to the highest intended value.
To select the upper part as valid band, set the upper boundary to the maximum value (FFF_H) and set the lower boundary to the lowest intended value.

1) Limit checking uses the direct conversion result, even if FIR/IIR filters are activated.

Finding Extrema (Peak Detection)

The limit checking mechanism uses standard conversions and, therefore, always can provide the actual conversion result that was used for comparison. Combining this with a special FIFO mode (see also **"Result FIFO Buffer" on Page 99**), that only updates the corresponding FIFO stage if the result was above (or below) the current value of the stage, provides the usual conversion results and at the same time stores the highest (or lowest) result of a conversion sequence. For this operation the FIFO stage below the standard result must be selected as the compare value. Mode selection is done via bitfield FEN in register GxRCRy.

Before starting a peak detection sequence, write a reasonable start value to the result bitfield in the peak result register (e.g. 0000_H to find the maximum and $0FFF_H$ to find the minimum).

Note: Only the lower 12 bits are compared!
It is, therefore, recommended to use only standard right-aligned conversions.

Selecting Compare Values

Values for digital compare operations can be selected from several sources. The separate GxBOUND registers or the common GLOBBOUND register provide software-defined compare values. Compare values can also be taken from a result register where it can be provided by another channel building a reference.

The compare values are selected via bitfields BNDSELL/BNDSELU and BNDSELX in register GxCHCTR.

Boundary Select Register, Group x

The local boundary register GxBOUND defines group-specific boundary values.

GxBOUND (x=0-11)
Boundary Select Register, Group x ($04B8_H$+x*400_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				BOUNDARY1											
r				rw											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				BOUNDARY0											
r				rw											

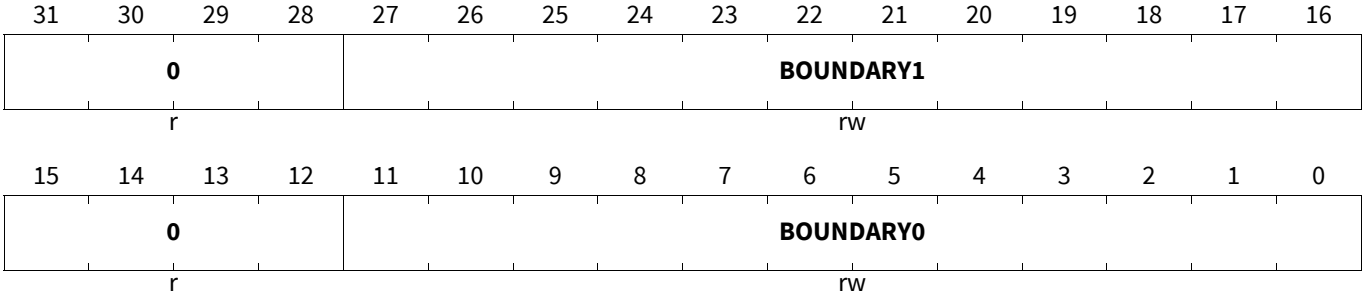
Field	Bits	Type	Description
BOUNDARY0	11:0	rw	**Boundary Value 0 for Limit Checking** This value is compared against the conversion result.
BOUNDARY1	27:16	rw	**Boundary Value 1 for Limit Checking** This value is compared against the conversion result.
0	15:12, 31:28	r	**Reserved, write 0, read as 0**

Global Boundary Select Register

The global boundary register GLOBBOUND defines general compare values for all channels.

GLOBBOUND

Global Boundary Select Register **($00B8_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
BOUNDARY0	11:0	rw	**Boundary Value 0 for Limit Checking** This value is compared against the conversion result.
BOUNDARY1	27:16	rw	**Boundary Value 1 for Limit Checking** This value is compared against the conversion result.
0	15:12, 31:28	r	**Reserved, write 0, read as 0**

32.8 Fast Compare Channel Operation

A Fast Compare channel compares the input signal directly to a reference value[1] stored in bitfield FCREF in register **FCxFCM (x=0-7)**. Delta values define a hysteresis. This comparison returns a binary result (available in bit FCR) which indicates if the compared input voltage is above or below the given reference value. This comparison only requires a few clock cycles and, thus, allows a high sampling rate.

The request timer generates the internal compare triggers for each compare operation. The trigger interval defines the compare rate in the range of 16 … 4 096 clocks (0.1 … 25.6 µs for f_{ADC} = 160 MHz).

*Note: Select the trigger interval according to the configured compare timing (interval > compare time, i.e. $t_{FCTRIV} > t_{FC}$ + 1 CLK, see also **"Fast Compare Channels Timing" on Page 87**).*

The selected external trigger controls the ramp generation or selects the reference value in alternate value mode. In lock BFL mode, it controls the output of the boundary flag. See **Figure 256** and **Section 32.8.1**.

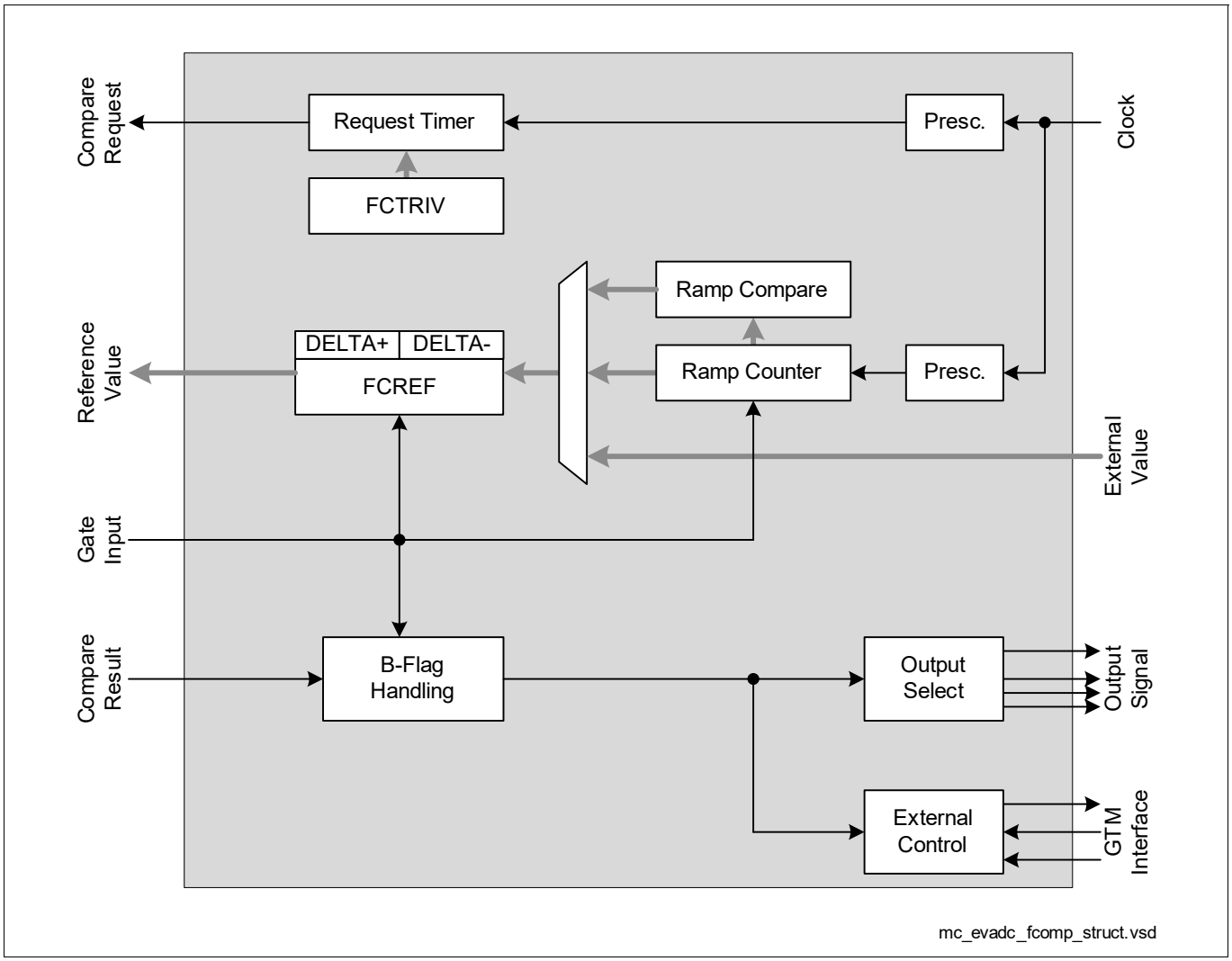


Figure 255 Fast Compare Channel Structure

*Note: See also **Figure 260 "Boundary Flag Control" on Page 83**.*

1) Fast compare channels use 10-bit values and reference to V_{AREF}. The idle precharge function is not supported.

A channel event can be generated when the input signal becomes higher (or lower) than the reference value including a hysteresis that can be defined asymmetrically (see **Figure 258**).

The control interface of a fast compare channel is optimized for its specific functionality. The figure below summarizes the available control functions.

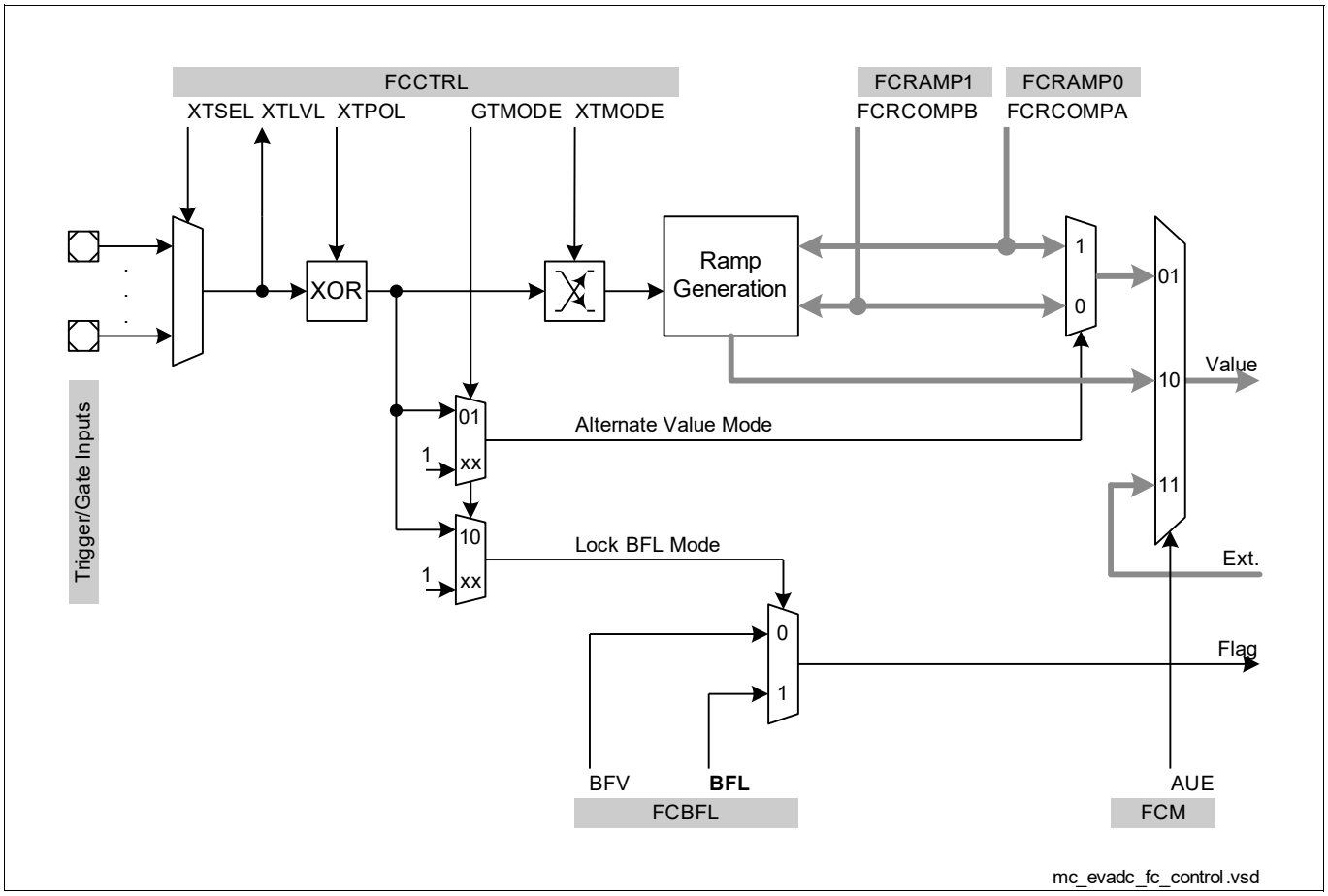


Figure 256 Fast Compare Channel Control Signals

32.8.1 Peak-and-Hold Operation

The Fast Compare channels can implemented hardware-controlled peak-and-hold control loops that operate without software support. The value to be controlled (e.g. current) is observed via the analog input, the target value is given by the reference value, the boundary flag output controls the external circuitry (e.g. power switch).

The reference value can be generated from several sources. Four operating modes are available:

- **Software Mode**
- **Alternate Value Mode**
- **Ramp Mode**
- **External Mode**

The intended operating mode is selected via bitfield AUE in register **FCxFCM (x=0-7)**.

Software Mode

This mode uses a static reference value which is written to bitfield FCREF in register **FCxFCM (x=0-7)**. Software can change this reference value to generate a certain waveform of the controlled signal (see “Update” in **Figure 257**).

Alternate Value Mode

In this mode, one of two predefined reference values is selected by an external gate signal. The reference values are defined by bitfield FCRCOMPA and bitfield FCRCOMPB, depending on the level of the selected hardware gate signal.

- While the gate signal is active (high),
  the reference value in FCRCOMPA (**FCxFCRAMP0 (x=0-7)**) is applied.
- While the gate signal is inactive (low),
  the reference value in FCRCOMPB (**FCxFCRAMP1 (x=0-7)**) is applied.

Ramp Mode

The Fast Compare channel can automatically generate a ramp waveform of the controlled signal. Start and end values for the reference are defined in bitfields FCRCOMPA in register **FCxFCRAMP0 (x=0-7)** and FCRCOMPB in register **FCxFCRAMP1 (x=0-7)**. Bitfield FCRSTEP defines the speed of the generated ramp.

The stepwidth is defined as (FCRSTEP+1) × 8 / f_{ADC}, i.e. 50 ns to 12.8 µs (f_{ADC} = 160 MHz).

This feature can implement e.g. soft start without software support (see “Ramp Up/Down” in **Figure 257**).

The defined slope can also represent an adaptive threshold within one period of the control loop. This is supported by starting the ramp counter upon the leading edge of the gate signal (gate becomes active).

The ramp can also be started automatically by writing a start value to register FCxFCRAMP0.

The ramp stops when the counter reaches FCRCOMPB or, optionally, triggered by an external signal.

See register **FCxFCM (x=0-7)**.

External Mode

The reference value for the Fast Compare channel can be derived from the associated converter group. The result of an analog input of this group can be automatically copied to bitfield FCREF in register **FCxFCM (x=0-7)**. In this mode, the controlled signal will follow the selected input channel of the associated converter group (see “Update” in **Figure 257**).

The reference converter group uses the **“Hardware Data Interface” on Page 107** to copy its result values.

The association of groups and fast compare channels is described in the product-specific appendix.

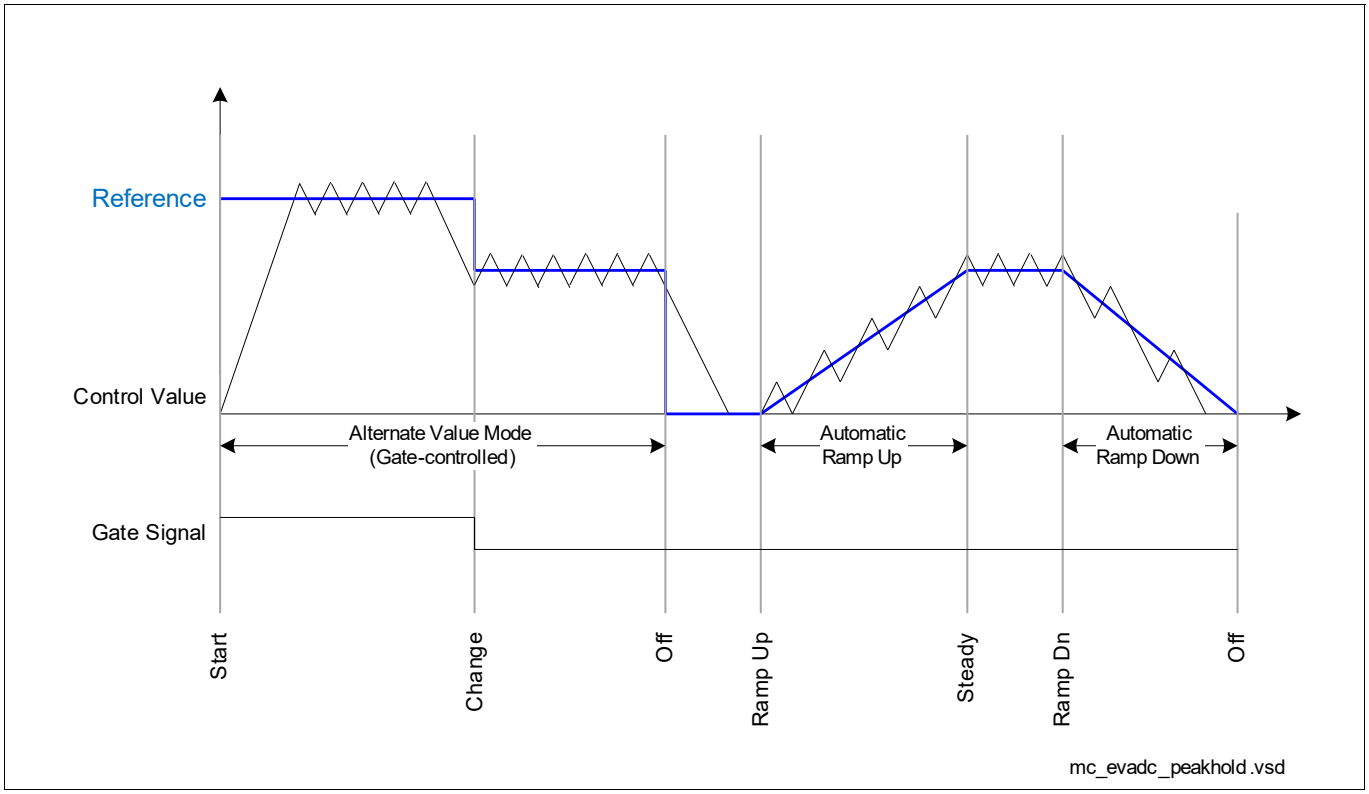


Figure 257 Peak-and-Hold Modes

32.8.2 Fast Compare Channel Control Registers

The following set of registers provides initialization and control of the operation of the fast compare channels.

Note: Only change the configuration while the respective channel is stopped.

Fast Compare Control Register, FC Channel x

The control register of the fast compare channel selects the external gate and/or trigger signals. Write control bits allow separate control of each function with a simple write access.

FCxFCCTRL (x=0-7)

Fast Compare Control Register, FC Channel x(3400_H+x*100_H) **Application Reset Value: $0000\ 0C20_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XTWC	FCCHNR					GTMODE		XTPOL	XTMODE		XTLVL	XTSEL			
w	rw					rw		rw	rw		rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPWC	DIVA					CHEVMODE		AIPF		RPE	STCF				
w	rw					rw		rw		rw	rw				

Field	Bits	Type	Description
STCF	4:0	rw	**Sample Time Control for Fast Comparisons** Number of additional clock cycles to be added to the minimum sample phase of 2 analog clock cycles: Coding and resulting sample time see **Table 255**.
RPE	5	rw	**Reference Precharge Enable** *Note: Enabled after reset.* 0_B No reference precharge Only use V_{AREF}/V_{AGND} for the conversion. 1_B Precharge enabled Use V_{DDM}/V_{SSM} for precharging and V_{AREF}/V_{AGND} for the final adjustment during a conversion. Precharge the reference input for 1 clock phase.
AIPF	7:6	rw	**Analog Input Precharge Enable for Fast Comparisons** The buffer is enabled automatically while AIPF ≠ 00_B. 00_B No precharge 01_B Precharge for 8 clock cycles 10_B Precharge for 16 clock cycles 11_B Precharge for 32 clock cycles

Field	Bits	Type	Description
CHEVMODE	9:8	rw	**Channel Event Mode** Generate a channel event: For a service request summary, refer to **"Service Requests for Fast Compare Channels" on Page 143**. 00_B Never 01_B If result becomes high (above compare value) 10_B If result becomes low (below compare value) 11_B If result switches to either level
DIVA	14:10	rw	**Divider Factor for the Analog Internal Clock** Defines the frequency of the analog converter clock f_{ADCI} (base clock for conversion steps), derived from the peripheral clock: $f_{ADCI} = f_{ADC}$ / CP. 00_H CP = 2 (max. frequency) 01_H CP = 2 02_H CP = 3 … $1F_H$ CP = 32
CPWC	15	w	**Write Control for Control Parameters** 0_B No write access to control parameters 1_B Bitfields DIVA, CHEVMODE, AIPF, RPC, STCF can be written
XTSEL	19:16	rw	**External Trigger Input Selection** The connected trigger input signals are listed in the product-specific appendix. *Note: The selected input signal can be used as a trigger source or as a gate signal, depending on the operating mode.*
XTLVL	20	rh	**External Trigger Level** Current level of the selected trigger input
XTMODE	22:21	rw	**Trigger Operating Mode** 00_B No external trigger 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B Trigger event upon any edge
XTPOL	23	rw	**External Trigger Polarity** 0_B Use selected input signal directly 1_B Invert selected input signal
GTMODE	25:24	rw	**Gate Operating Mode** 00_B No gate function 01_B Alternate value mode (see **Section 32.8.1**) 10_B Lock boundary flag (see **Section 32.8.4**) 11_B Reserved

Field	Bits	Type	Description
FCCHNR	30:26	rw	**Fast Compare Channel: Cannel Number** Selects the input signal to be compared: $1C_H$…$1F_H$ select internal signals (used for testing), all other channel numbers are reserved. See “Common Input Signals” in the product-specific appendix. 00_H Channel 0 is the one available input pin
XTWC	31	w	**Write Control for Trigger/Gate Configuration** 0_B No write access to trigger configuration 1_B Bitfields FCCHNR, GTMODE, XTPOL, XTMODE, XTSEL can be written

Table 256 Access Mode Restrictions of FCxFCCTRL (x=0-7) sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **CPWC**	rw	AIPF, CHEVMODE, DIVA, STCF	Set CPWC during write access
write 1 to **XTWC**	rw	FCCHNR, GTMODE, XTMODE, XTPOL, XTSEL	Set XTWC during write access
(default)	r	AIPF, CHEVMODE, DIVA, FCCHNR, GTMODE, STCF, XTMODE, XTPOL, XTSEL	

Fast Compare Mode Register, FC Channel x

The Fast Compare Mode Register configures the operating mode of a fast compare channel.

Note: Software can only write to bitfield FCREF while bitfield AUE = 00_B.

FCxFCM (x=0-7)

Fast Compare Mode Register, FC Channel x (3404_H+x*100_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FCREF										FCMWC	SSE	AUE		SRG	
rwh										w	rw	rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FCTRIV								ACSD		ANON	FCRDIR	RUNRAMP		RUNCOMP	
rw								rw		rw	rw	rw		rw	

Field	Bits	Type	Description
RUNCOMP	1:0	rw	**Run Control for Compare Channel** Defines the basic run conditions of the fast compare channel. 00_B Stop, no operation 01_B Always run 10_B Reserved 11_B Reserved

Field	Bits	Type	Description
RUNRAMP	3:2	rw	**Run Control for Ramp** Defines the run conditions for the ramp generation. *Note: Before changing the operating mode, stop the ramp timer, i.e. RUNRAMP = 00_B. REQTRx is the selected trigger signal.* 00_B Stop, no operation 01_B Start immediately when GxFCRAMP0 is written 10_B Start upon the selected trigger event of signal REQTRx 11_B Start immediately when GxFCRAMP0 is written and stop upon the selected trigger event of signal REQTRx
FCRDIR	4	rw	**Fast Compare Ramp Direction** 0_B Ramp down decrement ramp counter and stop when <counter> <= FCRCOMPB 1_B Ramp up increment ramp counter and stop when <counter> >= FCRCOMPB
ANON	5	rw	**Analog Converter Control** *Note: The extended wakeup time is required before the analog part is fully operable (see **"Analog Converter Control" on Page 24**).* 0_B Analog converter off 1_B Normal operation
ACSD	7:6	rw	**Analog Clock Synchronization Delay** Defines the delay of the analog clock in clocks after the sync signal. *Note: Do not exceed the current DIVA setting for the clock delay. Valid only if the phase synchronizer is selected (USC = 0_B)* 00_B 0, no delay 01_B 1 clock cycles delay … 11_B 3 clock cycles delay
FCTRIV	15:8	rw	**Fast Compare Trigger Interval** Defines the interval at which fast compare operations are triggered in steps of 16 × $1/f_{ADC}$. 00_H Interval is 16 clock cycles 01_H Interval is 32 clock cycles … FF_H Interval is 4096 clock cycles

Field	Bits	Type	Description
SRG	17:16	rw	**Service Request Generation** For a service request summary, refer to **"Service Requests for Fast Compare Channels" on Page 143**. 00_B Off, no service requests are generated 01_B Ramp end, issue service request when the ramp counter stops (either by reaching FCRCOMPB or by RUNRAMP becoming 00_B) 10_B New value, issue service request when a value is written to FCREF 11_B New result available
AUE	19:18	rw	**Automatic Update Enable** Defines the source of the value(s) in bitfield FCREF. 00_B No automatic update value(s) written by software. 01_B Alternate value While gate is active (high): value copied from bitfield FCRCOMPA While gate is inactive (low): value copied from bitfield FCRCOMPB 10_B Ramp counter value(s) copied from ramp counter on ramp start or counter update. 11_B Analog source value(s) written by the associated converter (see product-specific appendix).
SSE	20	rw	**Sample Synchronization Enable** *Note: See section "Synchronous Sampling" on Page 113.* 0_B No synchronization 1_B Sample timing is synchronized Recommended for operation of several ADCs.
FCMWC	21	w	**Write Control for Fast Compare Mode Configuration** 0_B No write access to FCM configuration 1_B Bitfields SSE, AUE, SRG, FCTRIV, ACSD, ANON, FCRDIR, RUNRAMP, RUNCOMP can be written
FCREF	31:22	rwh	**Fast Compare Reference Value** The input level is compared to this value. The resulting reference level is V_{AREF} / 1 024 × <FCREF>.

Table 257 Access Mode Restrictions of FCxFCM (x=0-7) sorted by descending priority

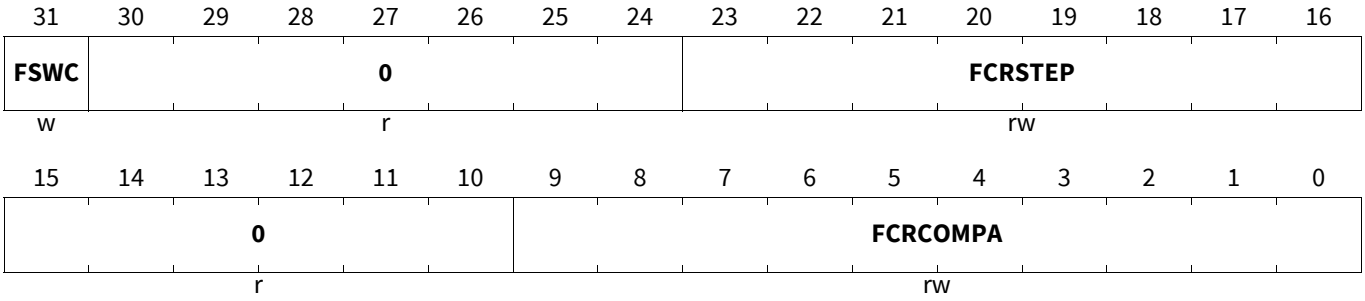
Mode Name	Access Mode		Description
write 1 to **FCMWC**	rw	ACSD, ANON, AUE, FCRDIR, FCTRIV, RUNCOMP, RUNRAMP, SRG, SSE	Set FCMWC during write access
.AUE=0x0	rwh	FCREF	
(default)	r	ACSD, ANON, AUE, FCRDIR, FCTRIV, RUNCOMP, RUNRAMP, SRG, SSE	
	rh	FCREF	

Fast Compare Ramp Register 0, FC Channel x

The Fast Compare Ramp Registers define a ramp (start level, end level) for the compare value. This can be used to realize soft start/stop functions or adaptive thresholds.

FCxFCRAMP0 (x=0-7)

Fast Compare Ramp Register 0, FC Channel x(3408_H+x*100_H) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FCRCOMPA	9:0	rw	**Fast Compare Ramp Compare Value A** The content of FCRCOMPA is copied to the ramp counter when a ramp is started, i.e. upon the selected trigger event. The ramp counter defines the reference value during ramp generation. It is, therefore, copied to bitfield FCREF when the ramp is started and each time the counter changes. FCRCOMPA is also used in alternate value mode while the gate is active (high).
FCRSTEP	23:16	rw	**Fast Compare Ramp Step Width** Configures the prescaler for FCRCOUNT in increments of $8 \times 1/f_{ADC}$. 00_H Stepwidth is 8 clock cycles 01_H Stepwidth is 16 clock cycles … FF_H Stepwidth is 2048 clock cycles
FSWC	31	w	**Write Control for Fast Compare Stepwidth** 0_B No write access to stepwidth 1_B Bitfield FCRSTEP can be written
0	15:10, 30:24	r	**Reserved, write 0, read as 0**

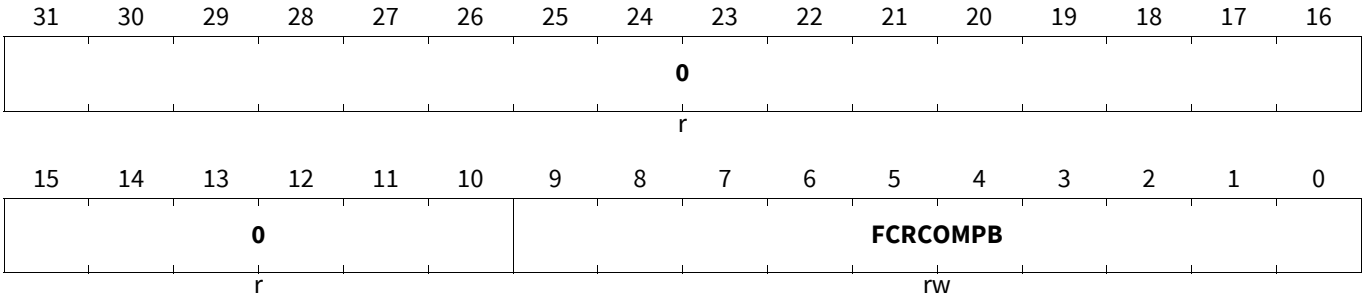
Table 258 Access Mode Restrictions of FCxFCRAMP0 (x=0-7) sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **FSWC**	rw	FCRSTEP	Set FSWC during write access
(default)	r	FCRSTEP	

Fast Compare Ramp Register 1, FC Channel x

FCxFCRAMP1 (x=0-7)

Fast Compare Ramp Register 1, FC Channel x($340C_H$+x*100_H) **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
FCRCOMPB	9:0	rw	**Fast Compare Ramp Compare Value B** Defines the stop level of the generated ramp. FCRCOMPB is also used in alternate value mode while the gate is inactive (low).
0	31:10	r	**Reserved, write 0, read as 0**

32.8.3 Boundary Definition

The actual boundaries are defined by the reference value **FCxFCM (x=0-7)**.FCREF and the positive and negative delta values defining the hysteresis band.

Bitfields DELTAMINUS and DELTAPLUS in register **FCxFCHYST (x=0-7)** store the delta values.

The actual used compare value depends on the current result value FCR (see **FCxFCBFL (x=0-7)**):

- FCR = 0: Reference value + upper delta (FCREF + DELTAPLUS)
- FCR = 1: Reference value - lower delta (FCREF - DELTAMINUS)

Note: The resulting compare values are saturated to fit into the fixed result range (000_H … $3FF_H$).

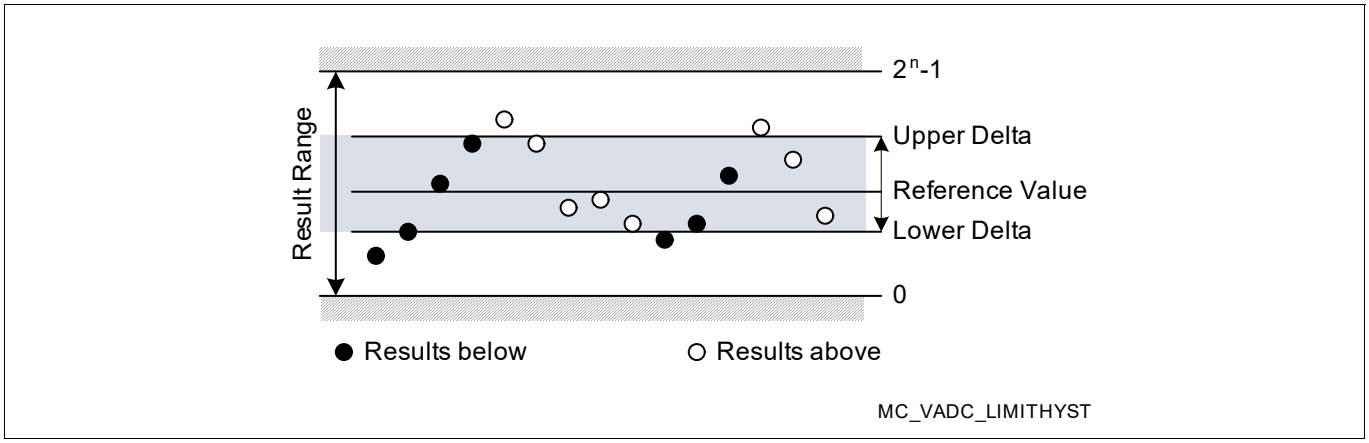


Figure 258 Result Monitoring through Compare with Hysteresis

Fast Comp. Hysteresis Register, FC Channel x

The upper and lower delta limits for Fast Compare operation are defined in the fast compare hysteresis register.

FCxFCHYST (x=0-7)

Fast Comp. Hysteresis Register, FC Channel x(3424_H+x*100_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				DELTAPLUS										0	
r				rw										r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				DELTAMINUS										0	
r				rw										r	

Field	Bits	Type	Description
DELTAMINUS	11:2	rw	**Lower Delta Value** This value is subtracted from the reference value while the last result is 1
DELTAPLUS	27:18	rw	**Upper Delta Value** This value is added to the reference value while the last result is 0
0	1:0, 17:12, 31:28	r	**Reserved, write 0, read as 0**

32.8.4 Boundary Flag Control

The limit checking mechanism can be configured to automatically control the boundary flags. These boundary flags are also available as control signals for other modules. The flags can be set or cleared when the defined level is exceeded and the polarity of the output signal can be selected. A gate signal can be selected to enable the boundary flag operation while the gate is active.

A boundary flag reflects the result of the comparisons with an upper or lower boundary. The basic compare value is provided in bitfield FCxFCM.FCREF, the delta values in register FCxFCHYST define a hysteresis band around the compare value.

A boundary flag is modified by several events:

- when a compare result is generated, the boundary flag is cleared or set as configured by bit FCxFCBFL.BFA, if FCxFCBFL.BFM = 1
- when software writes to bitfield FCxFCBFL.BFS, the boundary flag is cleared, set or toggled
- when the channel is disabled (ANON = 0), the result flag FCR is cleared, the boundary flag is cleared or set as configured by bit FCxFCBFL.BFA, if FCxFCBFL.BFM = 1

Each boundary flag is available at group-specific output lines. Node pointers additionally route them to one of four common boundary signals or one of the associated common service request lines, see bitfield BFLNP.

Note: Clear register GxFCHYST (i.e. the deltas) if a hysteresis is not wanted.

Boundary flags can reflect the result of compare operations, or can be forced to a configurable level (bit BFV) while the corresponding request source gate signal is inactive (low). See bitfield GTMODE in register **FCxFCCTRL (x=0-7)**.

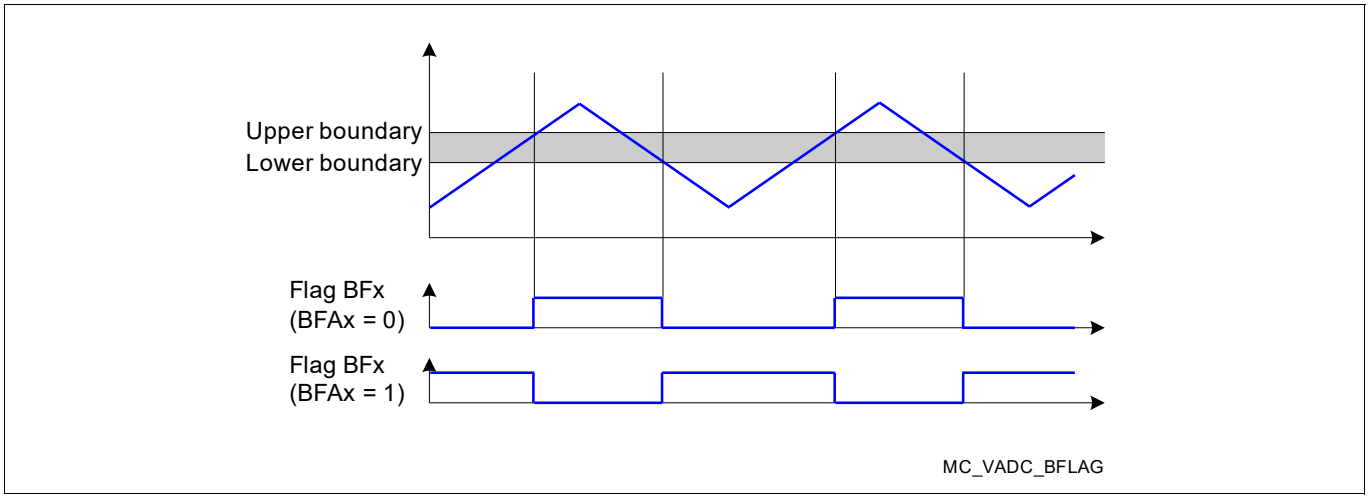


Figure 259 Boundary Flag Switching

The output signal derived from a boundary flag can be controlled by the GTM. A select input and a data input are available to temporarily replace the boundary flag signal before sending it to the output pin (see **Figure 260**).

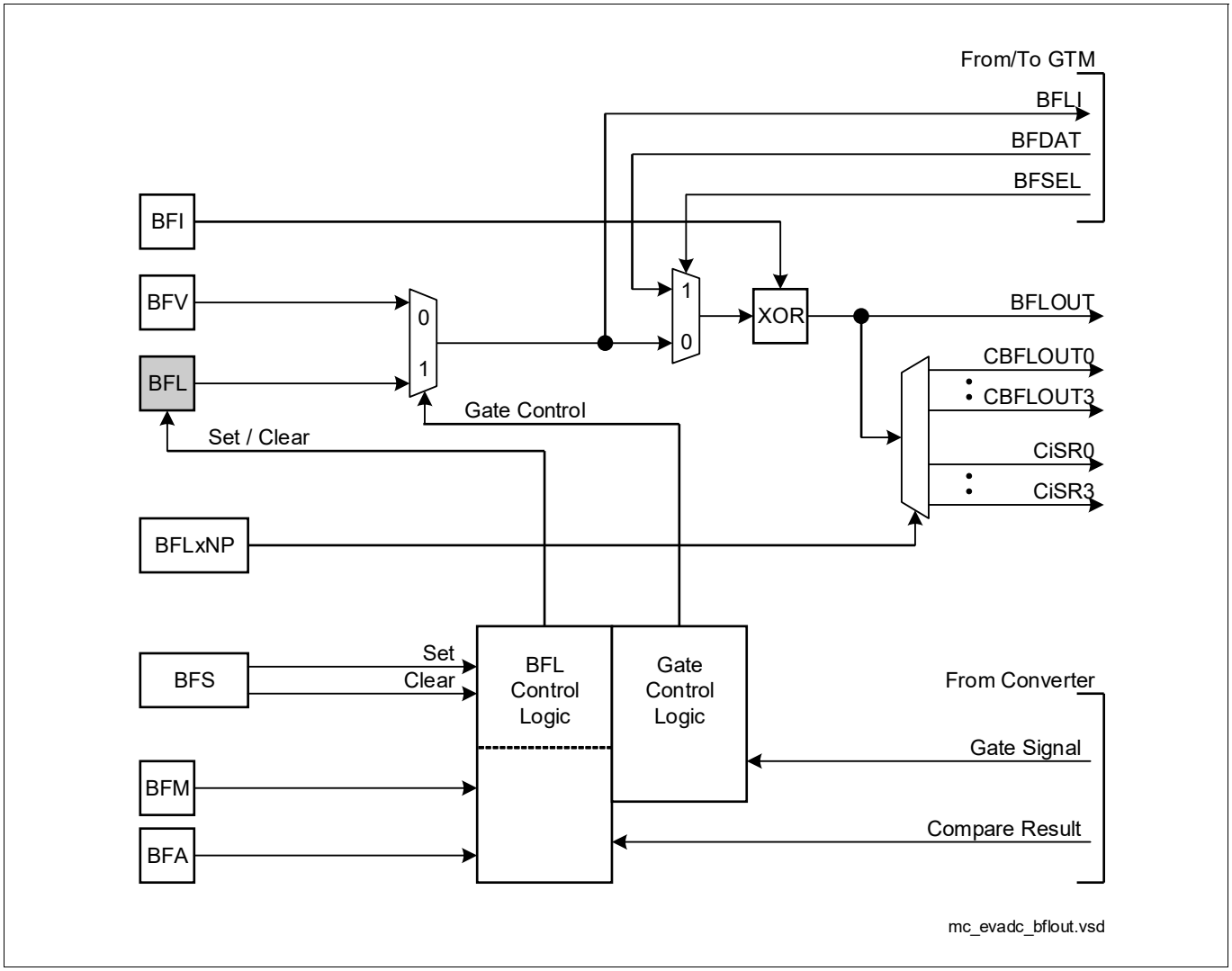


Figure 260 Boundary Flag Control

Software Interaction

Although in an application a fast compare channel will most probably control external hardware directly via the boundary flag outputs, also software interaction is possible, for example for debugging purposes at low speed.

The availability of a new compare result in bit FCxFCBFL.FCR is indicated by setting the valid flag FCxFCBFL.VF. Reading register FCxFCBFL automatically clears the valid flag. Due to internal synchronization this takes several clock cycles.

Polling the valid flag is only recommended for low speed operation, because

- due to the delayed clearing of bit VF a set valid flag may be read several times
- compare operations at a high rate may set a cleared valid flag before software can detect VF = 0

The availability of a new compare result in bit FCxFCBFL.FCR can also be indicated by generating a service request. Select FCxFCM.SRG = 11_B and FCxFCCTRL.CHEVMODE ≠ 00 (depending on the intended operation) to enable the service request path. This can be useful for low speed operation because short trigger intervals generate a high service request rate for the system.

Boundary Flag Register, FC Channel x

The Boundary Flag Register holds the boundary flag itself together with bits to select activation condition, output signal polarity, and basic operating mode.

The node pointer directs signal FCxBFLOUT to alternate on-chip connections with other modules (in addition to the group-specific output). Possible targets are the corresponding common service request lines or the common boundary flag outputs (CBFLOUT0 … CBFLOUT3).

Also software can set or clear the boundary flag.

FCxFCBFL (x=0-7)

Boundary Flag Register, FC Channel x **(3420_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VF	**0**		**FCR**	**BFLNP**				**0**						**BFV**	**BFM**
rh	r		rh	rw				r						rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**BFS**		**0**			**BFI**	**0**			**BFA**	**0**			**BFL**
r		w		r			rw	r			rw	r			rh

Field	Bits	Type	Description
BFL	0	rh	**Boundary Flag** 0_B Passive state result has not yet crossed the activation boundary (see bitfield BFA), or selected gate signal is inactive, or this boundary flag is disabled 1_B Active state result has crossed the activation boundary
BFA	4	rw	**Boundary Flag Activation Select** 0_B Set boundary flag BFL if new result FCR = 1 (input gets above the defined band or compare value), clear if FCR = 0 (input below) 1_B Set boundary flag BFL if new result FCR = 0 (input gets below the defined band or compare value), clear if FCR = 1 (input above)
BFI	8	rw	**Boundary Flag Inversion Control** 0_B Use BFL directly 1_B Use inverted BFL
BFS	13:12	w	**Boundary Flag Software Control** 00_B No action 01_B Clear BFL 10_B Set BFL 11_B Toggle BFL
BFM	16	rw	**Boundary Flag Mode Control** 0_B Disable boundary flag, BFL is not changed by FCR 1_B Enable boundary flag (see also bitfield GTMODE in register CTRL)

Field	Bits	Type	Description
BFV	17	rw	**Boundary Flag Value** Defines the logic value that replaces the compare result while the gate input is inactive (low) in lock mode (see also bitfield GTMODE in register FCxFCCTRL)
BFLNP	27:24	rw	**Boundary Flag Node Pointer** *Note: For shared service request lines see common groups in the product-specific appendix.* Not listed combinations are reserved. 0_H Select common boundary flag output 0 … 3_H Select common boundary flag output 3 4_H Select shared service request line 0 … 7_H Select shared service request line 3 F_H Disabled, no common output signal
FCR	28	rh	**Fast Compare Result** Indicates the last generated compare result *Note: If the input signal equals the reference value the analog comparator will return either "above" or "below".* 0_B Signal level below reference value 1_B Signal level above reference value
VF	31	rh	**Valid Flag** Indicates a new result in bit FCR. *Note: Bit VF is automatically cleared upon reading register FCxFCBFL.* 0_B No new result available 1_B Bit FCR has been updated with new value and has not yet been read
0	3:1, 7:5, 11:9, 15:14, 23:18, 30:29	r	**Reserved, write 0, read as 0**

32.9 Conversion Timing

The term “conversion timing” not only covers the time required for the conversion to a digital result itself, but also the time required for sampling the respective input signal, the time required for calibration, and additional internal steps.

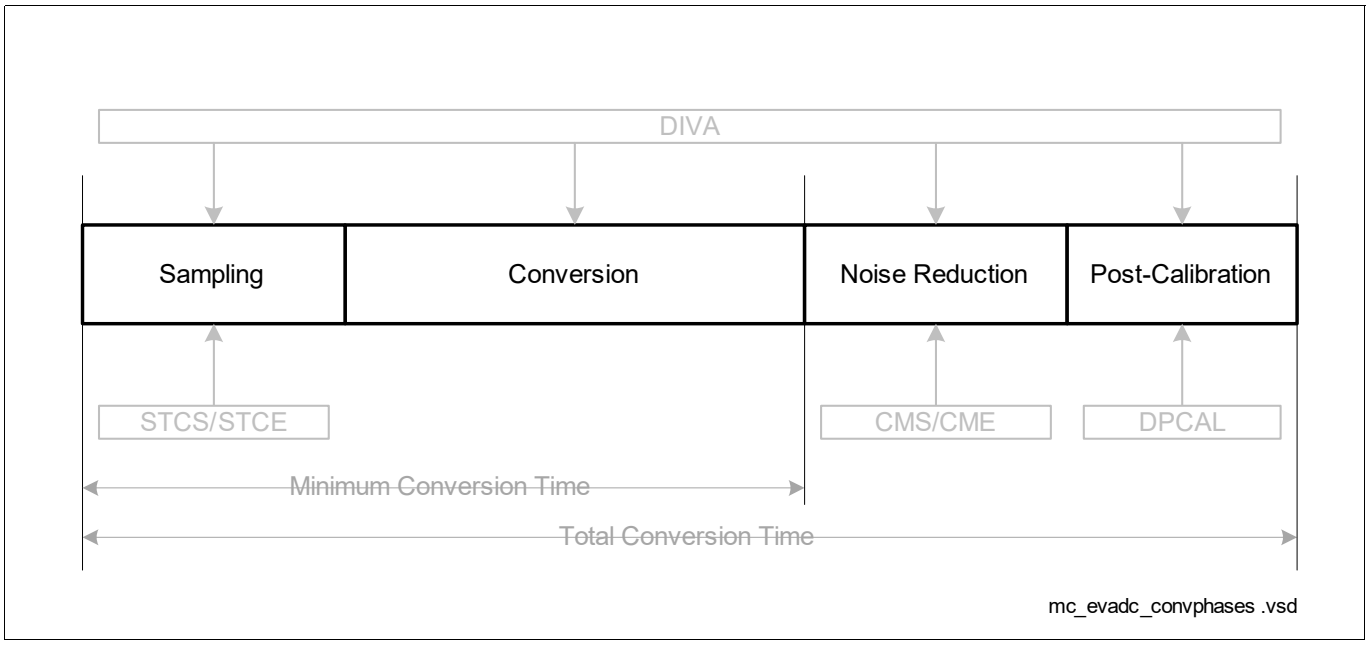


Figure 261 Conversion Phases

All phases of a conversion are configurable by the application. Consequently, the conversion time also depends on the respective configuration. The sections below list the configurable factors and also give some examples for the resulting conversion times.

Note: The time from the trigger event that requests the corresponding conversion until the start of the sample phase depends on the arbitration and can, therefore, only be determined with a known system setup.

32.9.1 Start-Up CalibrationTiming

Start-up calibration is required before operating the EVADC channels. The calibration time can be computed with the following formula:

$t_{SUCAL} = 256 \times ((4 + 2{\times}CALSTC) \times t_{ADCI} + 5 \times t_{ADC})$ [1)]

Timing Examples Start-Up Calibration

System assumptions:f_{ADC} = 160 MHz i.e. t_{ADC} = 6.25 ns, DIVA = 5, f_{ADCI} = 26.67 MHz i.e. t_{ADCI} = 37.5 ns, CALSTC = 00_B.

According to the given formula the following calibration time is required:

$t_{SUCAL} = 256 \times (4 \times t_{ADCI} + 5 \times t_{ADC}) = 256 \times (4 \times 37.5\ ns + 5 \times 6.25\ ns) = 46.4\ \mu s$

Note: The start-up calibration is not part of the actual conversions. Also, the corresponding calibration time does not contribute to the conversion timing.
It is, however, a prerequisite for accurate conversion results.

1) For f_{ADCI} > 40 MHz CALSTC must be 01_B.

32.9.2 Standard Converter Channels Timing

The total conversion time comprises the time from the start of the sample phase until the availability of the result.

The frequency at which conversions can be triggered also depends on several configurable factors:

- The selected sample time, according to the input class definitions. For conversions using an external multiplexer, also the extended sample times count.
- Delays induced by cancelled conversions that must be repeated.
- Delays due to equidistant sampling of other channels.
- The frequency of external trigger signals, if enabled.
- The period of the request timer, if enabled,

The conversion timing depends on the following user-definable factors:

- The ADC conversion clock frequency, where $f_{ADCI} = f_{ADC} / (DIVA+1)$
- The selected sample time, where $t_S = (2 + STC) \times t_{ADCI}$ (STC = additional sample time, see also **Table 255**)[1)]
- The actual result generation (12 bits)
- The duration of the selected noise reduction conversion steps NRS × t_{NR}, where NRS can be 0, 1, 3, 7
- The post-calibration time t_{PC}, if enabled
- Internal steps done at module clock speed

The conversion time is the sum of sample time, conversion steps, and internal steps. It can be computed with the following formula:

$t_{C12} = [(2 + STC) \times t_{ADCI}] + [13 \times t_{ADCI}] + [NRS \times t_{NR}] + [t_{PC}] + [3 \times t_{ADC}]$

with:

$t_{NR} = 4 \times t_{ADCI} + 3 \times t_{ADC}$ per conversion step

$t_{PC} = (4 + 2\times CALSTC) \times t_{ADCI} + 5 \times t_{ADC}$ if enabled, otherwise 0

Timing examples in **Table 259**.

32.9.3 Fast Compare Channels Timing

Fast Compare channels execute single compare operations instead of the complete SAR algorithm. The sample time is required in all cases.

The conversion timing depends on the following user-definable factors:

- The ADC conversion clock frequency, where $f_{ADCI} = f_{ADC} / (DIVA+1)$
- The selected sample time, where $t_S = (2 + STCF) \times t_{ADCI}$ (STCF = additional sample time, see also **Table 255**)
- The actual compare operation (1 bit)
- Internal steps done at module clock speed

The comparison time is the sum of sample time, compare step, and internal steps. It can be computed with the following formula:

$t_{FC} = [(2 + STCF) \times t_{ADCI}] + [2 \times t_{ADCI}] + [3 \times t_{ADC}]$

Timing examples in **Table 259**.

Note: To ensure equidistant results, make sure the programmed request period is longer than the maximum compare time.

1) $t_S = (2 + (STC-15)\times16)) \times t_{ADCI}$ for STC > $0F_H$.

32.9.4 Conversion Timing Configurations

The examples in the table below assume a minimum sample time of 100 ns as required for the primary groups. Longer sample times, required by secondary groups or by the properties of the external signal source, lead to increased total conversion times.

The maximum conversion rate can be achieved with postcalibration disabled. This can be done either temporarily or permanently, depending on the requirements of the application.

System assumptions:f_{ADC} = 160 MHz i.e. t_{ADC} = 6.25 ns, single MSB, CALSTC = 0 for $f_{ADCI} \leq$ 40 MHz, CALSTC = 1 for f_{ADCI} = 53.3 MHz

Table 259 Conversion Timing Overview

Element	20.0 MHz (DIVA = 7)	26.7 MHz (DIVA = 5)	40.0 MHz (DIVA = 3)	53.3 MHz (DIVA = 2)
Sample time = 100 ns	2 × 50 ns = 100 ns	3 × 37.5 ns = 112.5 ns	4 × 25 ns = 100 ns	6 × 18.75 ns = 112.5 ns
Sample time = 500 ns	10 × 50 ns = 500 ns	14 × 37.5 ns = 525 ns	34 × 25 ns = 850 ns	34 × 18.75 ns = 637.5 ns
Result generation	13 × 50 ns = 650 ns	13 × 37.5 ns = 487.5 ns	13 × 25 ns = 325 ns	13 × 18.75 ns = 243.75 ns
Postcalibration	4 × 50 ns = 200 ns	4 × 37.5 ns = 150 ns	4 × 25 ns = 100 ns	6 × 18.75 ns = 112.5 ns
Sync postcalibration	5 × 6.25 ns = 31.25 ns	5 × 6.25 ns = 31.25 ns	5 × 6.25 ns = 31.25 ns	5 × 6.25 ns = 31.25 ns
Sync statemachine	3 × 6.25 ns = 18.75 ns	3 × 6.25 ns = 18.75 ns	3 × 6.25 ns = 18.75 ns	3 × 6.25 ns = 18.75 ns
Noise reduction step (0, 1, 3, 7)	218.75 ns	168.75 ns	118.75 ns	93.75 ns
Conversion with postcalibration	1000 ns	800 ns	575 ns	518.75 ns
Conversion with 3 noise red. steps (CMS = 10_B) and postcalibration	1656.25 ns	1306.25 ns	931.25 ns	800 ns
Conversion without postcalibration, primary groups	768.75 ns	618.75 ns	443.75 ns	375 ns
Conversion without postcalibration, secondary groups	1168.75 ns	1031.25 ns	1193.75 ns	900 ns
Maximum conversion rate	1.3 MS/s	1.6 MS/s	2.2 MS/s	2.6 MS/s
Compare steps	2 × 50 ns = 100 ns	2 × 37.5 ns = 75 ns	2 × 25 ns = 50 ns	2 × 18.75 ns = 37.5 ns
Fast compare operation	218.75 ns	206.25 ns	168.75 ns	168.75 ns
Maximum fast compare rate	4.5 MS/s	4.8 MS/s	5.9 MS/s	5.9 MS/s

32.10 Conversion Result Handling

The A/D converters can preprocess the conversions result data to a certain extent before storing them for retrieval by the CPU or a DMA channel. This supports the subsequent handling of result data by the application software.

Conversion result handling comprises the following functions:

- **Storage of Conversion Results** to user-configurable registers
- **Data Alignment** according to endianess
- **Wait-for-Read Mode** to avoid loss of data
- **Result FIFO Buffer** to store subsequent result values
- **Result Event Generation**
- **Data Modification** supports data reduction or anti-aliasing filtering
- **Hardware Data Interface** provides result values to other modules

32.10.1 Storage of Conversion Results

The conversion result values of a certain group can be stored in one of the 16 associated group result registers or in the common global result register (see **Selecting a Result Register**).

This structure provides different locations for the conversion results of different sets of channels. Depending on the application needs (data reduction, auto-scan, alias feature, etc.), the user can distribute the conversion results to minimize CPU load and/or optimize the performance of DMA transfers.

The compare result values of the fast compare channels is stored in bit FCR of register **FCxFCBFL (x=0-7)**.

Each result register has an individual data valid flag (VF) associated with it. This flag indicates when “new” valid data has been stored in the corresponding result register or bit and can be read out.

Note: The valid flag indicates results generated by the converter. The standard result registers can also be written by software. In this case, after writing to a result register the valid flag is undefined.

For standard conversions, result values are available in bitfield RESULT.

Result registers can be read via two different views. These views use different addresses but access the same register data:

- When a result register is read via the **application view**, the corresponding valid flag is automatically cleared when the result is read. This provides an easy handshake between result generation and retrieval. This also supports wait-for-read mode.
- When a result register is read via the **debug view**, the corresponding valid flag remains unchanged when the result is read. This supports debugging by delivering the result value without disturbing the handshake with the application.

The application can retrieve conversion results through several result registers:

- Group result register:
  Returns the result value and the channel number
- Global result register:
  Returns the result value and the channel number and the group number

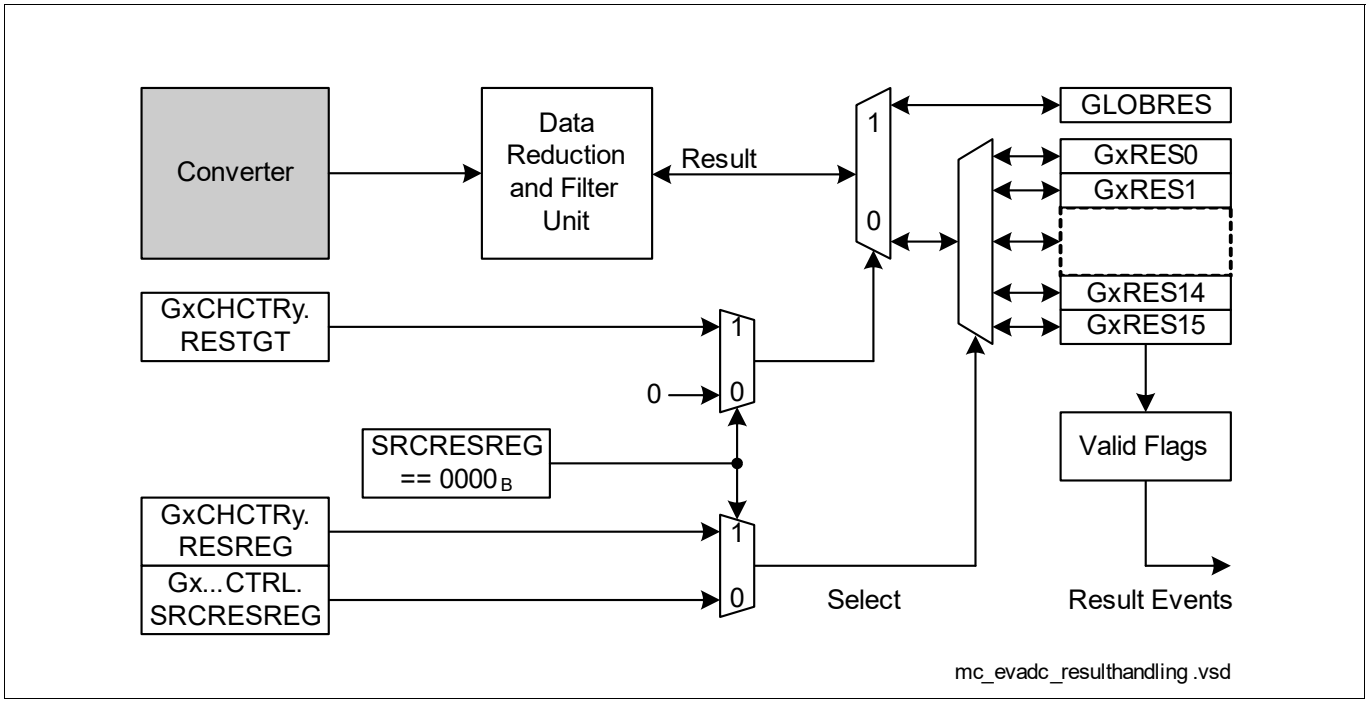


Figure 262 Conversion Result Storage

Selecting a Result Register

Conversion results are stored in result registers that can be assigned by the user according to the requirements of the application. The following bitfields direct the results to a register:

- SRCRESREG in register **GxQCTRLi (i=0-2;x=0-11)** etc.
  Selects the group-specific result register GxRES1 … GxRES15
  when source-specific result registers are used
- RESTGT in register GxCHCTRy
  Selects the global result register instead of the group-specific result registers
- RESREG in register GxCHCTRy
  Selects the group-specific result register GxRES0 … GxRES15
  when channel-specific result registers are used (see below).

Using source-specific result registers allows separating results from the same channel that are requested by different request sources. Usually these request sources are used by different tasks and are triggered at different times.

Using the global result register allows convenient handling of results from different groups, e.g. in the case of daisy chaining. This way, a single DMA channel can unload all results.
When using GLOBRES as the target, make sure only one result at a time is written to it, so no result values get lost.

Note: Bitfield SRCRESREG is disregarded for synchronized conversions triggered by another master. In this case no request source is involved and the selection is taken from the channel control register.

Group x Result Control Register y

The group result control registers select the behavior of the result registers of a given group.

GxRCRy (x=0-11;y=0-15)

Group x Result Control Register y **(0680_H+x*400_H+y*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRGEN	0				FEN		WFR	0		DMM		DRCTR			
rw	r				rw		rw	r		rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

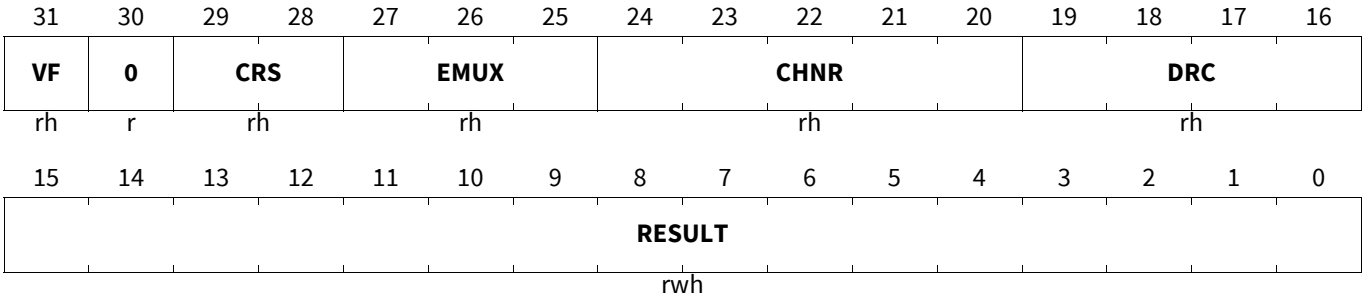
Field	Bits	Type	Description
DRCTR	19:16	rw	**Data Reduction Control** Defines how result values are stored/accumulated in this register for the final result. The data reduction counter DRC can be loaded from this bitfield. The function of bitfield DRCTR is determined by bitfield DMM.
DMM	21:20	rw	**Data Modification Mode** See **"Data Modification" on Page 101** 00_B Standard data reduction (accumulation) 01_B Result filtering mode The filter registers are cleared while bitfield DMM ≠ 01_B. 10_B Difference mode 11_B Reserved
WFR	24	rw	**Wait-for-Read Mode Enable** 0_B Overwrite mode 1_B Wait-for-read mode enabled for this register
FEN	26:25	rw	**FIFO Mode Enable** 00_B Separate result register 01_B Part of a FIFO structure: copy each new valid result 10_B Maximum mode: copy new result if bigger 11_B Minimum mode: copy new result if smaller
SRGEN	31	rw	**Service Request Generation Enable** 0_B No service request 1_B Service request after a result event
0	15:0, 23:22, 30:27	r	**Reserved, write 0, read as 0**

Group x Result Register y

The group result registers provide a selectable storage location for all channels of a given group.

Note: The content of result register GxRES15 is available on the hardware data interface.

GxRESy (x=0-11;y=0-15)
Group x Result Register y **(0700_H+x*400_H+y*4)** **Application Reset Value: 0000 0000_H**



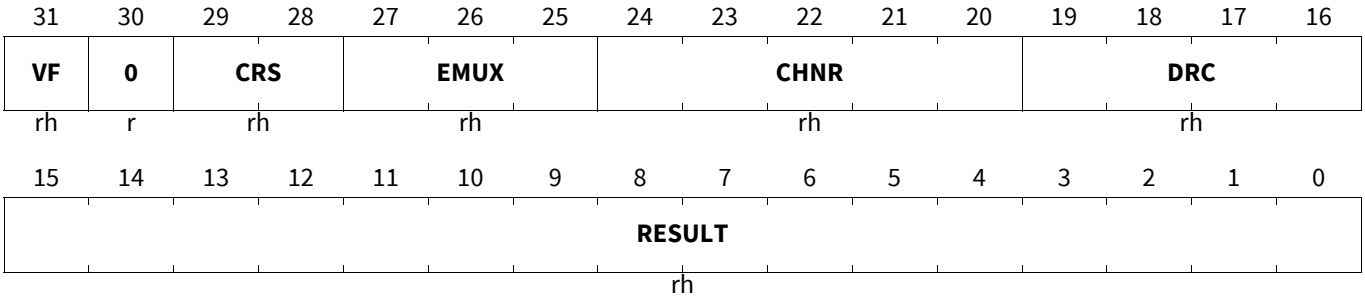
Field	Bits	Type	Description
RESULT	15:0	rwh	**Result of Most Recent Conversion** The position of the result bits within this bitfield depends on the configured operating mode. Refer to **Section 32.10.2**. Bitfield RESULT is writeable by the application to set the initial value for min/max detection.
DRC	19:16	rh	**Data Reduction Counter** Indicates the number of values still to be accumulated for the final result. The final result is available and valid flag VF is set when bitfield DRC becomes zero (by decrementing or by reload). See **"Data Modification" on Page 101**
CHNR	24:20	rh	**Channel Number** Indicates the channel number corresponding to the value in bitfield RESULT.
EMUX	27:25	rh	**External Multiplexer Setting** Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT. *Note: Available in GxRES0 only. Use GxRES0 if EMUX information is required.*
CRS	29:28	rh	**Converted Request Source** Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs. 00_B Request source 0 01_B Request source 1 10_B Request source 2 11_B Synchronized conversion

Field	Bits	Type	Description
VF	31	rh	**Valid Flag** Indicates a new result in bitfield RESULT. *Note: Bit VF is automatically cleared upon reading register GxRESy.* 0_B No new result available 1_B Bitfield RESULT has been updated with new result value and has not yet been read
0	30	r	**Reserved, write 0, read as 0**

Group x Result Reg. y, Debug

The debug view of the group result registers provides access to all result registers of a given group, however, without clearing the valid flag.

GxRESDy (x=0-11;y=0-15)
Group x Result Reg. y, Debug **(0780_H+x*400_H+y*4)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RESULT	15:0	rh	**Result of Most Recent Conversion** The position of the result bits within this bitfield depends on the configured operating mode. Refer to **Section 32.10.2**.
DRC	19:16	rh	**Data Reduction Counter** Indicates the number of values still to be accumulated for the final result. The final result is available and valid flag VF is set when bitfield DRC becomes zero (by decrementing or by reload). See **"Data Modification" on Page 101**
CHNR	24:20	rh	**Channel Number** Indicates the channel number corresponding to the value in bitfield RESULT.
EMUX	27:25	rh	**External Multiplexer Setting** Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT. *Note: Available in GxRESD0 only. Use GxRESD0 if EMUX information is required.*
CRS	29:28	rh	**Converted Request Source** Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs. 00_B Request source 0 01_B Request source 1 10_B Request source 2 11_B Synchronized conversion
VF	31	rh	**Valid Flag** Indicates a new result in bitfield RESULT. 0_B No new result available 1_B Bitfield RESULT has been updated with new result value and has not yet been read (via GxRESy)
0	30	r	**Reserved, write 0, read as 0**

Global Result Control Register

The global result control register selects the behavior of the global result register.

GLOBRCR

Global Result Control Register **(0280_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRGEN	0						WFR	0				DRCTR			
rw	r						rw	r				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
DRCTR	19:16	rw	**Data Reduction Control** Defines how result values are stored/accumulated in this register for the final result. The data reduction counter DRC can be loaded from this bitfield. Coding see **"Function of Bitfield DRCTR" on Page 101**[1)] 0_H Data reduction disabled
WFR	24	rw	**Wait-for-Read Mode Enable** Refer also to "Wait-for-Read Mode". 0_B Overwrite mode 1_B Wait-for-read mode enabled for this register ###
SRGEN	31	rw	**Service Request Generation Enable** 0_B No service request 1_B Service request after a result event
0	15:0, 23:20, 30:25	r	**Reserved, write 0, read as 0**

1) Only standard data reduction is available for the global result register, i.e. DMM is assumed as 00_B.

Global Result Register

The global result register provides a common storage location for all channels of all groups.

GLOBRES

Global Result Register **(0300_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VF	0	CRS		EMUX			CHNR					GNR			
rwh	r	rh		rh			rh					rh			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
rwh															

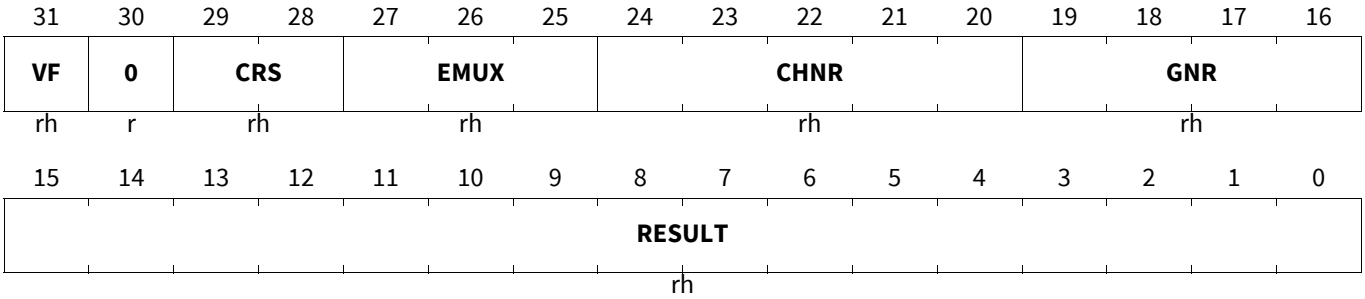
Field	Bits	Type	Description
RESULT	15:0	rwh	**Result of most recent conversion** The position of the result bits within this bitfield depends on the configured operating mode. Refer to **Section 32.10.2**.
GNR	19:16	rh	**Group Number** Indicates the group to which the channel number in bitfield CHNR refers.
CHNR	24:20	rh	**Channel Number** Indicates the channel number corresponding to the value in bitfield RESULT.
EMUX	27:25	rh	**External Multiplexer Setting** Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT.
CRS	29:28	rh	**Converted Request Source** Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs.
VF	31	rwh	**Valid Flag** Indicates a new result in bitfield RESULT. 0_B No new valid data available Write access: No effect 1_B Bitfield RESULT contains valid data and has not yet been read Write access: Clear this valid flag and the data reduction counter (overrides a hardware set action)
0	30	r	**Reserved, write 0, read as 0**

Global Result Register, Debug

The debug view of the global result register provides access to the global result register,however, without clearing the valid flag.

GLOBRESD

Global Result Register, Debug **(0380_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RESULT	15:0	rh	**Result of most recent conversion** The position of the result bits within this bitfield depends on the configured operating mode. Refer to **Section 32.10.2**.
GNR	19:16	rh	**Group Number** Indicates the group to which the channel number in bitfield CHNR refers.

Field	Bits	Type	Description
CHNR	24:20	rh	**Channel Number** Indicates the channel number corresponding to the value in bitfield RESULT.
EMUX	27:25	rh	**External Multiplexer Setting** Indicates the setting of the external multiplexer, corresponding to the value in bitfield RESULT.
CRS	29:28	rh	**Converted Request Source** Indicates the request source that as requested the conversion to which the result value in bitfield RESULT belongs.
VF	31	rh	**Valid Flag** Indicates a new result in bitfield RESULT. 0_B No new valid data available 1_B Bitfield RESULT contains valid data and has not yet been read
0	30	r	**Reserved, write 0, read as 0**

Valid Flag Register, Group x

The valid flag register summarizes the valid flags of all result registers.

Clear valid flags by writing a 1 to the respective bit position.

GxVFR (x=0-11)

Valid Flag Register, Group x **($05F8_H$+x*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VF15	**VF14**	**VF13**	**VF12**	**VF11**	**VF10**	**VF9**	**VF8**	**VF7**	**VF6**	**VF5**	**VF4**	**VF3**	**VF2**	**VF1**	**VF0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
VFy (y=0-15)	y	rwh	**Valid Flag of Result Register x** Indicates a new result in bitfield RESULT. 0_B No new valid data available Write access: No effect 1_B Result register x contains valid data and has not yet been read Write access: Clear this valid flag and bitfield DRC in register GxRESy (overrides a hardware set action).
0	31:16	r	**Reserved, write 0, read as 0**

32.10.2 Data Alignment

The number of valid bits of a conversion result value within the selected result register depends on the configuration (see **Figure 263**):

- The selected result position (Left/Right-aligned[1)], for standard conversion results)
- The selected data modification mode (accumulation, difference, filter: always right-aligned, RESPOS = 0!)

These options provide the conversion results in a way that minimizes data handling for the application software.

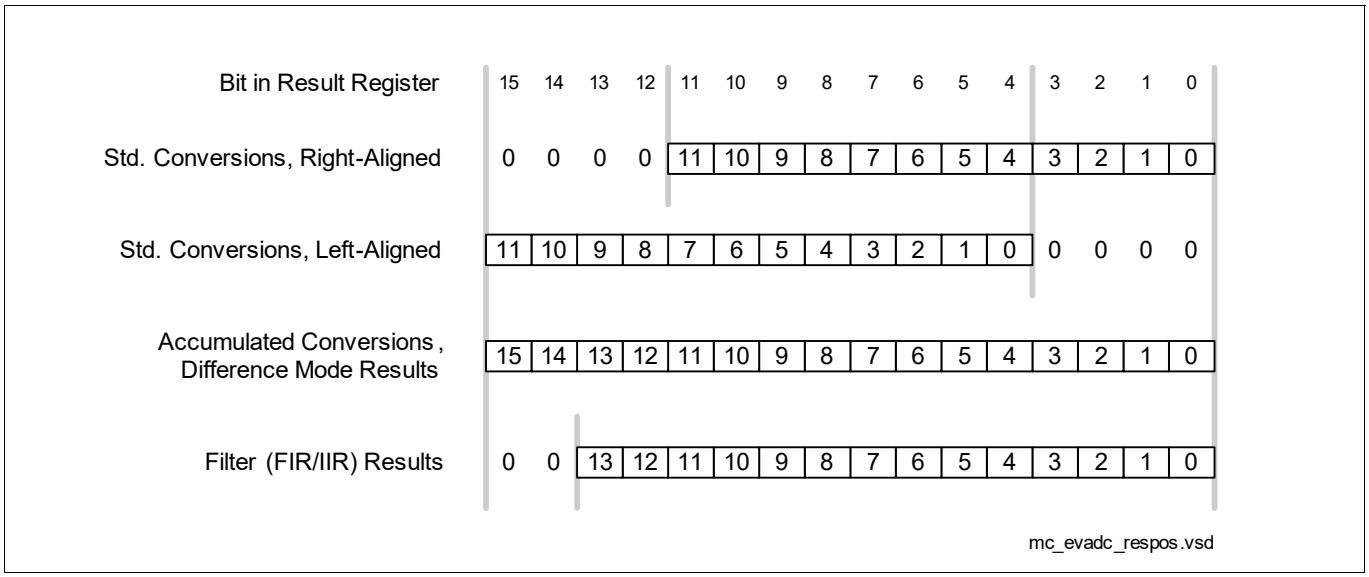


Figure 263 Result Storage Options

32.10.3 Wait-for-Read Mode

The wait-for-read mode prevents data loss due to overwriting a result register with a new conversion result before the CPU (or DMA) has read the previous data. For example, auto-scan conversion sequences or other sequences with “relaxed” timing requirements may use a common result register. However, the results come from different input channels, so an overwrite would destroy the result from the previous conversion[2)].

Wait-for-read mode automatically suspends the start of a conversion for this channel from this source until the current result has been read. So a conversion or a conversion sequence can be requested by a hardware or software trigger, while each conversion is only started after the result of the previous one has been read. This automatically aligns the conversion sequence with the CPU/DMA capability to read the formerly converted result (latency).

If wait-for-read mode is enabled for a result register (bit GxRCRy.WFR = 1), a request source does not generate a conversion request while the targeted result register contains valid data (indicated by the valid flag VF = 1) or if a currently running conversion targets the same result register.

If two request sources target the same result register with wait-for-read mode selected, a higher priority source cannot interrupt a lower priority conversion request started before the higher priority source has requested its conversion. Cancel-inject-repeat mode does not work in this case. If the higher priority request targets a different result register, the lower priority conversion can be cancelled and repeated afterwards.

1) RESPOS = 1 left-shifts the read result value by four bits. This, therefore, only makes sense for standard 12-bit conversion results. The global result register GLOBRES always stores results right-aligned.

2) Repeated conversions of a single channel that use a separate result register will not destroy other results, but rather update their own previous result value. This way, always the actual signal data is available in the result register.

If request sources from different groups target the global result register with wait-for-read mode selected (bit GLOBRCR.WFR = 1), a conversion request can only be suppressed when GLOBRES.VF is already set. Concurrent requests may lead to conflicts, because several request sources may see the global valid flag cleared during the begin of the sequence.

To avoid this, ensure that these conversions are only requested sequentially, as it is the case for daisy chaining.

*Note: Wait-for-read mode is ignored for synchronized conversions of synchronization slaves (see **Section 32.11**).*

32.10.4 Result FIFO Buffer

Result registers can either be used as direct target for conversion results or they can be concatenated with their upper neighbor result register of the same ADC group to form a result FIFO buffer (first-in-first-out buffer mechanism). A result FIFO stores several measurement results that can be read out later with a “relaxed” CPU response timing. It is possible to set up more than one FIFO buffer structure with the available result registers.

Result FIFO structures of two or more registers are built by concatenating result registers to their upper neighbor result register (with next higher index, see **Figure 264**). This is enabled by setting bitfield GxRCRy.FEN = 01_B. Conversion results are stored to the register with the highest index of a FIFO structure (GxRCRy.FEN = 00_B). Software reads the values from the FIFO register with the lowest index.

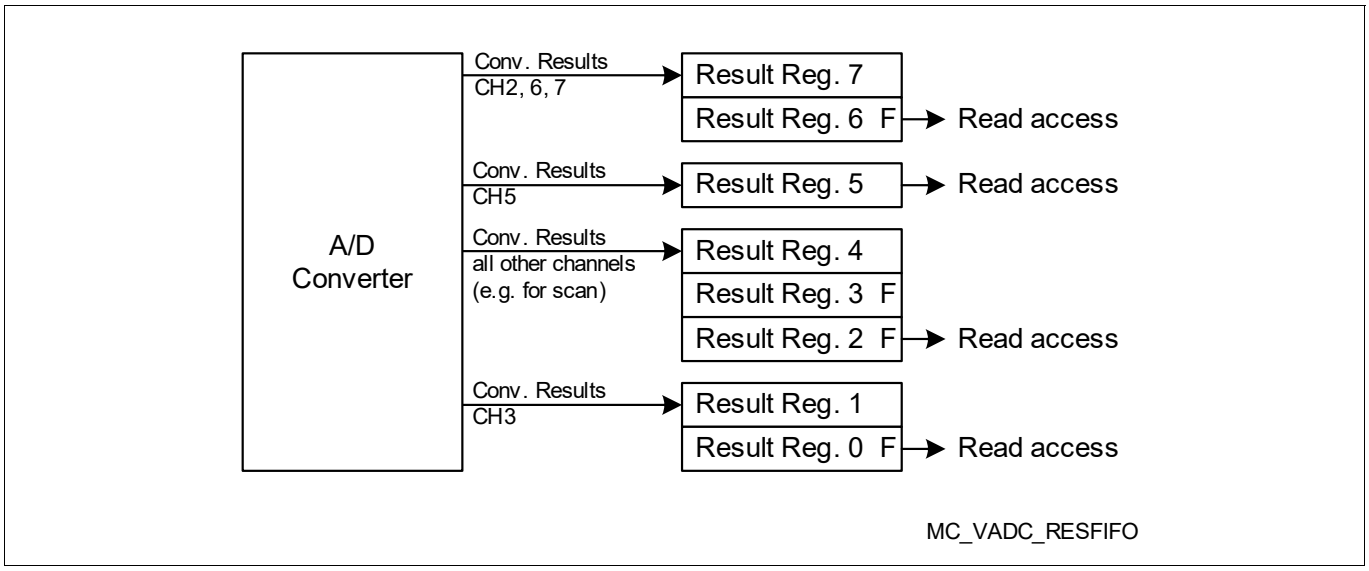


Figure 264 Result FIFO Buffers

In the example shown the result registers have been configured in the following way:

- 2-stage buffer consisting of result registers 7-6
- dedicated result register 5
- 3-stage buffer consisting of result registers 4-3-2
- 2-stage buffer consisting of result registers 1-0

Note: If GxRES15 is the input stage of a FIFO buffer, result values written to it are only reflected on the HDI if this value fills the FIFO completely. Intermediate results are not written to the HDI. If the HDI shall be used, select a lower result register as the input stage.

Table 260 summarizes the required configuration of result registers if they are combined to build result FIFO buffers.

Table 260 Properties of Result FIFO Registers

Function	Input Stage	Intermed. Stage	Output Stage
Result target[1)]	YES	no	no
Application read	no	no	YES
Data reduction mode	YES	no	no
Wait-for-read mode	YES	no	no
Result event interrupt	no	no	YES
Valid read data	no	no	YES
FIFO enable (FEN)	00_B	01_B	01_B
Registers in example	7, 4, 1	3	6, 2, 0

1) To enable proper functionality of a FIFO, direct results only to the input stage of a FIFO structure.

Note: If enabled, a result interrupt is generated for each data word in the FIFO.

Minimum/Maximum Detection

A special FIFO mode, that only updates the corresponding FIFO stage if the result was above (or below) the current value of the stage, stores the highest (or lowest) result of a conversion sequence. This is enabled by setting bitfield GxRCRy.FEN = 10_B (maximum) or GxRCRy.FEN = 11_B (minimum).

Note: Write a reasonable start value to the result bitfield in the peak result register before the conversion sequence to be monitored begins.
Use 0000_H to find the maximum and $0FFF_H$ to find the minimum (only the lower 12 bits are compared!).
The actual result value is available from the result register targetted by the respective channel.

Result FIFO buffer timing

To update the output of the result FIFO buffer, there is a latency of 6 system peripheral bus cycles (f_{SPB}) and 6 ADC clock cycles (f_{ADC}). This latency has to consider for the read-out by CPU or DMA:

- The data transfer from result FIFO buffer to the CPU is usually done with a consecutive procedure of single read commands. The architectural defined access time from CPU to system peripheral bus is given by 5 system peripheral bus cycles (f_{SPB}). Therefore, a waiting time between the consecutive reads of 1 f_{SPB} cycle and 6 f_{ADC} cycles has to consider.
- To ensure data integrity, it is mandatory to use a DMA configuration that considers the described FIFO timing. From DMA perspective, the configuration based on a single transfer and multiple data moves would be the preferred choice. However, this configuration does not fulfill the FIFO timing. For the initial data move, the execution time is defined by 6 system peripheral bus clock cycles (f_{SPB}) and 13 system resource interface clock cycles (f_{SRI}). All subsequent data moves require 3 f_{SPB} clock cycles and 11 (f_{SRI}) clock cycles. To use the result FIFO buffer together with the DMA, a Linked List DMA configuration could be used. Using this kind of configuration, it is ensured that the DMA latency ($6\times f_{SPB} + 13\times f_{SRI}$) is longer than the update of the result FIFO buffer needs ($6\times f_{SPB} + 6\times f_{ADC}$).

32.10.5 Data Modification

The data resulting from conversions can be automatically modified before being used by an application. Several options can be selected (bitfield DMM in register **GxRCRy (x=0-11;y=0-15)** etc.) which reduce the CPU/DMA load required to unload and/or process the conversion data.

- **Standard Data Reduction Mode (for GxRES0 ... GxRES15):**Accumulates 2 up to 16 result values within each result register before generating a result interrupt. This can remove noise from the input signal or preprocess the result data for the application.
- **Result Filtering Mode (FIR, for GxRES7, GxRES15):**Applies a 3rd order Finite Impulse Response Filter (FIR) with selectable coefficients to the conversion results for the selected result register.
- **Result Filtering Mode (IIR, for GxRES7, GxRES15):**Applies a 1st order Infinite Impulse Response Filter (IIR) with selectable coefficients to the conversion results for the selected result register.
- **Difference Mode (for GxRES1 ... GxRES15):**Subtracts the contents of result register GxRES0 from the conversion results for the selected result register. Bitfield DRCTR is not used in this mode.

Table 261 Function of Bitfield DRCTR

DRCTR	Standard Data Reduction Mode (DMM = 00_B)	DRCTR	Result Filtering Mode (DMM = 01_B)[1)]
0000_B	Data Reduction disabled	0000_B	FIR filter: a=2, b=1, c=0
0001_B	Accumulate 2 result values	0001_B	FIR filter: a=1, b=2, c=0
0010_B	Accumulate 3 result values	0010_B	FIR filter: a=2, b=0, c=1
0011_B	Accumulate 4 result values	0011_B	FIR filter: a=1, b=1, c=1
0100_B	Accumulate 5 result values	0100_B	FIR filter: a=1, b=0, c=2
0101_B	Accumulate 6 result values	0101_B	FIR filter: a=3, b=1, c=0
0110_B	Accumulate 7 result values	0110_B	FIR filter: a=2, b=2, c=0
0111_B	Accumulate 8 result values	0111_B	FIR filter: a=1, b=3, c=0
1000_B	Accumulate 9 result values	1000_B	FIR filter: a=3, b=0, c=1
1001_B	Accumulate 10 result values	1001_B	FIR filter: a=2, b=1, c=1
1010_B	Accumulate 11 result values	1010_B	FIR filter: a=1, b=2, c=1
1011_B	Accumulate 12 result values	1011_B	FIR filter: a=2, b=0, c=2
1100_B	Accumulate 13 result values	1100_B	FIR filter: a=1, b=1, c=2
1101_B	Accumulate 14 result values	1101_B	FIR filter: a=1, b=0, c=3
1110_B	Accumulate 15 result values	1110_B	IIR filter: a=2, b=2
1111_B	Accumulate 16 result values	1111_B	IIR filter: a=3, b=4

1) The filter registers are cleared while bitfield DMM ≠ 01_B.

Standard Data Reduction Mode

The data reduction mode can be used as digital filter for anti-aliasing or decimation purposes. It accumulates a maximum of 16 conversion values to generate a final result.

Each result register can be individually enabled for data reduction, controlled by bitfield DRCTR in registers GxRCRy and GLOBRCR. The data reduction counter DRC indicates the actual status of the accumulation.

Note: Conversions for other result registers can be inserted between conversions to be accumulated.

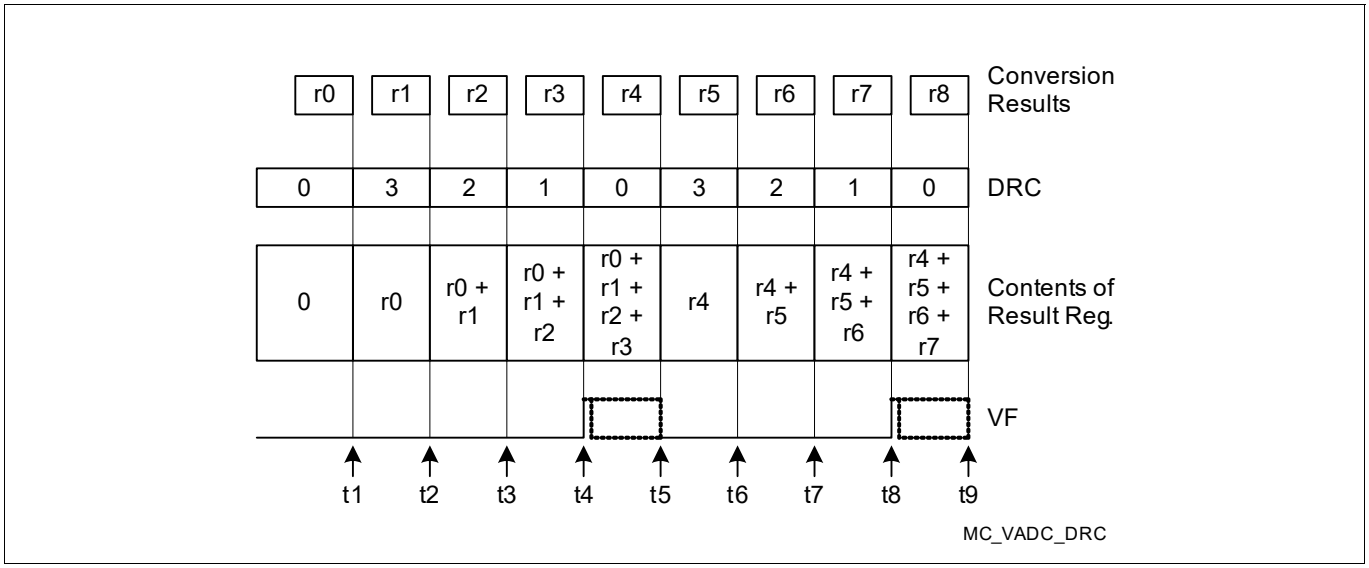


Figure 265 Standard Data Reduction Filter

This example shows a data reduction sequence of 4 accumulated conversion results. Eight conversion results (r0 … r7) are accumulated and produce 2 final results.

When a conversion is complete and stores data to a result register that has data reduction mode enabled, the data handling is controlled by the data reduction counter DRC:

- If DRC = 0 (t1, t5, t9 in the example), the conversion result is stored to the register. DRC is loaded with the contents of bitfield DRCTR (i.e. the accumulation begins).
- If DRC > 0 (t2, t3, t4 and t6, t7, t8 in the example), the conversion result is added to the value in the result register.
  DRC is decremented by 1.
- If DRC becomes 0, either decremented from 1 (t4 and t8 in the example) or loaded from DRCTR, the valid bit for the respective result register is set and a result register event occurs.
  The final result must be read before the next data reduction sequence starts (before t5 or t9 in the example). This automatically clears the valid flag.

*Note: Software can clear the data reduction counter DRC by clearing the corresponding valid Flag (via **GxVFR (x=0-11)**).*

The response time to read the final data reduction results can be increased by associating the adjacent result register to build a result FIFO (see **Figure 266**). In this case, the final result of a data reduction sequence is loaded to the adjacent register. The value can be read from this register until the next data reduction sequence is finished (t8 in the 2nd example).

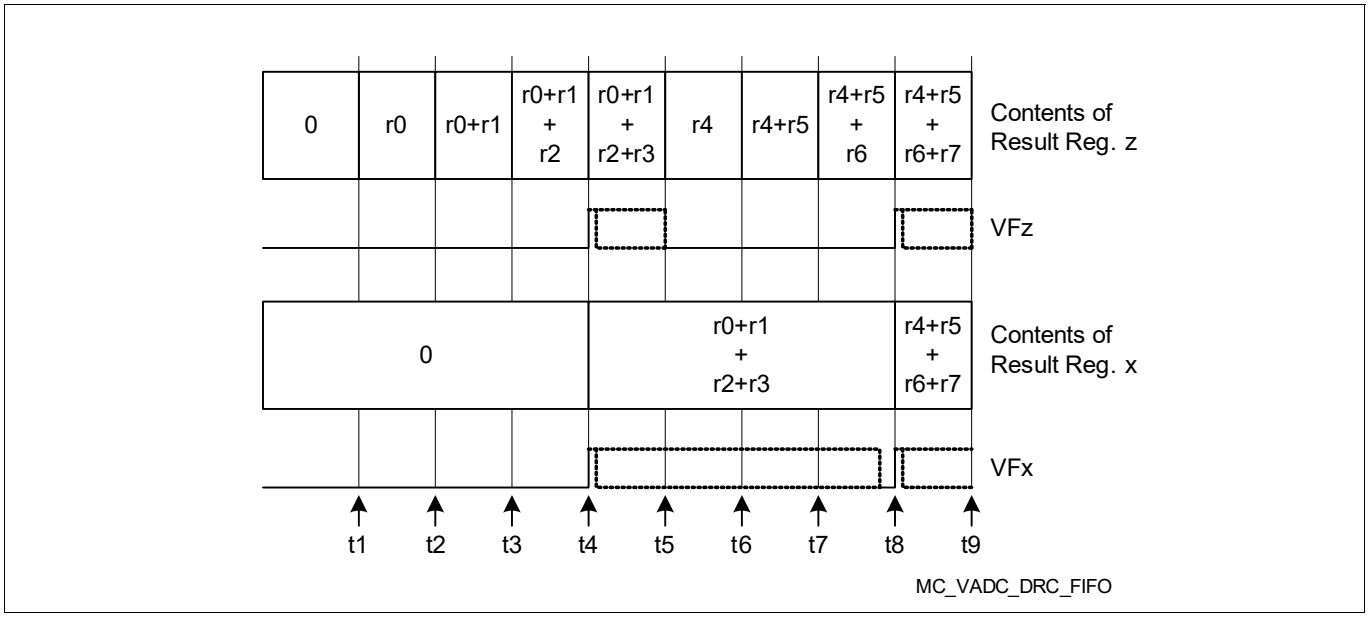


Figure 266 Standard Data Reduction Filter with FIFO Enabled

Finite Impulse Response Filter Mode (FIR)

The FIR filter (see **Figure 267**) provides 2 result buffers for intermediate results (RB1, RB2) and 3 configurable tap coefficients (a, b, c).

The conversion result and the intermediate result buffer values are added weighted with their respective coefficients to form the final value for the result register. Several predefined sets of coefficients can be selected via bitfield DRCTR (coding listed in **Table 261**) in registers GxRCR7 and GxRCR15. These coefficients lead to a gain of 3 or 4 to the ADC result producing a 14-bit value. The valid flag (VF) is activated for each sample after activation, i.e. for each sample generates a valid result.

Note: Conversions for other result registers can be inserted between conversions to be filtered.

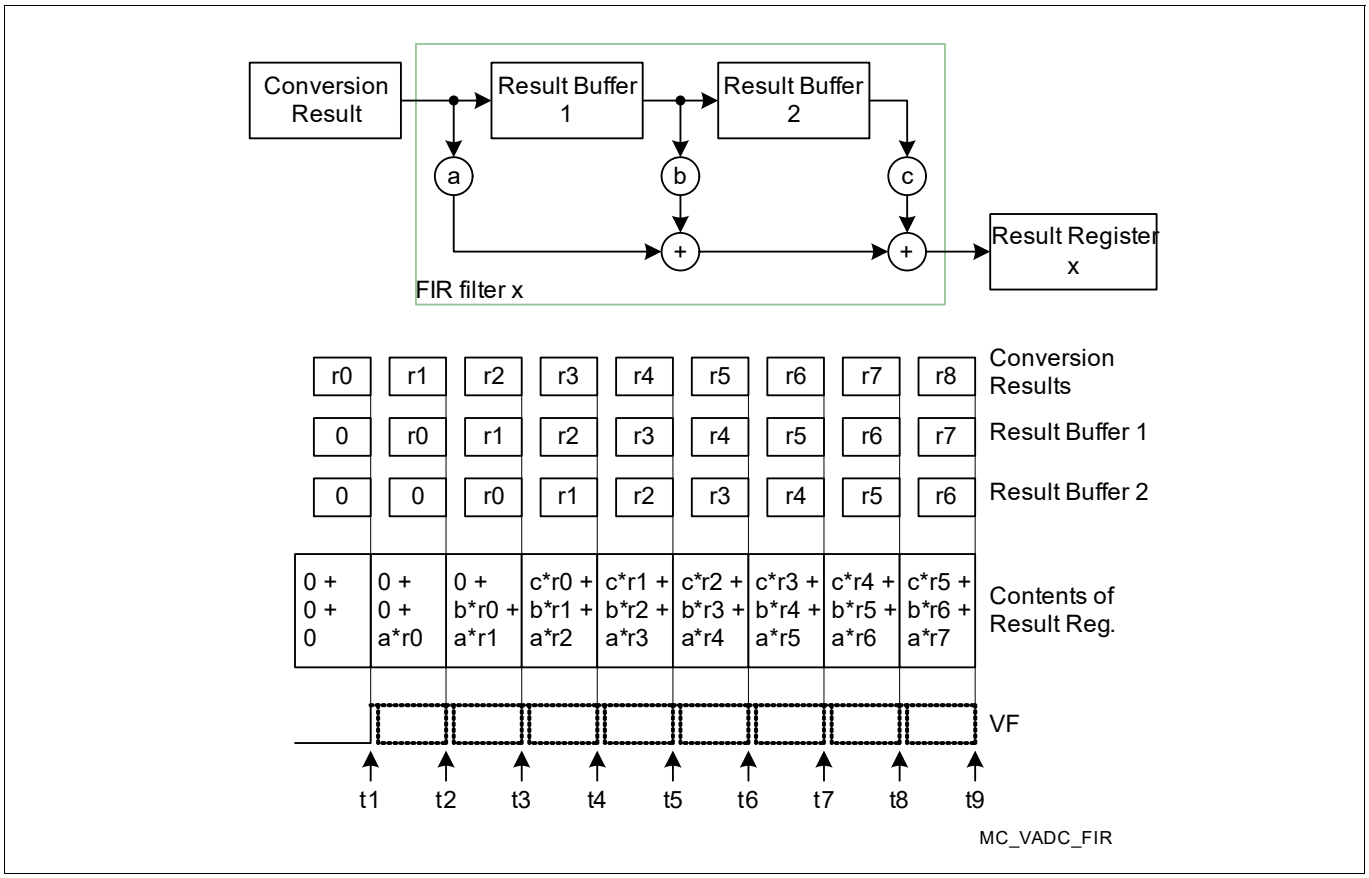


Figure 267 FIR Filter Structure

Note: The filter registers are cleared while bitfield DMM ≠ 01_B.

Infinite Impulse Response Filter Mode (IIR)

The IIR filter (see **Figure 268**) provides a result buffer (RB) and 2 configurable coefficients (a, b). It represents a first order low-pass filter.

The conversion result, weighted with the respective coefficient, and a fraction of the previous result are added to form the final value for the result register. Several predefined sets of coefficients can be selected via bitfield DRCTR (coding listed in **Table 261**) in registers GxRCR7 and GxRCR15. These coefficients lead to a gain of 4 to the ADC result producing a 14-bit value. The valid flag (VF) is activated for each sample after activation, i.e. for each sample generates a valid result.

Note: Conversions for other result registers can be inserted between conversions to be filtered.

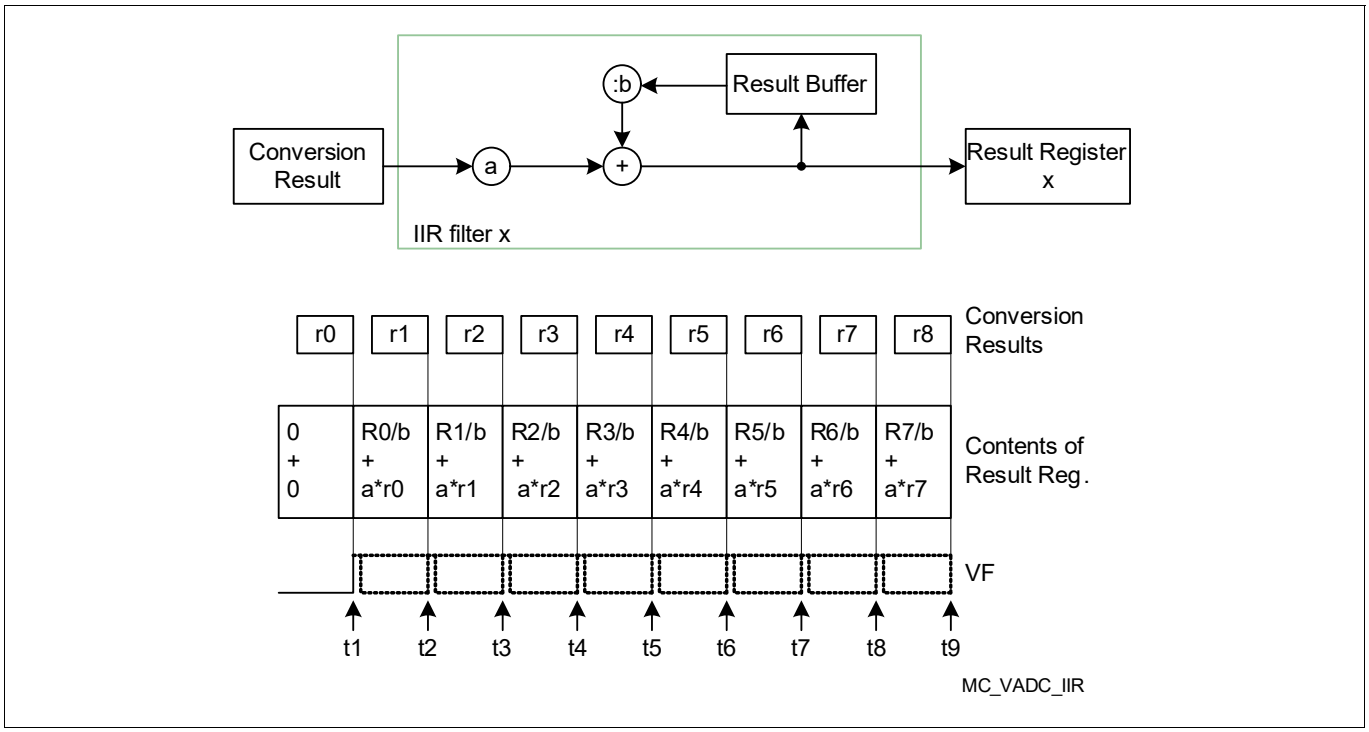


Figure 268 IIR Filter Structure

Note: The filter registers are cleared while bitfield DMM ≠ 01_B.

Difference Mode

Subtracting the contents of result register 0 from the actual result puts the results of the respective channel in relation to another signal. No software action is required.

The reference channel must store its result(s) into result register 0. The reference value can be determined once and then be used for a series of conversions, or it can be converted before each related conversion.

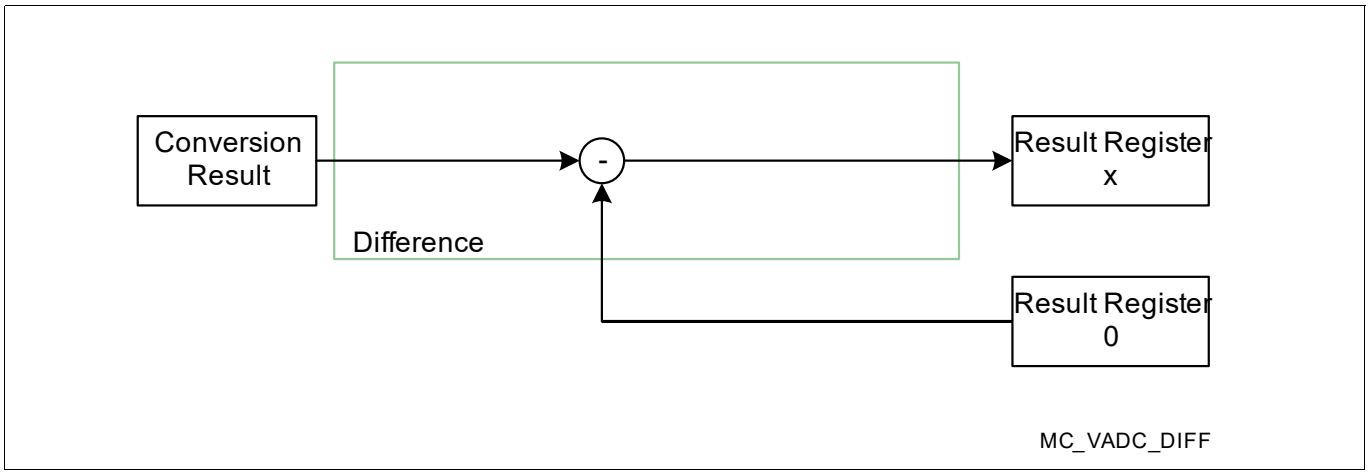


Figure 269 Result Difference

32.10.6 Result Event Generation

A result event can be generated when a new value is stored to a result register. Result events can be restricted due to data accumulation and be generated only if the accumulation is complete.

Result events can also be suppressed completely.

32.10.7 Hardware Data Interface

The digital conversion results are directly available to other modules via the hardware data interface (HDI). Result register GxRES15 outputs its content to this interface. Each time GxRES15 is updated, the HDI updates the data vector and generates a write strobe indicating the availability of a new result value.

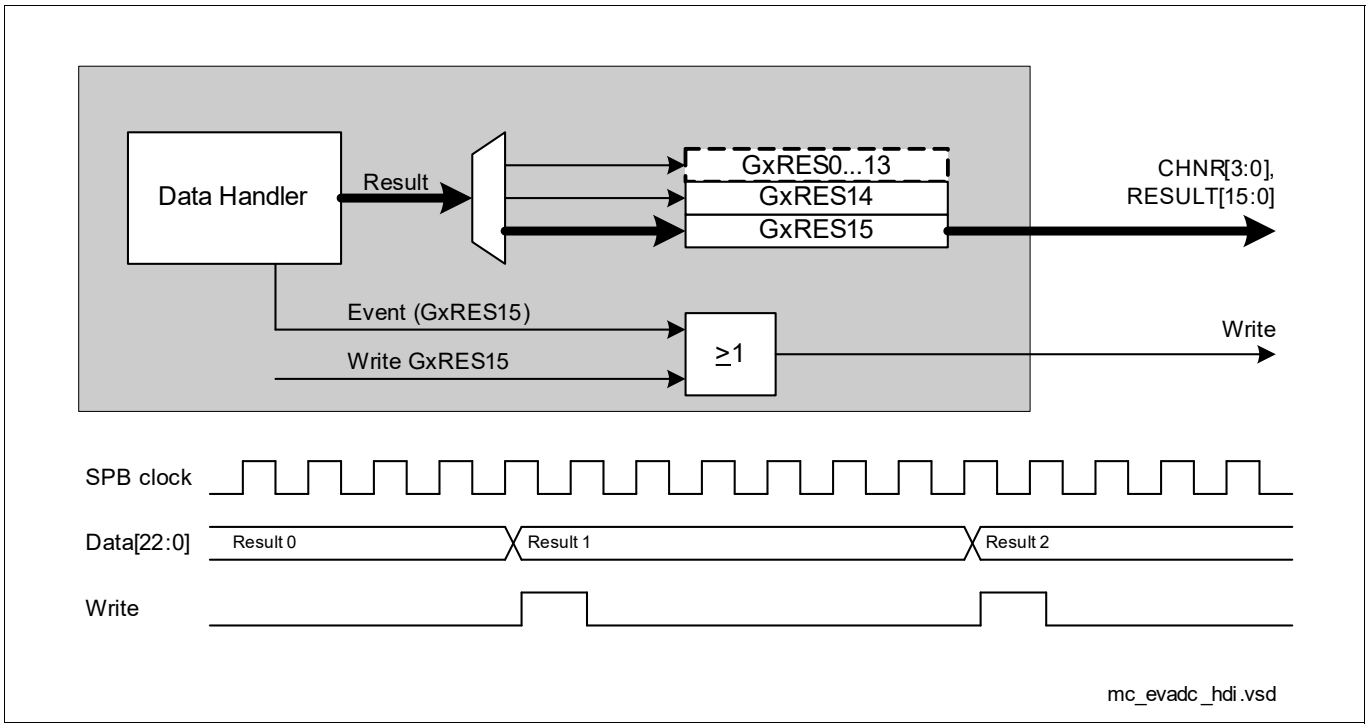


Figure 270 Hardware Data Interface

The data from this interface can be used for several purposes:

- Update the reference value of the associated Fast Compare Channel (see appendix)
- Write results to the SPU (signal processing unit) via the RIF (radar interface)
- Write results to the GTM to make them available for the MCSs equipped with an analog data interface (ADI)

Note: If GxRES15 is the input stage of a FIFO buffer, result values written to it are only reflected on the HDI if this value fills the FIFO completely. Intermediate results are not written to the HDI. If the HDI shall be used, select a lower result register as the input stage.

The hardware interface provides access to the result data of result register 15 of each group. The following data elements are available through the HDI:

Table 262 HDI Data Assignment

Bit Position	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Content				CHNR					Conversion Result (GxRES15.RESULT)															

Note: The number of valid CHNR bits depends on the respective multiplexer. Unused bits are 0.
When GxRES15 is updated via software, bits 20:16 of the HDI reflect the last value stored in bitfield GxRES15.CHNR, because bitfield GxRES15.CHNR is not writeable.

32.11 Synchronization of Conversions

The conversions of an ADC kernel can be scheduled either self-timed according to the kernel's configuration or triggered by external (outside the ADC) signals:

Synchronized conversions support parallel conversion of channels within a synchronization group[1]. This optimizes e.g. the control of electrical drives.

Equidistant sampling supports conversions in a fixed raster with minimum jitter. This optimizes e.g. filter algorithms or audio applications.

32.11.1 Synchronized Conversions for Parallel Sampling

Several independent ADC kernels[1] can be synchronized for simultaneous measurements of analog input channels. While no parallel conversion is requested, the kernels can work independently.

The synchronization mechanism for parallel conversions ensures that the sample phases of the related channels start simultaneously. Synchronized kernels convert the same channel that is requested by the master. Different values for the sample phase length of each kernel for a parallel conversion are supported.

A parallel conversion can be requested individually for each input channel (one or more). In the example shown in **Figure 271**, input channels CH3 of the ADC kernels 0, 1, 2 are converted synchronously, whereas other input channels do not lead to parallel conversions.

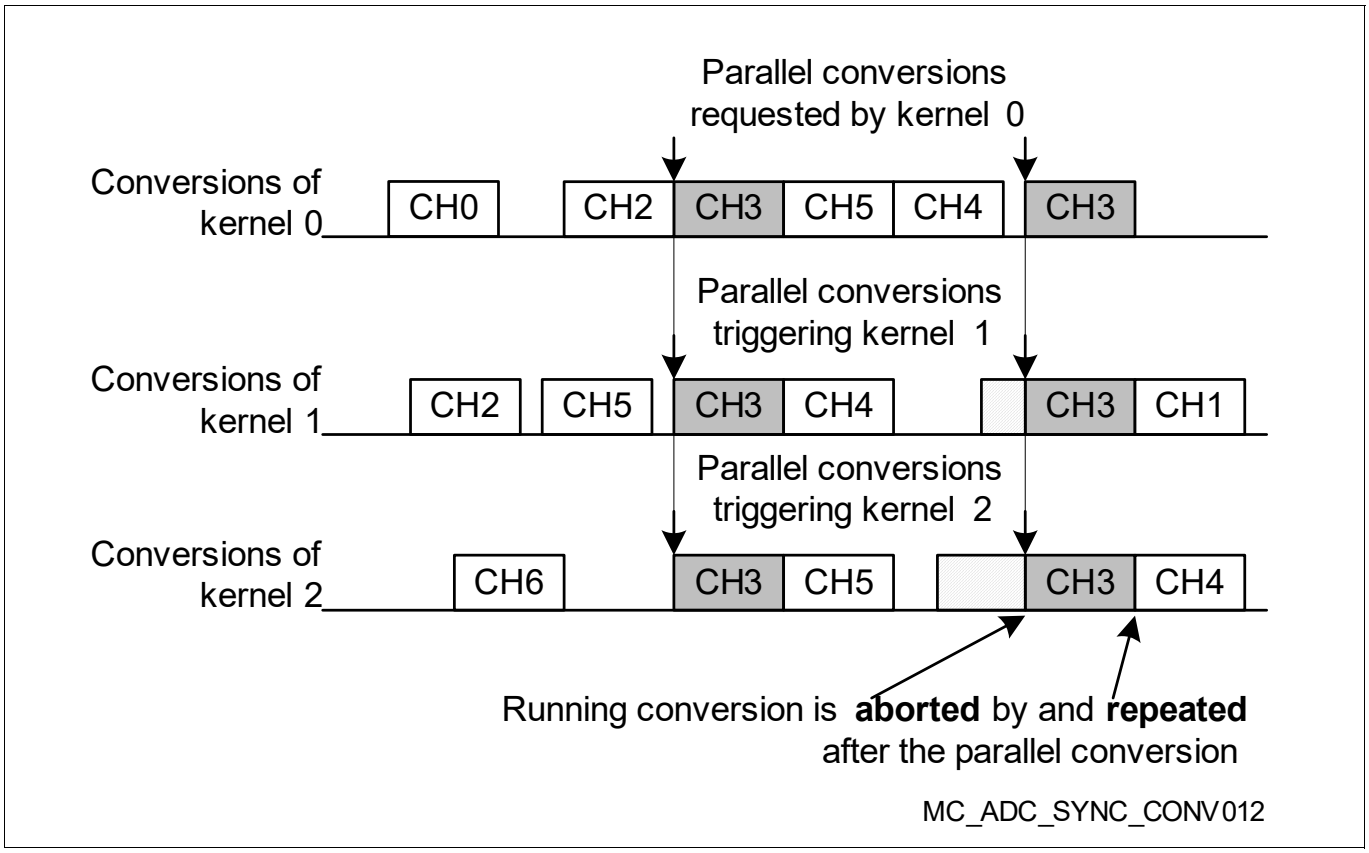


Figure 271 Parallel Conversions

One kernel operates as synchronization master, the other kernel(s) operate(s) as synchronization slave. Each kernel can play either role.

Master and slave kernels form a "synchronization group" to control parallel sampling:

1) For a summary, refer to "Synchronization Groups" in the product-specific appendix.

- **The synchronization master** controls the slave(s) by providing the control information **GxARBCFG (x=0-11)**.ANONS (see **Figure 272**) and the requested channel number.
  - Bitfield **GxSYNCTR (x=0-11)**.STSEL = 00_B selects the master's ANON information as the source of the ANON information for all kernels of the synchronization group.[1)]
  - The ready signals indicate when a slave kernel is ready to start the sample phase of a parallel conversion. Bit **GxSYNCTR (x=0-11)**.EVALRy = 1 enables the control by the ready signal (in the example, kernels 1 and 2 are slaves, so EVALR1 = EVALR2 = 1).
  - The master requests a synchronized conversion of a certain channel (SYNC = 1 in the corresponding channel control register GxCHCTRy), which is also requested in the connected slave ADC kernel(s).
  - Wait-for-read mode is supported for the master.
- **The synchronization slave** reacts to incoming synchronized conversion requests from the master. While no synchronized conversions are requested, the slave kernel can execute "local" conversions.
  - Bitfield **GxSYNCTR (x=0-11)**.STSEL = $01_B/10_B/11_B$ selects the master's ANON information as the source of the ANON information for all kernels of the synchronization group[1)] (in the example kernel 0 is the master, so STSEL = 01_B).
  - The ready signals indicate when the master kernel and the other slave kernels are ready to start the sample phase of a parallel conversion. Bit **GxSYNCTR (x=0-11)**.EVALRy = 1 enables the control by the ready signal (in the example kernel 0 is the master, so EVALR1 = 1, kernel 1 and 2 are slaves, so EVALR2 = 1).
  - A parallel conversion request is always handled with highest priority and cancel-inject-repeat mode.
  - Wait-for-read mode is ignored in the slave. Previous results may be overwritten, in particular, if the same result register is used by other conversions.
- Once started, a parallel conversion cannot be aborted.
  This ensures consistent results for the respective synchronization group.
  To achieve a deterministic behaviour in an application, make sure the master receives highest priority for synchronized conversions, so that it is not interrupted. This way, synchronous and standard conversions are arbitrated in the same way.

*Note: Synchronized conversions request the same channel number, defined by the master. Using the alias feature (see **Section 32.7.2**), analog signals from different input channels can be converted. This is advantageous if e.g. CH0 is used as alternate reference.*

1) STSEL = 00_B selects the own ANON information. The other control inputs (STSEL = $01_B/10_B/11_B$) are connected to the other kernels of a synchronization group in ascending order (see also "Synchronization Groups" in the product-specific appendix).

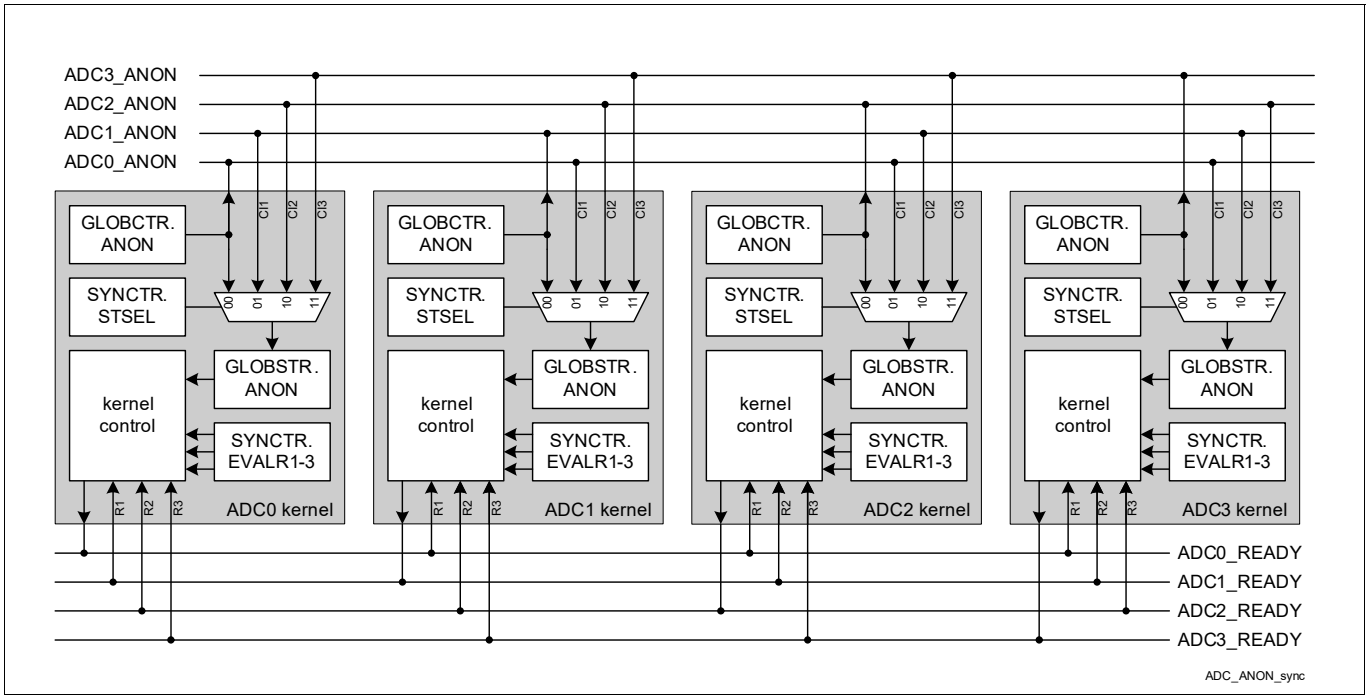


Figure 272 Synchronization via ANON and Ready Signals

Synchronization Control Register, Group x

The Synchronization Control Register controls the synchronization of kernels for parallel conversions.

Note: Program register GxSYNCTR only while bitfield GxARBCFG.ANONS = 00_B in all ADC kernels of the synchronization group. Set the master's bitfield ANONC to 11_B afterwards.

GxSYNCTR (x=0-11)

Synchronization Control Register, Group x ($04C0_H$+x*400_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									EVALR3	EVALR2	EVALR1	0		STSEL	
r									rw	rw	rw	r		rw	

Field	Bits	Type	Description
STSEL	1:0	rw	**Start Selection** Controls the synchronization mechanism of the ADC kernel. *Note: Control inputs CIx see **Figure 272**, connected kernels see product-specific appendix.* 00_B Kernel is synchronization master Use own bitfield GxARBCFG.ANONC. 01_B Kernel is synchronization slave Control information from input CI1. 10_B Kernel is synchronization slave Control information from input CI2. 11_B Kernel is synchronization slave Control information from input CI3.
EVALRi (i=1-3)	i+3	rw	**Evaluate Ready Input Ri** Enables the ready input signal for a kernel of a synchronization group. 0_B No ready input control 1_B Ready input Ri is considered for the start of a parallel conversion of this synchronization group
0	3:2, 31:7	r	**Reserved, write 0, read as 0**

32.11.2 Equidistant Sampling

To optimize the input data e.g. for filter applications, conversions can be executed in a fixed timing raster. Conversions for equidistant sampling are triggered by a timer signal.

Select timer mode (TMEN = 1 in register **GxQCTRLi (i=0-2;x=0-11)** etc.) for the intended source of equidistant conversions. In timer mode, a request of this source is triggered (by the rising trigger signal edge) and arbitrated, but only started when the trigger signal is removed (see **Figure 273**) and the converter is idle.

The frequency of the trigger signal REQTRx defines the sampling rate. Its high time defines the preface time interval where the corresponding request source takes part in the arbitration, while no other conversions are started.

The preface time (see **Figure 273**) must be longer than two clock cycles plus the highest possible conversion time.

The period of the trigger signal must be long enough to let the equidistant conversion and optional lower priority conversions complete before the next rising trigger edge.

The trigger signal for timer mode can be generated locally by the source request timer, which runs on the module clock. It generates periods up to ~100 µs. The preface time can be selected via bitfield SEQTIMOFF.

The trigger signal for timer mode can also be generated by an external timer. This allows a wider range of periods on the expense of the synchronization jitter, because the timer has an independent time base.

To ensure that the converter is idle and the start of conversion can be controlled by the trigger signal, the equidistant conversion requests must receive highest priority. The preface time between request trigger and conversion start must be long enough for a currently active conversion to finish.

Equidistant sampling is also supported for a sequence of channels. It is also possible to do equidistant sampling for more than one request source in parallel if the preface times and the equidistant conversions do not overlap.

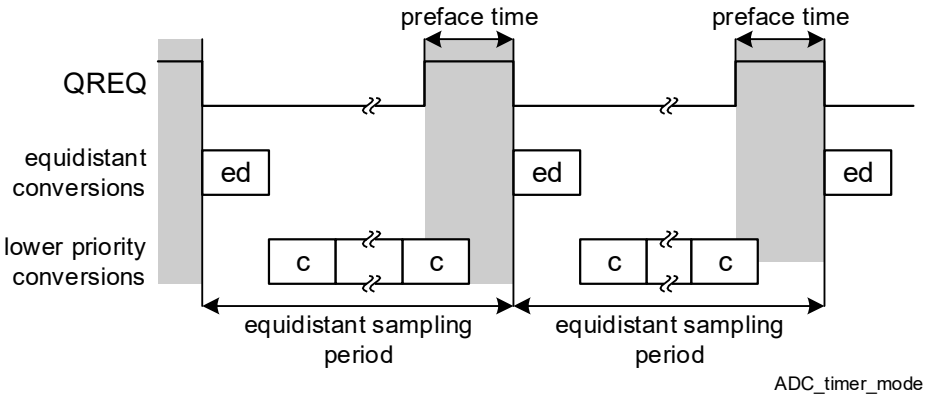


Figure 273 Timer Mode for Equidistant Sampling

Note: In timer mode, the trigger edge detector reacts on each rising edge. To achieve a deterministic behaviour (i.e. only one trigger), only use XTMODE = 10_B (i.e. rising edge) while timer mode is active.

32.11.3 Synchronous Sampling

Also the sampling phase can be aligned with the phase synchronizer. In this case (enabled by setting bit SSE in register **GxANCFG (x=0-11)**) a sampling phase is only started upon a trigger signal from the phase synchronizer.

The figure below summarizes the effects of sampling phase synchronization in combination with conversion synchronization.

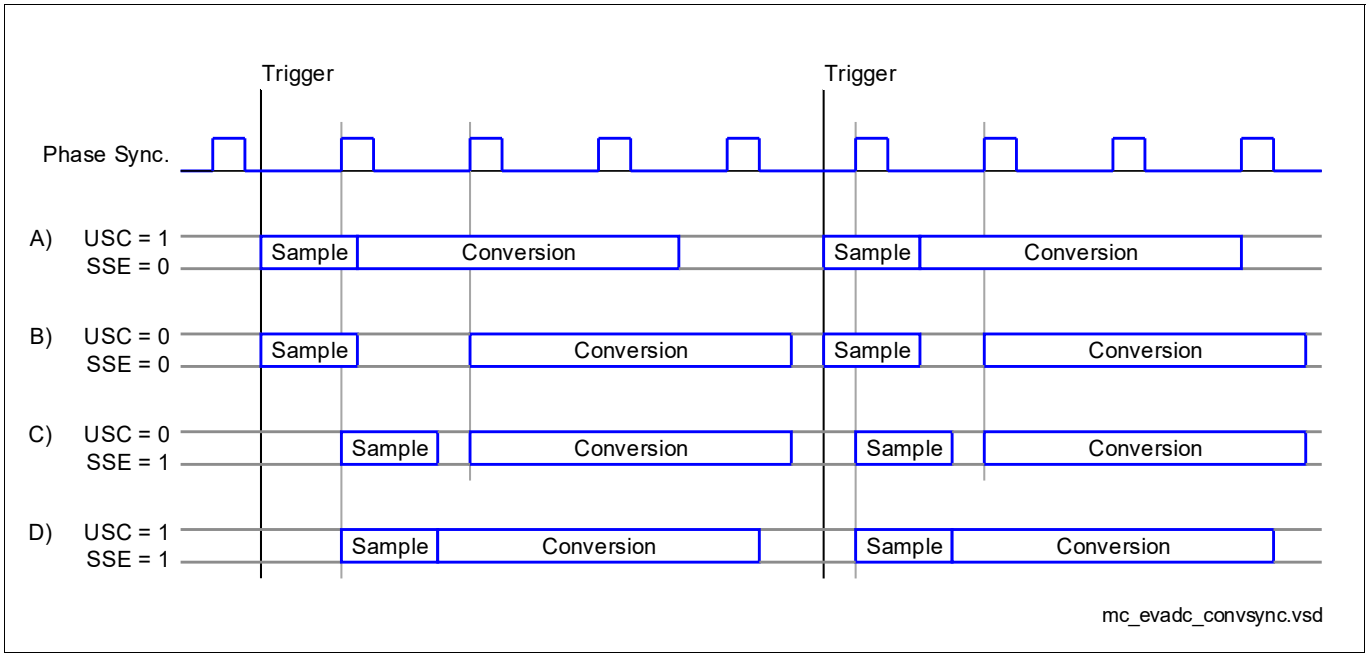


Figure 274 Synchronization of Sampling Phase

Table 263 Effects of Synchronization of Sampling and Conversion

Example in Figure	Sampling Synchronization	Conversion Synchronization	Description	Note
A	SSE = 0, unsynchronized	USC = 1, unsynchronized	Sampling starts immed. after the trigger, conversion starts immed. after sampling	As in previous products
B [1)]	SSE = 0, unsynchronized	USC = 0, synchronized	Sampling starts immed. after the trigger, conversion starts upon next PHSYNC pulse	Default after Reset
C [2)]	SSE = 1, synchronized	USC = 0, synchronized	Sampling starts upon next PHSYNC pulse, conversion starts upon next PHSYNC pulse	Optimized synchronization
D	SSE = 1, synchronized	USC = 1, unsynchronized	Sampling starts upon next PHSYNC pulse, conversion starts immed. after sampling	Not recommended

1) Provides minimum sample jitter with respect to the trigger signal.

2) Further reduces coupling effects when EDSADC channels are active.

*Note: Bit SSE controls the sampling of analog inputs. If the analog reference voltages are sampled (directly via CH30, CH31, or implicitly during the presample phase when broken wire detection is enabled) the sampling is controlled by bit USC in register **GLOBCFG**.*

32.12 Safety Features

Most analog inputs are connected to both EVADC and EDSADC channels, thus providing a basic redundancy.

Several additional test structures can be activated to test the signal path from the sensor to the input pin and the internal signal path from the input pin through the multiplexer to the converter. These test structures apply additional loads to the signal path (see summary in **Figure 275**) and aim at different sections of the signal flow. They can be controlled directly by software or can be configured for automatic execution.

- **Pull-Down Diagnostics** validates the connection of the external sensor
- **Multiplexer Diagnostics** validates the operation of the internal analog input multiplexer
- **Converter Diagnostics** validates the operation of the Analog/Digital converter itself
- **Broken Wire Detection** validates the connection from the sensor to the input pin
- **On-Chip Supervision Signals** enable additional monitoring

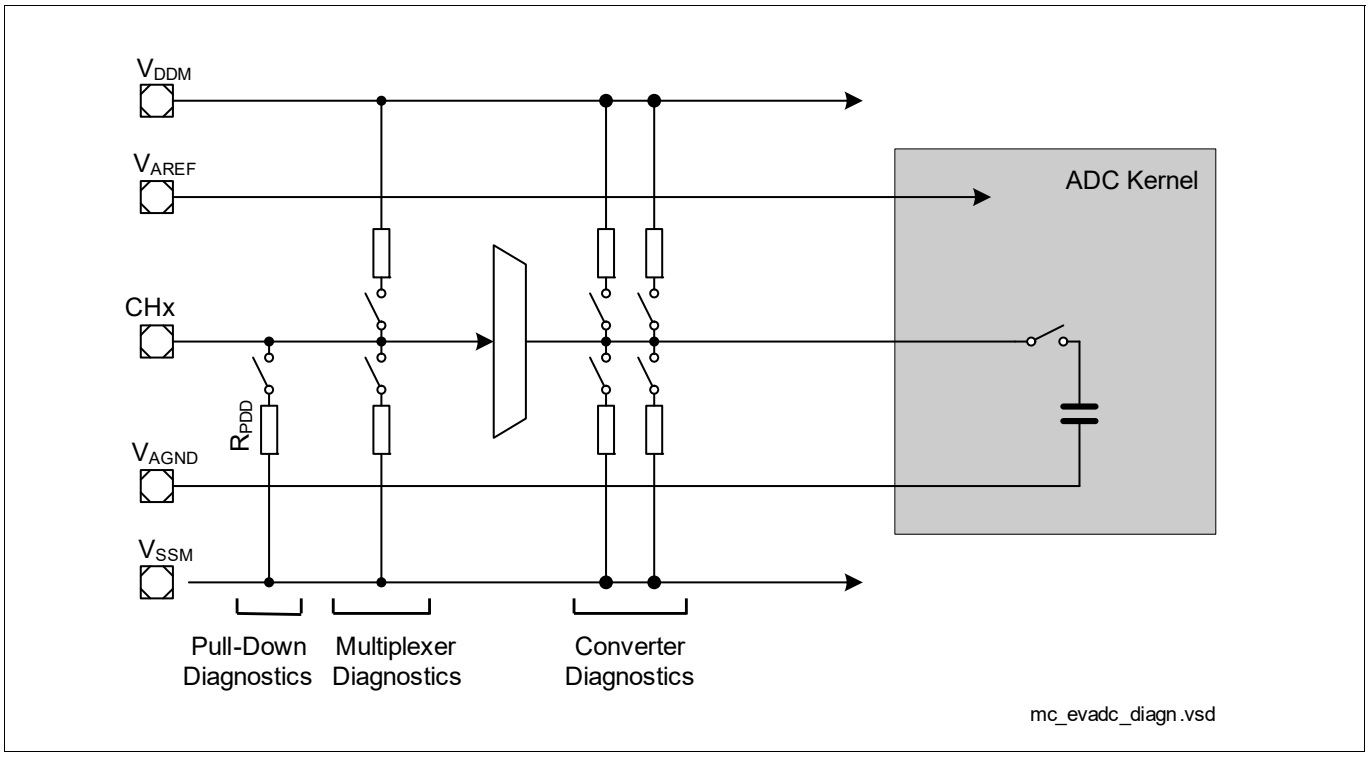


Figure 275 Signal Path Test

32.12.1 Pull-Down Diagnostics

One single input channel provides a further strong pull-down (R_{PDD}) that can be activated to verify the external connection to a sensor. The strong pull-down can be used to discharge an external buffer capacitor, which allows to generate diagnostic results quickly.

Pads belonging to IO ports need to be configured in the following way:

- Pn_PDISC.PDISi = 0, to enable the pull device, for normal operation disable the digital pad circuitry.
- Pn_PCSR.SELi = 1, to enable the test function.

32.12.2 Multiplexer Diagnostics

To test the proper operation of the internal analog input multiplexer, additional pull-up and/or pull-down devices can be connected to channels CH1 and CH2. In combination with a known external input signal this test function shows if the multiplexer connects any pin to the converter input and if this is the correct pin.

Multiplexer diagnostics uses the pull devices built into the pads. The strength of the pull devices depends on the respective pad properties. The Datasheet defines values for automotive level and TTL level.

Note: Class D pads (see pin description in Datasheet) are not configurable and use automotive level values.

Pads belonging to IO ports need to be configured in the following way:

- Pn_PDISC.PDISi = 0, to enable the pull devices.
- Pn_PCSR.SELi = 1, to enable the test function.
- Pn_PDRi.PLx selects the input level of the pad (if possible) and along with this the value of the pull devices.

Note: For normal operation (without multiplexer diagnostics) it is recommended to disable the digital pad circuitry (Pn_PDISC.PDISi = 1) to minimize noise.

32.12.3 Converter Diagnostics

To test the proper operation of the converter itself, several signals can be connected to the converter input. The test signals can be connected to the converter input either instead of the standard input signal or in parallel to the standard input signal.

To use the converter diagnostics without a connected input, select a non-existent channel number via the alias feature (**Section 32.7.2**). Redirect channel 0 or 1 to channel 24 (not connected), the conversion result can then be read from the selected result register for channel 0 or 1. This operating mode is recommended, as the test signals are not influenced by external circuitry in this case.

Bitfield CDSEL in register **GLOBTF** or **GxQINRi (i=0-2;x=0-11)** selects four different test signals (see **Figure 275**): V_{DDM}, V_{SSM}, $1/2 \times V_{DDM}$, $2/3 \times V_{DDM}$.

Note: The maximum result value (when using V_{DDM}) the converter can generate is 4 095 (FFF_H), the minimum result value (when using V_{SSM}) is 0 (000_H). Results are generated according to V_{AREF}.

32.12.4 Broken Wire Detection

To test the proper connection of an external analog sensor to its input pin, the converter's capacitor can be discharged before the regular sample phase. If the connection to the sensor is interrupted the subsequent conversion value will rather represent a reduced value than the expected sensor result. Because the precharge voltage is outside the expected result range (broken wire detection uses V_{AGND}) a valid measurement (sensor connected) can be distinguished from a failure (sensor detached).

While broken wire detection is disabled, the converter's capacitor is precharged to $V_{AREF}/2$, if idle precharge is enabled.

Note: The duration of the complete conversion is increased by the preparation phase (same as the sample phase) if the broken wire detection is enabled. This influences the timing of conversion sequences.

Broken wire detection can be enabled for each channel separately by bitfield BWDEN in the corresponding channel control register (GxCHCTRy).

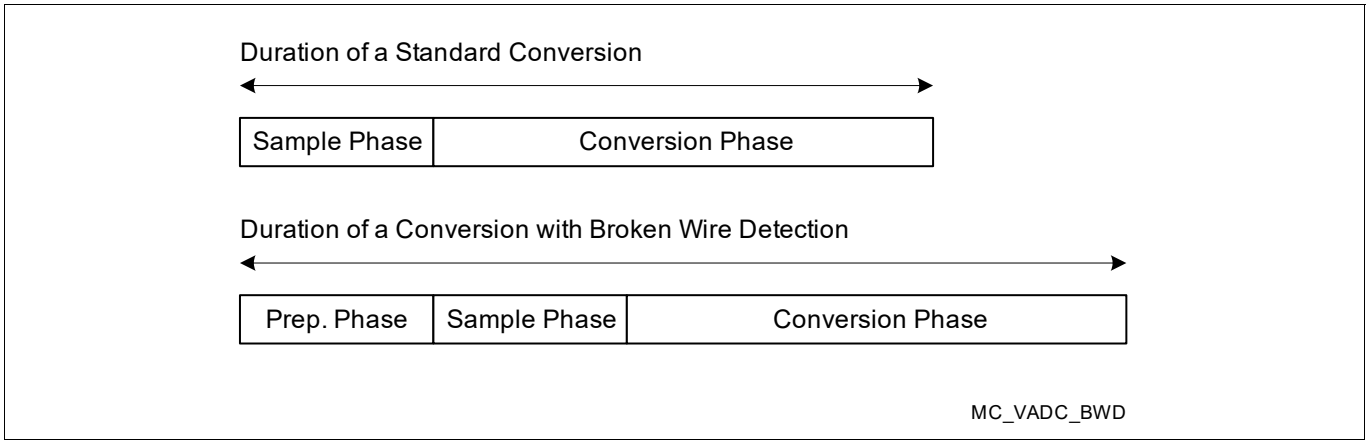


Figure 276 Broken Wire Detection

32.12.5 On-Chip Supervision Signals

Information about the basic functionality of the EVADC can be obtained via special on-chip signals, which supports common cause diagnosis. Every group can measure (primary/secondary) or compare (fast compare) a replica of a reference voltage generated by a bandgap inside the power management system (PMS). The expected results are related to the external reference voltage V_{AREF}.

Table 264 Supervision Signals

Channel	Signal	Description
GxCH28[1)]	$V_{ANACOMM}$	Common reference signal, available to all converters. Can be fed to the converters through analog input pin AN11.
GxCH29[1)]	V_{MTS}	Module test signal, provides the comparator supply voltage V_{DDK}, which is controlled by the bandgap in the power subsystem. See figure below.
GxCH30[1)]	V_{AGND}	External reference ground.
GxCH31[1)]	V_{AREF}	External reference voltage.
G10CH15	V_{EDSADC}	Supervision signal from module EDSADC. This supervision signal is enabled and selected within the EDSADC.[2)]

1) Use the alias feature to select these channels.

2) This signal requires an increased sample time of at least 1 µs.

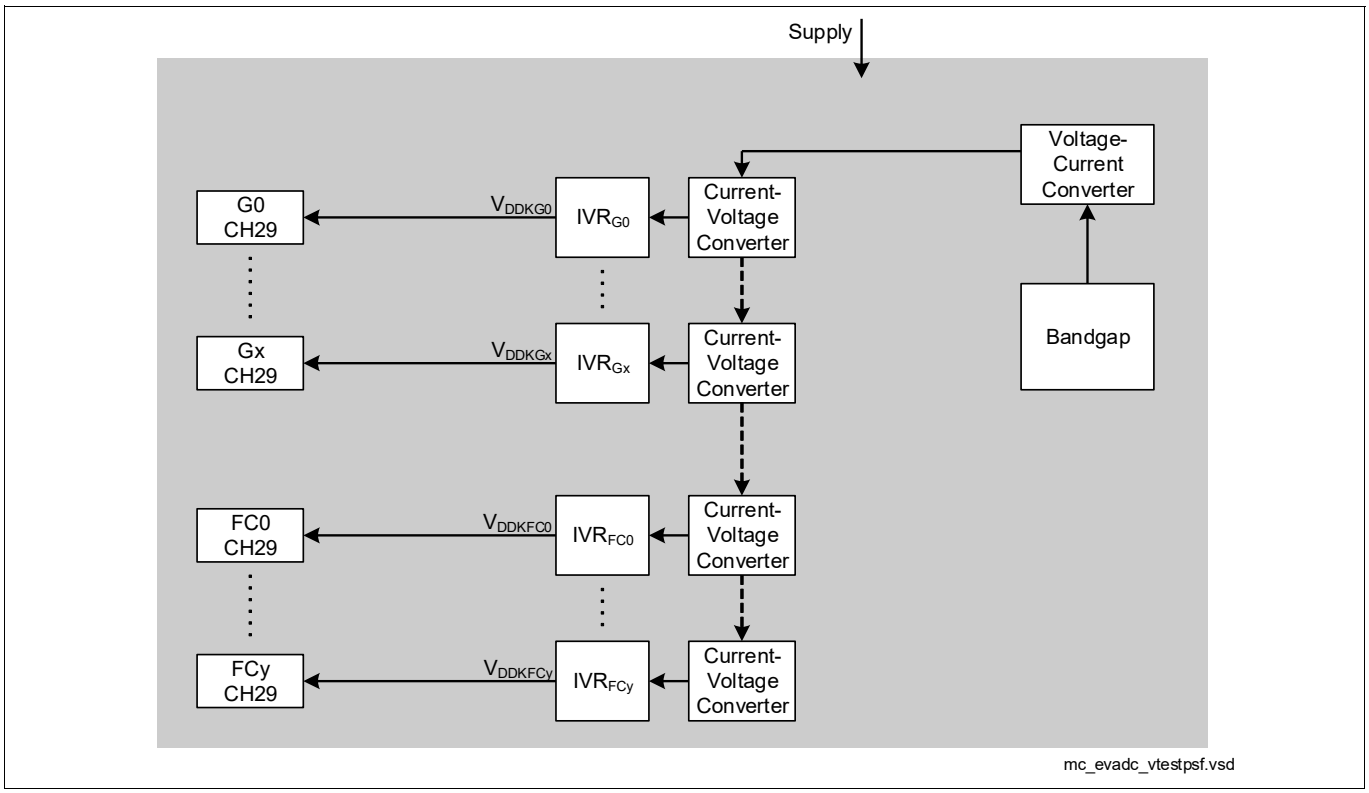


Figure 277 Test Voltages for Supervision

Measuring V_{DDK} enables two test features:

- Compare the result with the expected value (RESULT = V_{DDK} / V_{AREF} × 2^{12}).
- Compare the individual results of all converters to find deviations.

The variation of the values for V_{DDK} depends on manufacturing variation, on the supply voltage V_{DDM}, and on the die temperature. To compensate this variation, device-specific base values are measured during production and are stored in the on-chip Flash. The lower halfword of a record (see **Table 265**) stores the voltage at -40°C (VDDKC in [mV]), the next halfword stores the temperature deviation (DVDDK in [µV/K]).

The following formula uses the current die temperature (T_J in [°C]):

V_{DDK} = VDDKC + (DVDDK × (T_J + 40) / 1 000) mV

This provides a device-specific base value for V_{DDK} at the current temperature.

Example:

For T_J = 125°C, VDDKC = $04D9_H$ (1 241 mV), DVDDK = $005A_H$ (90 µV/K):

V_{DDK} = $04D9_H$ + (($5A_H$ × (125 + 40) / 1 000) mV = 1 241 + (90 × 165 / 1 000) mV =
= 1 241 + 15 mV = 1.256 V

Applying a deviation of +/-2% results in a range (for V_{AREF} = 5.0 V) of

1.231 V ... 1.281 V --> 1 008 ... 1 049 = $03F0_H$ ... 0419_H.

Stored Calibration Values

The device-specific calibration values are stored in the User Configuration Block (UCB_USER) within the on-chip Flash memory. The area assigned to the EVADC begins at offset 0020_H and provides records of two 32-bit words per group. The channel-specific records, therefore, can be read from offset [0020_H + 8×GroupNr.] within block UCB_USER. The table below shows the structure of a record.

Table 265 Structure of EVADC Calibration Records

	Halfword 3	Halfword 2	Halfword 1	Halfword 0
Location G0	0026_H	0024_H	0022_H	0020_H
Location G1	$002E_H$	$002C_H$	$002A_H$	0028_H
:	:	:	:	:
Stored value	Reserved	Reserved	DVDDK	VDDKC

The parameters are stored in the following format as 16-bit 2's complement numbers:

The low-temperature voltage VDDKC is stored in [mV], ranging from 1.15 V … 1.35 V ($047E_H$ … 0546_H).

The temperature deviation DVDDK is stored in [µV/K], ranging from -500 µV/K … 500 µV/K ($FE0C_H$ … $01F4_H$).

32.12.6 Configuration of Test Functions

Test functions allow software control of pull-up and pull-down devices and internal reference channels. Various test levels can be applied controlling the devices in an adequate way. Because these test functions interfere with the normal operation of the A/D Converters, they are controlled by a separate register set.

Not all test options are available for each channel. Selecting an unavailable function has no effect.

Test conversion sequences (i.e. conversions with activated test functions) can automatically be executed via request source Q2.

Global Test Functions Register

Note: All diagnostic functions are activated for the selected group (CDGR). Converter diagnostics (CDEN, CDSEL) and pull-down diagnostics (PDD) are activated directly by setting the associated bits, multiplexer diagnostics (MDPD, MDPU) are activated for the selected channel (CDCH).

GLOBTF
Global Test Functions Register (0160$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								MDWC	0				MDPU	MDPD	PDD
r								w	r				rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CDWC	0				CDSEL		CDEN	CDGR				CDCH			
w	r				rw		rw	rw				rw			

Field	Bits	Type	Description
CDCH	3:0	rw	**Conversion Diagnostics Channel** Defines the channel number to be used for diagnostic conversions. Applies to MDPD, MDPU.
CDGR	7:4	rw	**Conversion Diagnostics Group** Defines the group number to be used for diagnostic conversions. Applies to all test functions of primary and secondary groups.
CDEN	8	rw	**Converter Diagnostics Enable** 0_B All diagnostic pull devices are disconnected 1_B Diagnostic pull devices connected as selected by bitfield CDSEL
CDSEL	10:9	rw	**Converter Diagnostics Pull-Devices Select** 00_B Connected to VDDM 01_B Connected to VSSM 10_B Connected to 1/2 VDDM 11_B Connected to 2/3rd VDDM
CDWC	15	w	**Write Control for Conversion Diagnostics** 0_B No write access to parameters 1_B Bitfields CDSEL, CDEN, CDGR, CDCH can be written

Field	Bits	Type	Description
PDD	16	rw	**Pull-Down Diagnostics Enable** *Note: Channels with pull-down diagnostics device are marked in the product-specific appendix.* 0_B Disconnected 1_B The pull-down diagnostics device is active
MDPD	17	rw	**Multiplexer Diagnostics Pull-Down-Devices Enable** Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with pull-down diagnostics device are marked in the product-specific appendix.* 0_B Disconnected 1_B The respective pull-down device is active
MDPU	18	rw	**Multiplexer Diagnostics Pull-Up-Devices Enable** Connecting combinations of pull-up and/or pull-down devices generate various loads for testing. *Note: Channels with pull-up diagnostics device are marked in the product-specific appendix.* 0_B Disconnected 1_B The respective pull-up device is active
MDWC	23	w	**Write Control for Multiplexer Diagnostics** 0_B No write access to parameters 1_B Bitfields MDPD, MDPU, PDD can be written
0	14:11, 22:19, 31:24	r	**Reserved, write 0, read as 0**

Table 266 Access Mode Restrictions of GLOBTF sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **CDWC**	rw	CDCH, CDEN, CDGR, CDSEL	Set CDWC during write access
write 1 to **MDWC**	rw	MDPD, MDPU, PDD	Set MDWC during write access
(default)	r	CDCH, CDEN, CDGR, CDSEL, MDPD, MDPU, PDD	

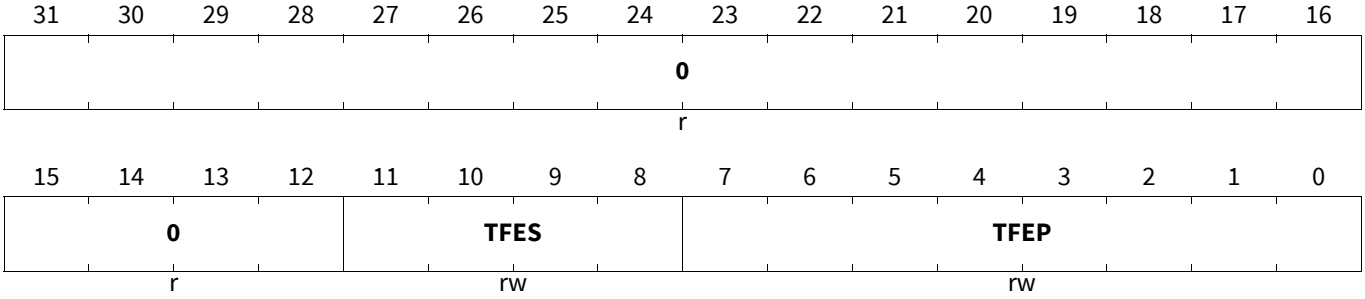
When automatic test sequences (see **Section 32.12.7**) are enabled, the scheduled conversions will activate the configured test devices. To make sure that this does not happen inadvertently, the activation of the test functions must be particularly enabled.

The release is done by setting the corresponding bit in the Global Test Enable register (GLOBTE). This prevents spurious control signals from spoiling a configured test sequence. GLOBTE can be write-protected via bit APTF in register ACCPROT2 (ACCPROT2 itself is safe-endinit protected).

*Note: The pull device control bits in register **GLOBTF** can only activate the test pull devices of a group, when the corresponding control bit in register **GLOBTE** is zero.*

Global Test Enable Register

GLOBTE
Global Test Enable Register (0164$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TFEP	7:0	rw	**Test Function Enable, Primary Groups** Each bit of this bitfield is associated with the corresponding group. 0: No test functions for group x sequences 1: Test functions can be activated by group x sequences
TFES	11:8	rw	**Test Function Enable, Secondary Groups** Each bit of this bitfield is associated with the corresponding group. 0: No test functions for group x sequences 1: Test functions can be activated by group x sequences
0	31:12	r	**Reserved, write 0, read as 0**

32.12.7 Automatic Execution of Test Sequences

Validating the correct connection of an input signal (sensor) to the converter in a system requires some action, depending on the properties of the signal to be converted. For example, sensors with a limited signal range can be validated by checking the conversion result for values outside of the defined signal range.

In other cases pull devices (up / down) can be connected to the signal input, which alter the signal level and, as a consequence, the conversion result in a known way. The alteration can be anticipated when the signal source impedance is known.

Timing Considerations

Test sequences that are executed during the initialization of a system usually are not time-critical. They increase the start-up time but otherwise have no impact on the application. Safety-critical systems, however, monitor the operability of the system during the operation time. In this case, the test sequences must be interleaved with the normal application sequences. A test sequence shall be executed immediately after an application sequence, so the relaxation time until the next application sequence becomes maximal.

Test sequences can be triggered by each available request source.

In either case the overall timing must be selected so overlaps are avoided.

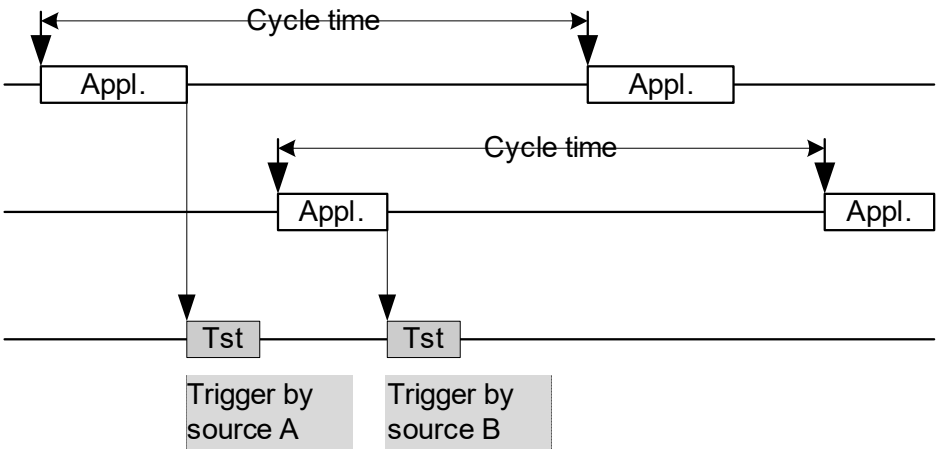


Figure 278 Positions of Test Sequences

Note that the test conversions can target arbitrary channels.

Using the Test Queue

The test queue request source (queue 2) is similar to the standard queued request sources with some additional control bits to select test functions of the pins.

Leaving these control bits as zero allows using queue 2 as an additional request source for the application.

Queue 2 can be activated by external triggers (same as other sources) or by internal triggers from the other group request sources. These additional internal trigger signals are selected by an additional multiplexer whose output is connected to the uppermost input of the gate multiplexer. To use the internal triggers, select the uppermost trigger input (XTSEL = 1111_B) and the uppermost gate input (GTSEL = 1111_B).

A trigger counter activates queue 2 after a defined number of triggers. Test sequences can, therefore, be inserted into the application-specific conversion sequences at a certain rate without additional software intervention.

Control the internal triggers and the trigger counter via register **GxTRCTR (x=0-11)**.

Queue 2 contains entries building a sequence of test conversions which enable arbitrary diagnostic functions. A trigger for a test sequence is generated automatically when the sequence of a standard request source is completed.

The trigger sequence counter starts a (test) sequence after a programmable number of triggers. The (test) sequence is started if the trigger sequence counter equals 0 when the trigger occurs. For other values of the trigger sequence counter, each trigger decrements the counter. No software intervention is required to interleave test sequences into the normal operation.

Activation/Deactivation of Pull-Devices

When specific test functions are enabled for an entry in queue 2, the corresponding enabled pull-devices are controlled automatically:

They are activated when the sample phase for this conversion begins.
They are deactivated automatically after the sample phase is completed.[1)]

Depending on the properties of the connected sensor, a test sequence may require an additional settling time. This can be covered by selecting an increased sample time for these test conversions.

Detecting Irregular Test Sequences

An overflow bit (OV) is set when an internal trigger for queue 2 occurs while the queue is active (ind. by bit QACT). This indicates that a (test) sequence has been triggered when it cannot be executed as expected.

Queue 2 is assumed to be active (QACT = 1) between the occurrence of an internal trigger (set QACT) and the generation of a source event of queue 2 (clear QACT). This phase is indicated by bit QACT = 1.

Software can clear bits OV and QACT by writing 1 to bit COV (see register **GxTRCTR (x=0-11)**).

Securing the Test Functions

When test sequences are enabled, the scheduled conversions will activate the configured test pull devices. To make sure that this does not happen inadvertently, the activation of the test functions must be explicitly enabled.

The release is done by setting the corresponding bit in the Global Test Enable register (**GLOBTE**). This prevents spurious control signals from spoiling a configured test sequence. GLOBTE can be write-protected via bit APTF in register ACCPROT2 (ACCPROT2 itself is safe-endinit protected).

*Note: The pull device control bits in register **GLOBTF** can only activate the test pull devices of a group, when the corresponding control bit in register **GLOBTE** is zero.*

Evaluation of Test Results

The effect of the selected test pull devices depends on the properties of the connected signal source. These properties are system-specific and are not known to the controller. Therefore, the system software must evaluate the test results by comparing them to the expected results. This is influenced by several parameters:

- Selected pull device
- Properties of signal source, e.g. impedance
- Properties of input signal (e.g. current level of dynamic signals)
- Acceptable failure detection time

These aspects must be evaluated to assess the validity of the input signal on the channel being checked.

1) Do not deactivate the converter inbetween, to avoid the pull devices remaining active.

32.13 External Multiplexer Control

The number of analog input channels can be increased by connecting external analog multiplexers to an input channel. The EVADC can be configured to control these external multiplexers automatically.

Each available EMUX control interface can be controlled by an arbitrary group (see register **EMUXSEL**). One channel or a set of channels from that group can be selected for this operating mode (see register **GxEMUXCS (x=0-11)**). The EVADC supports 1-out-of-8 multiplexers with several control options:

- **Steady mode** converts the configured external channel
  when the selected channel is encountered. Conversion sequences:
  --4-32-2-1-0- -4-32-2-1-0- -4-32-2-1-0- -… (for configuration as shown in **Figure 279**)
  --4-32-2-12-0- -4-32-2-12-0- -4-32-2-12-0- -… (for configuration with EMUX channels 1 and 3)
- **Single-step mode** converts one external channel of the configured sequence
  when the selected channel is encountered. Conversion sequences:
  --4-32-2-1-0- -4-31-2-1-0- -4-30-2-1-0- -4-32-… (for configuration as shown in **Figure 279**)
  --4-32-2-11-0- -4-30-2-12-0- -4-31-2-10-0- -4-32-… (for configuration with EMUX channels 1 and 3)
  (Single-step mode works best with one channel)
- **Sequence mode** automatically converts all configured external channels[1)]when the selected channel is encountered. Conversion sequences:
  --4-32-31-30-2-1-0- -4-32-31-30-2-1-0- -… (for configuration as shown in **Figure 279**)
  --4-32-31-30-2-12-11-10-0- -4-32-31-30-2-… (for configuration with EMUX channels 1 and 3)
- **Block mode** converts the same external channel of the configured sequence for all EMUX channels[2)]when a selected channel is encountered. Conversion sequences:
  --4-32-2-12-0- -4-31-2-11-0- -4-30-2-10-0- -4-32-… (for cfg. with EMUX channels 1 and 3, EMUXCCB = 01_H)

*Note: The example in **Figure 279** has an external multiplexer connected to channel CH3. The start selection value EMUXSET is assumed as 2. The conversion sequence is assumed as 4-3-2-1-0.*
The alternate sequences described in the list above assume a similar EMUX circuitry on channel CH1.

Bitfield EMUXACT determines the control information sent to the external multiplexer.
In single-step mode, EMUXACT is updated after each conversion of an enabled channel. If EMUXACT = 000_B it is reloaded from bitfield EMUXSET, otherwise it is decremented by 1.

Additional external channels may have different properties due to the modified signal path. Local filters may be used at the additional inputs (R_{EXT2}-C_{EXT2} on CH3x in **Figure 279**). For applications where the external multiplexer is located far from the ADC analog input, it is recommended to add an RC filter directly at the analog input of the ADC (R_{EXT1}-C_{EXT1} on CH3 in **Figure 279**). Each RC filter limits the bandwidth of the analog input signal.C_{EXT1} must be charged from the respective C_{EXT2} circuit after switching the external multiplexer.

Conversions for external channels, therefore, use the alternate settings (AIPE, CME, SESPE) defined in the ICLASS registers. This automatically selects a different conversion mode if required.

Switching the external multiplexer usually requires an additional settling time for the input signal. Therefore, the alternate sample time setting STCE is applied each time the external channel is changed. This automatically fulfills the different sampling time requirements in this case. If enabled by bit EMXST STCE can also be applied for each conversion for external channels.

1) In sequence mode, the main channel number is preserved in register GxQBURi (see **Section 32.5**).
Invalidating GxQBURi aborts the current EMUX sequence and must be avoided. Please note following restrictions:
- Discarding a cancelled conversion is not possible when an external multiplexer is operated in sequence mode. Bit RPTDIS must be 0 in this case. See also **Section 32.6.2**.
- Do not clear valid flags via software (GxQMR.CLRV).

2) The external channel number is switched when the channel defined in bitfield EMUXCCB is converted.
EMUXCCB should, therefore, contain the channel that is converted last within a sequence.

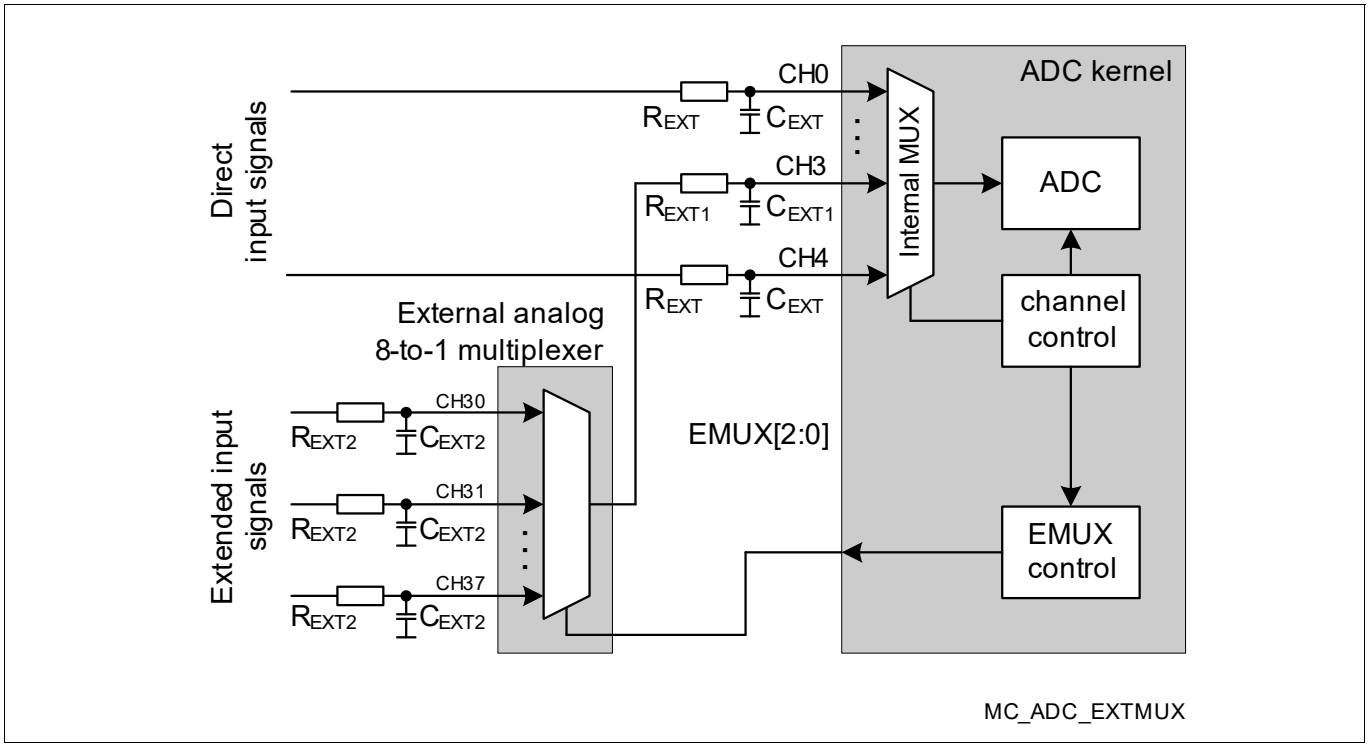


Figure 279 External Analog Multiplexer Example

Control Signals

The external channel number that controls the external multiplexer can be output in standard binary format or Gray-coded. Gray code avoids intermediate multiplexer switching when selecting a sequence of channels, because only one bit changes at a time. **Table 267** indicates the resulting codes.

Table 267 EMUX Control Signal Coding

Channel	0	1	2	3	4	5	6	7
Binary	000_B	001_B	010_B	011_B	100_B	101_B	110_B	111_B
Gray	000	001	011	010	110	111	101	100

Operation Without External Multiplexer

If no external multiplexers are used in an application, the reset values of the control registers provide the appropriate setup.

EMUXMODE = 00_B disables the automatic EMUX control.

Since the control output signals are alternate port output signals, they are only visible at the respective pins if explicitly selected.

In each group an arbitrary channel (EMXCCS = 0) or set of channels (EMXCCS = 1) can be assigned to external multiplexer control via register **GxEMUXCS (x=0-11)**.

Each available port interface selects the group whose control lines are output (register **EMUXSEL**).

External Multiplexer Control Reg., Group x

GxEMUXCTR (x=0-11)

External Multiplexer Control Reg., Group x (05F0$_H$+x*400$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							EMUXCCB					0	EMUXACT		
r							rw					r	rh		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMXWC	EMXCSS	EMXST	EMXCOD	0					EMUXMODE			0	EMUXSET		
w	rw	rw	rw	r					rw			r	rw		

Field	Bits	Type	Description
EMUXSET	2:0	rw	**External Multiplexer Start Selection** Defines the initial selection for the external multiplexer.
EMUXMODE	6:4	rw	**External Multiplexer Mode** Not listed combinations are reserved.[1)] 000_B Software control (no hardware action) 001_B Steady mode (use EMUXSET value) 010_B Single-step mode[2)] 011_B Sequence mode 100_B Block mode
EMXCOD	12	rw	**External Multiplexer Coding Scheme** 0_B Output the channel number in binary code 1_B Output the channel number in Gray code
EMXST	13	rw	**External Multiplexer Sample Time Control** 0_B Use STCE whenever the external channel selection changes 1_B Use STCE for each conversion of an external channel
EMXCSS	14	rw	**External Multiplexer Channel Selection Style** 0_B Channel number Bitfield EMUXCH selects an arbitrary channel 1_B Channel enable Each bit of bitfield EMUXCH selects the associated channel for EMUX control
EMXWC	15	w	**Write Control for EMUX Configuration** 0_B No write access to EMUX cfg. 1_B Bitfields EMXMODE, EMXCOD, EMXST, EMXCSS can be written
EMUXACT	18:16	rh	**External Multiplexer Actual Selection** Defines the current value for the external multiplexer selection. This bitfield is loaded from bitfield EMUXSET and modified according to the operating mode selected by bitfield EMUXMODE.

Field	Bits	Type	Description
EMUXCCB	24:20	rw	**External Multiplexer Channel Selection for Block Mode** Defines the channel that switches EMUXACT when converted. In block mode, all EMUX channels use the same control value.
0	3, 11:7, 19, 31:25	r	**Reserved, write 0, read as 0**

1) For single-step mode, sequence mode and block mode: Select the start value with EMUXMODE=000_B before selecting the respective mode.
2) Single-step mode modifies the EMUX channel number each time an EMUX-enabled channel is converted. Therefore, single-step mode works best with a single channel, because otherwise some external channels may be skipped.

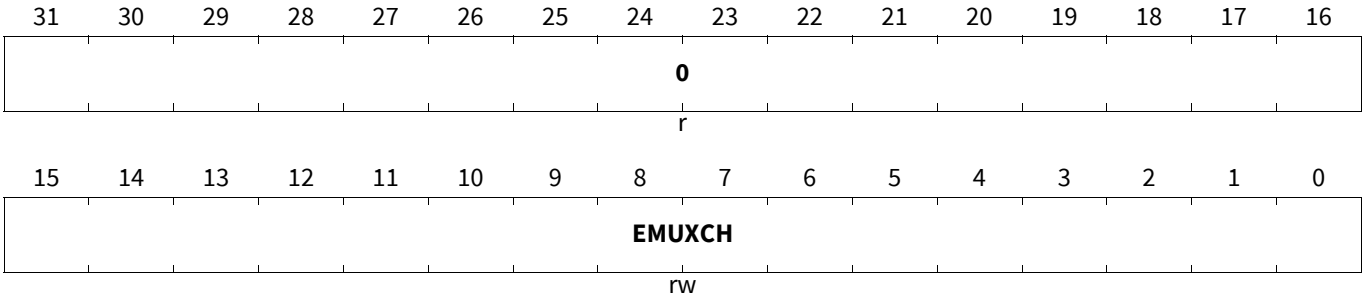
Table 268 Access Mode Restrictions of GxEMUXCTR (x=0-11) sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **EMXWC**	rw	EMUXMODE, EMXCOD, EMXCSS, EMXST	Set EMXWC during write access
(default)	r	EMUXMODE, EMXCOD, EMXCSS, EMXST	

Ext. Multiplexer Channel Select Reg., Group x

GxEMUXCS (x=0-11)

Ext. Multiplexer Channel Select Reg., Group x($05F4_H$+x*400_H) **Application Reset Value: 0000 0000_H**



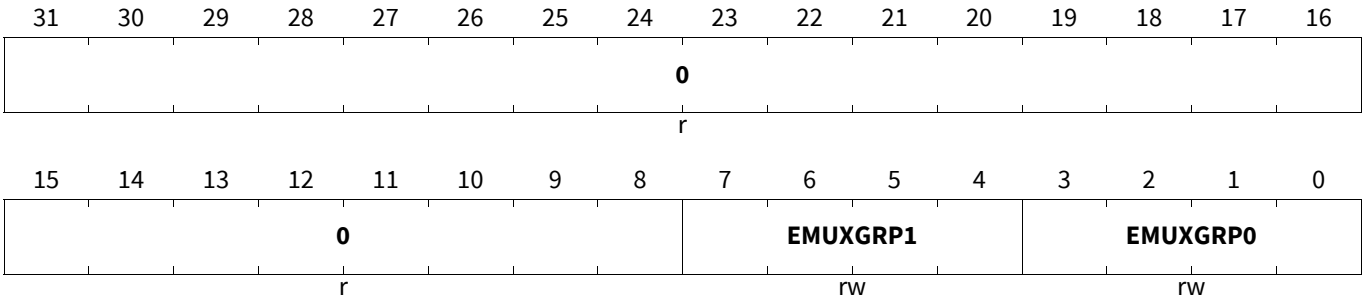
Field	Bits	Type	Description
EMUXCH	15:0	rw	**External Multiplexer Channel Select** Defines the channel(s) to which the external multiplexer control is applied. **EMXCSS = 0: Channel number** the lower 5 bits select an arbitrary channel (valid numbers are limited by the number of available channels, unused bits shall be 0) **EMXCSS = 1: Channel enable** each bit enables the associated channel (multiple channels can be selected/enabled)
0	31:16	r	**Reserved, write 0, read as 0**

External Multiplexer Interface Select Register

Register EMUXSEL is a global register which assigns an arbitrary group to each of the EMUX interfaces. [1)]

EMUXSEL

External Multiplexer Interface Select Register (03F0$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
EMUXGRP0	3:0	rw	**External Multiplexer Group for Interface 0** Defines the group whose external multiplexer control signals are routed to EMUX interface 0 (pins EMUX0x).
EMUXGRP1	7:4	rw	**External Multiplexer Group for Interface 1** Defines the group whose external multiplexer control signals are routed to EMUX interface 1 (pins EMUX1x).
0	31:8	r	**Reserved, write 0, read as 0**

1) The pins that are associated with each EMUX interface are listed in the product-specific appendix.

32.14 Service Request Generation

Each A/D Converter can activate up to 4 group-specific service request output signals and up to 4 shared service request output signals to issue an interrupt or to trigger a DMA channel. Two common service request groups are available, see product-specific appendix.

Several events can be assigned to each service request output. Service requests can be generated by three types of events:

- **Request source events:** indicate that a request source completed the requested conversion sequence and the application software can initiate further actions.
  For a group queue source, the event is generated according to the programming, i.e. when a channel with enabled source interrupt has been converted or when an invalid entry is encountered.
- **Channel events:** indicate that a conversion is finished. Optionally, channel events can be restricted to result values within a programmable value range. This offloads the CPU/DMA from background tasks, i.e. a service request is only activated if the specified conversion result range is met or exceeded.
- **Result events:** indicate a new valid result in a result register. Usually, this triggers a read action by the CPU (or DMA). Optionally, result events can be generated only at a reduced rate if data reduction is active.
  For example, a single DMA channel can read the results for a complete auto-scan sequence, if all channels of the sequence target the same result register and the transfers are triggered by result events.

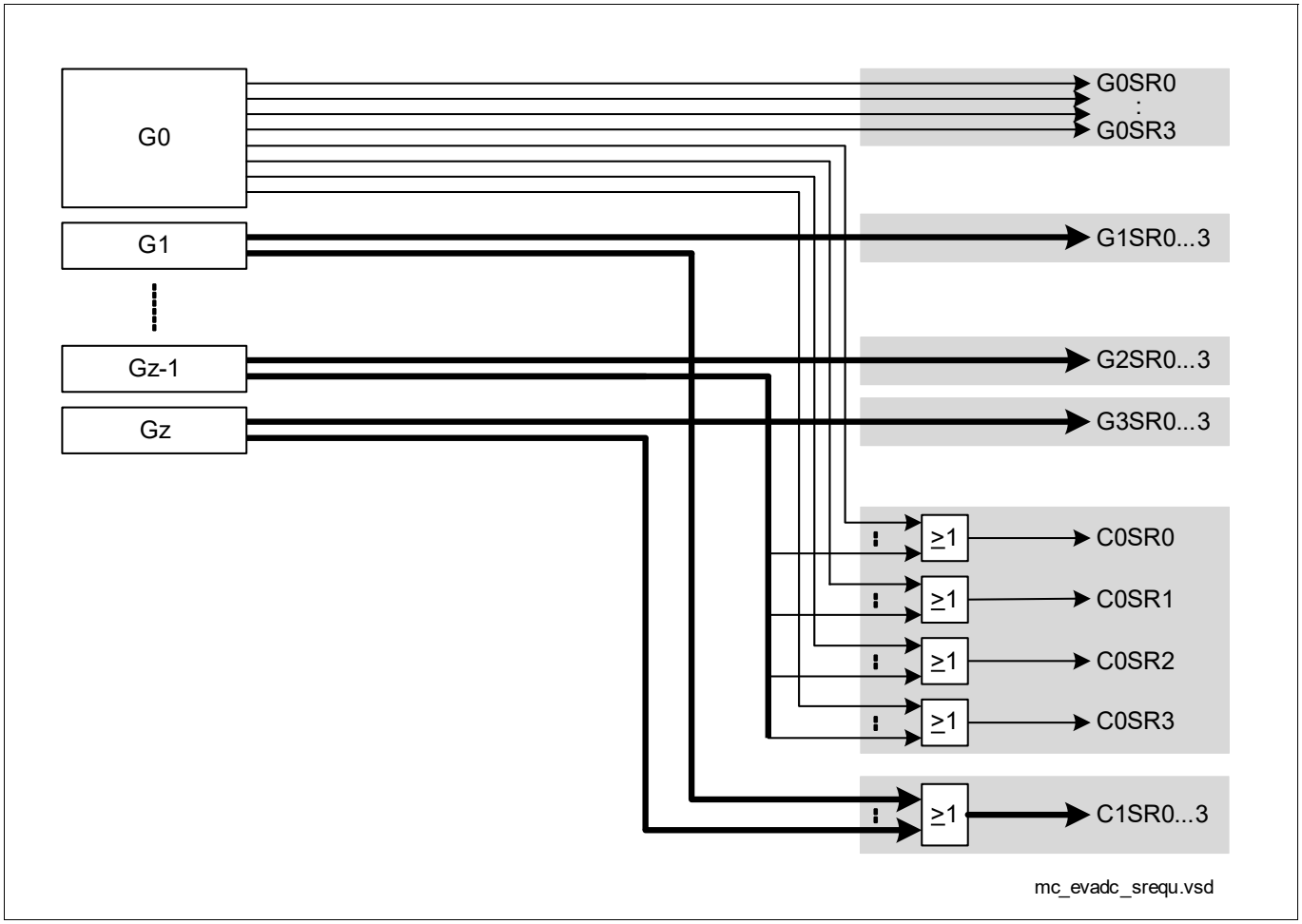


Figure 280 Service Request Overview

Each ADC event is indicated by a dedicated flag that can be cleared by software. If a service request is enabled for a certain event, the service request is generated for each event, independent of the status of the corresponding

event indication flag. This ensures efficient DMA handling of ADC events (the ADC event can generate a service request without the need to clear the indication flag).

Event flag registers indicate all types of events that occur during the ADC's operation. Software can set each flag by writing a 1 to the respective position in register GxCEFLAG/GxREFLAG to trigger an event. Software can clear each flag by writing a 1 to the respective position in register GxCEFCLR/GxREFCLR. If enabled, service requests are generated for each occurrance of an event, even if the associated flag remains set.

Node Pointer Registers

Requests from each event source can be directed to a set of service request nodes via associated node pointers. Requests from several sources can be directed to the same node; in this case, they are ORed to the service request output signal.

32.14.1 Source Event Flag Registers, Group x

These registers handle the events generated by the conversion request sources (Q0, Q1, Q2).

Note: Software can set all flags in register GxSEFLAG and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear all flags in register GxSEFLAG by writing 1 to the respective bit in register GxSEFCLR.

Source Event Flag Register, Group x

GxSEFLAG (x=0-11)

Source Event Flag Register, Group x **(0588_H+x*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													SEV2	SEV1	SEV0
r													rwh	rwh	rwh

Field	Bits	Type	Description
SEVi (i=0-2)	i	rwh	**Source Event i** 0_B No source event 1_B A source event has occurred
0	31:3	r	**Reserved, write 0, read as 0**

Source Event Flag Clear Reg., Group x

GxSEFCLR (x=0-11)

Source Event Flag Clear Reg., Group x **(0598_H+x*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

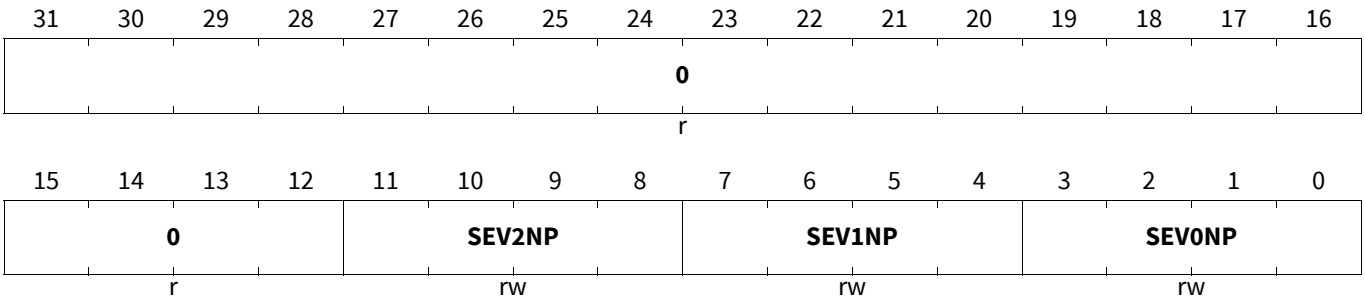
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													SEV2	SEV1	SEV0
r													w	w	w

Field	Bits	Type	Description
SEVi (i=0-2)	i	w	**Clear Source Event i** 0_B No action 1_B Clear the source event flag in GxSEFLAG
0	31:3	r	**Reserved, write 0, read as 0**

Source Event Node Pointer Reg., Group x

GxSEVNP (x=0-11)

Source Event Node Pointer Reg., Group x (05C0$_H$+x*400$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SEViNP (i=0-2)	4*i+3:4*i	rw	**Service Request Node Pointer Source Event i** Routes the corresponding event trigger to one of the service request lines (nodes). *Note: For shared service request lines see common groups in the product-specific appendix.* Not listed combinations are reserved. 0_H Select service request line 0 of group x … 3_H Select service request line 3 of group x 4_H Select shared service request line 0 … 7_H Select shared service request line 3 F_H No service request line selected
0	31:12	r	**Reserved, write 0, read as 0**

32.14.2 Channel Event Flag Registers, Group x

These registers handle the events generated by conversions, in particular during limit checking.

Note: Software can set all flags in register GxCEFLAG and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear all flags in register GxCEFLAG by writing 1 to the respective bit in register GxCEFCLR.

Channel Event Flag Register, Group x

The layout of this register depends on the multiplexer configuration.

Primary groups have 8 channels and, consequently, 8 channel event flags (CEV7 ... CEV0).

Secondary groups have 16 channels / channel event flags (CEV15 ... CEV0).

GxCEFLAG (x=0-7)

Channel Event Flag Register, Group x (0580_H+x*400_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CEV7	CEV6	CEV5	CEV4	CEV3	CEV2	CEV1	CEV0
								rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CEVy (y=0-7)	y	rwh	**Channel Event for Channel y** 0_B No channel event 1_B A channel event has occurred
0	31:16	r	**Reserved, write 0, read as 0**

GxCEFLAG (x=8-11)

Channel Event Flag Register, Group x (0580_H+x*400_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEV15	CEV14	CEV13	CEV12	CEV11	CEV10	CEV9	CEV8	CEV7	CEV6	CEV5	CEV4	CEV3	CEV2	CEV1	CEV0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CEVy (y=0-15)	y	rwh	**Channel Event for Channel y** 0_B No channel event 1_B A channel event has occurred
0	31:16	r	**Reserved, write 0, read as 0**

Channel Event Flag Clear Register, Group x

The layout of this register depends on the multiplexer configuration.

Primary groups have 8 channels and, consequently, 8 channel event clear flags (CEV7 ... CEV0).

Secondary groups have 16 channels / channel event clear flags (CEV15 ... CEV0).

GxCEFCLR (x=0-7)

Channel Event Flag Clear Register, Group x (0590_H+x*400_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								CEV7	CEV6	CEV5	CEV4	CEV3	CEV2	CEV1	CEV0
								w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CEVy (y=0-7)	y	w	**Clear Channel Event for Channel y** 0_B No action 1_B Clear the channel event flag in GxCEFLAG
0	31:16	r	**Reserved, write 0, read as 0**

GxCEFCLR (x=8-11)

Channel Event Flag Clear Register, Group x (0590_H+x*400_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEV15	CEV14	CEV13	CEV12	CEV11	CEV10	CEV9	CEV8	CEV7	CEV6	CEV5	CEV4	CEV3	CEV2	CEV1	CEV0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CEVy (y=0-15)	y	w	**Clear Channel Event for Channel y** 0_B No action 1_B Clear the channel event flag in GxCEFLAG
0	31:16	r	**Reserved, write 0, read as 0**

Channel Event Node Pointer Reg. 0, Group x

GxCEVNP0 (x=0-11)

Channel Event Node Pointer Reg. 0, Group x(05A0$_H$+x*400$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CEV7NP				CEV6NP				CEV5NP				CEV4NP			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEV3NP				CEV2NP				CEV1NP				CEV0NP			
rw				rw				rw				rw			

Field	Bits	Type	Description
CEViNP (i=0-7)	4*i+3:4*i	rw	**Service Request Node Pointer Channel Event i** Routes the corresponding event trigger to one of the service request lines (nodes). *Note: For shared service request lines see common groups in the product-specific appendix.* Not listed combinations are reserved. 0_H Select service request line 0 of group x … 3_H Select service request line 3 of group x 4_H Select shared service request line 0 … 7_H Select shared service request line 3

Channel Event Node Pointer Reg. 1, Group x

GxCEVNP1 (x=0-11)

Channel Event Node Pointer Reg. 1, Group x(05A4$_H$+x*400$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CEV15NP				CEV14NP				CEV13NP				CEV12NP			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEV11NP				CEV10NP				CEV9NP				CEV8NP			
rw				rw				rw				rw			

Field	Bits	Type	Description
CEViNP (i=8-15)	4*i-29:4*i-32	rw	**Service Request Node Pointer Channel Event i** Routes the corresponding event trigger to one of the service request lines (nodes). *Note: For shared service request lines see common groups in the product-specific appendix.* Not listed combinations are reserved. 0_H Select service request line 0 of group x … 3_H Select service request line 3 of group x 4_H Select shared service request line 0 … 7_H Select shared service request line 3

32.14.3 Result Event Flag Registers, Group x

These registers handle the events generated by new result values being available in the result registers.

Note: Software can set all flags in register GxREFLAG and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear all flags in register GxREFLAG by writing 1 to the respective bit in register GxREFCLR.

Result Event Flag Register, Group x

GxREFLAG (x=0-11)

Result Event Flag Register, Group x (0584_H+x*400_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV15	REV14	REV13	REV12	REV11	REV10	REV9	REV8	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
REVy (y=0-15)	y	rwh	**Result Event for Result Register y** 0_B No result event 1_B New result was stored in register GxRESy
0	31:16	r	**Reserved, write 0, read as 0**

Result Event Flag Clear Register, Group x

GxREFCLR (x=0-11)

Result Event Flag Clear Register, Group x (0594_H+x*400_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV15	REV14	REV13	REV12	REV11	REV10	REV9	REV8	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
REVy (y=0-15)	y	w	**Clear Result Event for Result Register y** 0_B No action 1_B Clear the result event flag in GxREFLAG
0	31:16	r	**Reserved, write 0, read as 0**

Result Event Node Pointer Reg. 0, Group x

GxREVNP0 (x=0-11)

Result Event Node Pointer Reg. 0, Group x (05B0$_H$+x*400$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV7NP				REV6NP				REV5NP				REV4NP			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV3NP				REV2NP				REV1NP				REV0NP			
rw				rw				rw				rw			

Field	Bits	Type	Description
REViNP (i=0-7)	4*i+3:4*i	rw	**Service Request Node Pointer Result Event i** Routes the corresponding event trigger to one of the service request lines (nodes). *Note: For shared service request lines see common groups in the product-specific appendix.* Not listed combinations are reserved. 0_H Select service request line 0 of group x … 3_H Select service request line 3 of group x 4_H Select shared service request line 0 … 7_H Select shared service request line 3

Result Event Node Pointer Reg. 1, Group x

GxREVNP1 (x=0-11)

Result Event Node Pointer Reg. 1, Group x (05B4$_H$+x*400$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV15NP				REV14NP				REV13NP				REV12NP			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV11NP				REV10NP				REV9NP				REV8NP			
rw				rw				rw				rw			

Field	Bits	Type	Description
REViNP (i=8-15)	4*i-29:4*i-32	rw	**Service Request Node Pointer Result Event i** Routes the corresponding event trigger to one of the service request lines (nodes). *Note: For shared service request lines see common groups in the product-specific appendix.* Not listed combinations are reserved. 0_H Select service request line 0 of group x … 3_H Select service request line 3 of group x 4_H Select shared service request line 0 … 7_H Select shared service request line 3

32.14.4 Global Event Flag Registers

These registers handle the events generated by new results being available in the global result register.

Note: Software can set flag REVGLB and trigger the corresponding event by writing 1 to REVGLB. Writing 0 has no effect.
Software can clear this flag by writing 1 to bit REVGLBCLR.
Setting both bits simultaneously clears the flag.

Global Event Flag Register

GLOBEFLAG
Global Event Flag Register (00E0$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							REVGLBCLR	0							
r							w	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							REVGLB	0							
r							rwh	r							

Field	Bits	Type	Description
REVGLB	8	rwh	**Global Result Event** 0_B No result event 1_B New result was stored in register GLOBRES
REVGLBCLR	24	w	**Clear Global Result Event** 0_B No action 1_B Clear the result event flag REVGLB
0	7:0, 23:9, 31:25	r	**Reserved, write 0, read as 0**

Global Event Node Pointer Register

GLOBEVNP
Global Event Node Pointer Register (0140$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												REV0NP			
r												rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
REV0NP	19:16	rw	**Service Request Node Pointer Global Result** Routes the corresponding event trigger to one of the service request lines (nodes). *Note: For shared service request lines see common groups in the product-specific appendix.* Not listed combinations are reserved. 0_H Select shared service request line 0 of common service request group 0 … 3_H Select shared service request line 3 of common service request group 0 4_H Select shared service request line 0 of common service request group 1 … 7_H Select shared service request line 3 of common service request group 1
0	15:0, 31:20	r	**Reserved, write 0, read as 0**

32.14.5 Software Activation of Service Requests, Group x

This register provides software control for the service request lines.

Service Request Software Activation Trigger, Group x

Each service request can be activated via software by writing "1" to the corresponding bit in register GxSRACT. This can be used for evaluation and testing purposes.

Note: For shared service request lines see common groups in the product-specific appendix.

GxSRACT (x=0-11)

Service Request Software Activation Trigger, Group x($05C8_H$+x*400_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				ASSR3	ASSR2	ASSR1	ASSR0	0				AGSR3	AGSR2	AGSR1	AGSR0
r				w	w	w	w	r				w	w	w	w

Field	Bits	Type	Description
AGSRy (y=0-3)	y	w	**Activate Group Service Request Node y** 0_B No action 1_B Activate the associated service request line
ASSRy (y=0-3)	y+8	w	**Activate Shared Service Request Node y** 0_B No action 1_B Activate the associated service request line
0	7:4, 31:12	r	**Reserved, write 0, read as 0**

32.14.6 Service Requests for Fast Compare Channels

Each Fast Compare Channel can issue service requests via a dedicated service request line. Each service request can be activated by several configurable events.

The active event is selected by bitfields SRG in register **FCxFCM (x=0-7)** and CHEVMODE in register **FCxFCCTRL (x=0-7)**.

The figure below summarizes the configuration of Fast Compare Channel events to trigger a service request.

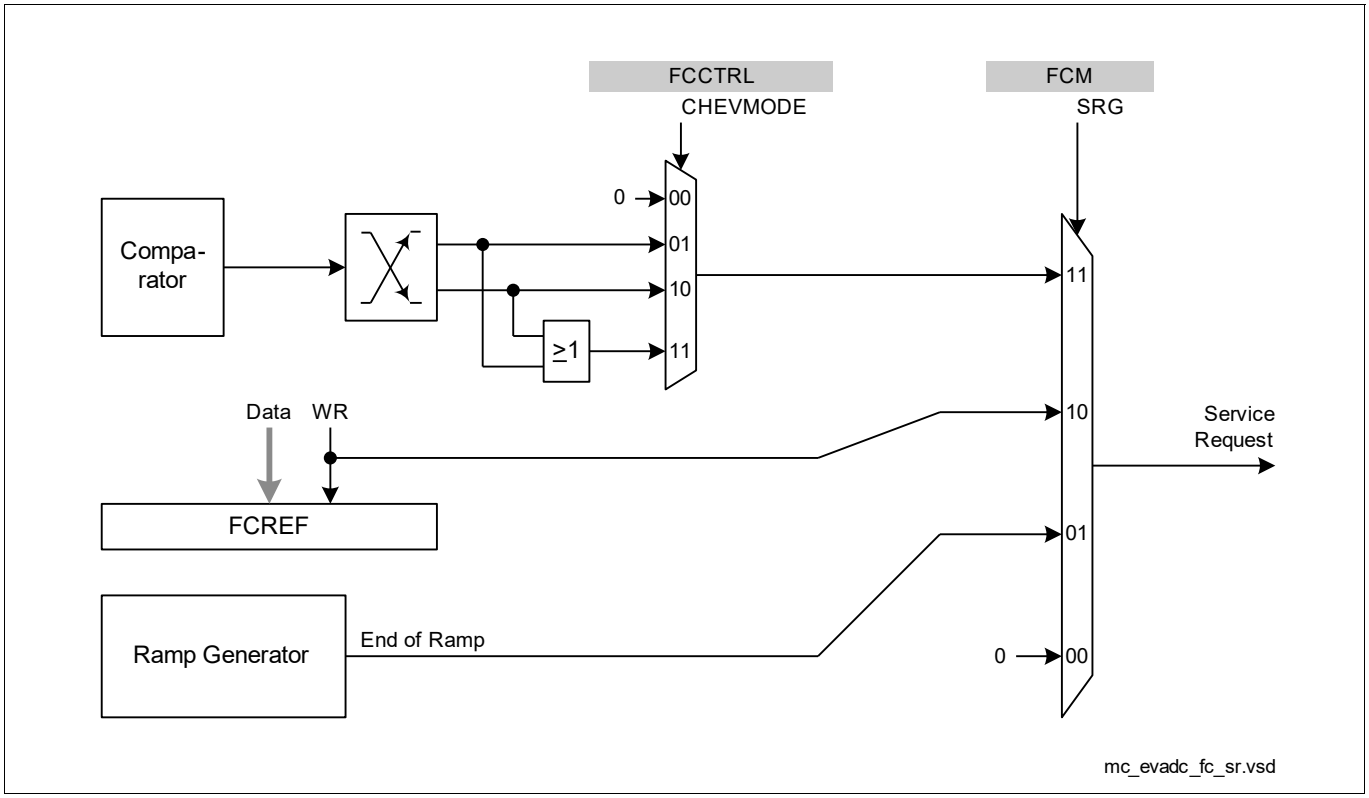


Figure 281 Fast Compare Channel Service Requests

32.15 Application Considerations

The operation of the EVADC and, hence, its behavior is programmable in a wide range. This makes it suitable for different applications while requiring a certain amount of initialization and/or handling during operation.

As far as possible the configuration options can be handled by initialization. Several functions, for example power control or calibration, have been automated so they require an absolute minimum of handling during operation.

Additional hints in section **"Changing the Configuration" on Page 9**.

32.15.1 Clock Synchronization

To eliminate the interference of concurrently operating ADC channels, the converters can operate in a synchronized way so each of them can reach its optimum performance. The phase synchronizer distributes a clock control signal which is used by the converters to start their operation. See **Section 32.3.5**.

After reset, the clock synchronization is active. Setting the phase synchronizer is mandatory for the EVADC to deliver its documented performance.

32.15.2 Calibration Recommendation

The actual calibration algorithm is executed automatically by hardware, so only a few configurations need to be selected before starting the algorithm.

Note: The calibration algorithm compensates manufacturing tolerances. It is, therefore, recommended to execute the start-up calibration once after a reset.

Before triggering the start-up calibration, enable the analog blocks and select the corresponding calibration sample time (CALSTC).

Attention: During the start-up calibration, all converters must be inactive. After reset this is the case anyway. Only start conversions after the calibration has completed (indicated by bit CAL = 0).

Trigger the start-up calibration by writing 1 to bit SUCAL in register **GLOBCFG**. The completion of the start-up calibration is indicated by bit CAL in register **GxARBCFG (x=0-11)**.

Calibration is done in steps smaller than LSB_{12}. Repeated calibration steps, therefore, may lead to reduced resolution due to the internally applied rounding.

In systems with a stable reference voltage, it is recommended to disable postcalibration permanently to avoid these effects by setting bit DPCAL in register **GxANCFG (x=0-11)**.

In systems where the reference voltage is influenced by temperature, postcalibration can be enabled temporarily during certain operation phases to compensate these effects.

32.15.3 Examples for Operation

The final configuration of the EVADC very much depends on the targeted application case. The following examples list some considerations to achieve stable operating modes.

- **Clocks**:
  In most cases the highest possible clock frequency shall be selected. The maximum analog clock frequency (DIVA) maximizes the performance and provides higher off-times to save energy when using the standby modes.
- **Sample time**:
  The minimum time required for sampling within the different clusters is specified in the corresponding Data Sheet.

If analog input precharge is used (AIPi $\neq 00_B$) the selected precharge time must be added to the minimum sample time. If the spread sample point feature is used (SESPi = 1) the maximum cut-off time must be added to the minimum sample time.

- **Noise handling**:
  Several features are provided to reduce the impact of noise (external or internal).
  The features are based on additional conversion steps or multiple conversions of the input signal. Computing averages from multiple conversion result, however, also requires increased overall conversion times.
  Therefore, the degree of usage depends on the properties of the input signal, i.e. its dynamic behavior.

32.15.4 Basic Initialization Sequence

After reset, the EVADC is disabled to minimize the initial power consumption. By executing the following steps the EVADC can be prepared for operation and be started:

Enable and configure the phase synchronizer according to the overall system requirements.

This is described in section “Application Considerations” in chapter CONVCTRL.

Enable the EVADC module and prepare it for operation

```
EVADC_CLC      = 0x00000000   ;Enable module EVADC
```

Enable a primary/secondary group and prepare it for operation

```
EVADC_GxANCFG  = 0x00300000   ;Analog clock frequency is 160 MHz / 4 = 40 MHz (example)
                              ;CALSTC = 00
EVADC_GxARBCFG= 0x00000003    ;Enable analog block
WAIT                          ;Pause for extended wakeup time (≥ 5 µs)
EVADC_GLOBCFG  = 0x80000000   ;Begin start-up calibration
                              ;(other operations can be executed in the meantime)
EVADC_GxARBPR  = 0x01000000   ;Enable arbitration slot 0
EVADC_GxQMR0   = 0x00000001   ;Enable request source 0
EVADC_GxICLASS0=0x00000002    ;Select 4 clocks for sampletime 4 / 40 MHz = 100 ns
                              ;The default setting stores results in GxRES0,
                              ;service requests are issued on GxSR0
EVADC_GxRCR0   = 0x80000000   ;Enable result service requests, if required
EVADC_GxQINR0  = 0x00000020   ;Request channel 0 in auto-repeat mode
WAIT                          ;Wait for start-up calibration to complete
                              ;(other operations can be executed in the meantime)
                              ;---> This starts continuous conversion of the channel
```

Note: Calibration start-up time tSUCAL or flag GxARBCFG.CAL=0

Enable fast compare channel and prepare it for operation

```
EVADC_FCxFCCTRL= 0x00000C00   ;Use default: Analog clock is 160 MHz / 4 = 40 MHz (ex.)
                              ;No trigger/gate, no service request, connect to input
EVADC_FCxFCBFL= 0x03010000    ;Output boundary flag directly on CBFLOUT3
EVADC_FCxFCHYST= 0x00A40050   ;Define hysteresis as +0.2 V / -0.1 V
EVADC_FCxFCM   = 0xA8E00421   ;Set reference to 3.3 V, no auto update, no service requ.,
                              ;trigger period 0.5 µs, enable analog block, start ch.
                              ;---> This starts continuous comparisons at the sel. rate
```

Retrieve Conversion Results

The selected channels will be converted/compared continuously now.

Conversions results of primary/secondary groups are indicated by service requests and can be read from GxRES0, compare results of fast compare channels are signaled on the boundary flags.

32.15.5 Module Handling in Sleep Mode

The EVADC does not change its operating mode in sleep mode. While sleep mode is evaluated (**CLC**.EDIS = 0, default after reset), the module clocks are stopped upon a sleep mode request. To achieve the power reduction that is usually intended during sleep mode, the application needs to disable the EVADC, or parts of it as required, before entering sleep mode.

*Note: If any activity is intended during sleep mode make sure that sleep mode requests are disregarded (**CLC**.EDIS = 1) and make sure the phase synchronizer is not disabled in this case.*

32.16 Electrical Models

Each conversion of an analog input voltage to a digital value consists of two consecutive phases:

- During the sample phase, the input voltage is sampled and stored.
  The **Input Signal Path** is a simplified model for this.
- During the conversion phase the stored voltage is converted to a digital result.
  The **Reference Voltage Path** is a simplified model for this.

Input Signal Path

The ADC uses a switched capacitor field represented by C_{AINS} (small parasitic capacitances are present at each input pin). During the sample phase, the capacitor field C_{AINS} is connected to the selected analog input CHx via the input multiplexer (modeled by ideal switches and series resistors R_{AIN}).

When precharge sampling is enabled (path (P)), the input charge consumption is defined by $Q_{AINS} = C_{SWT} \times V_{AINmax}$.

The switch to CHx is closed during the sample phase and connects the capacitor field to the input voltage V_{AINx}.

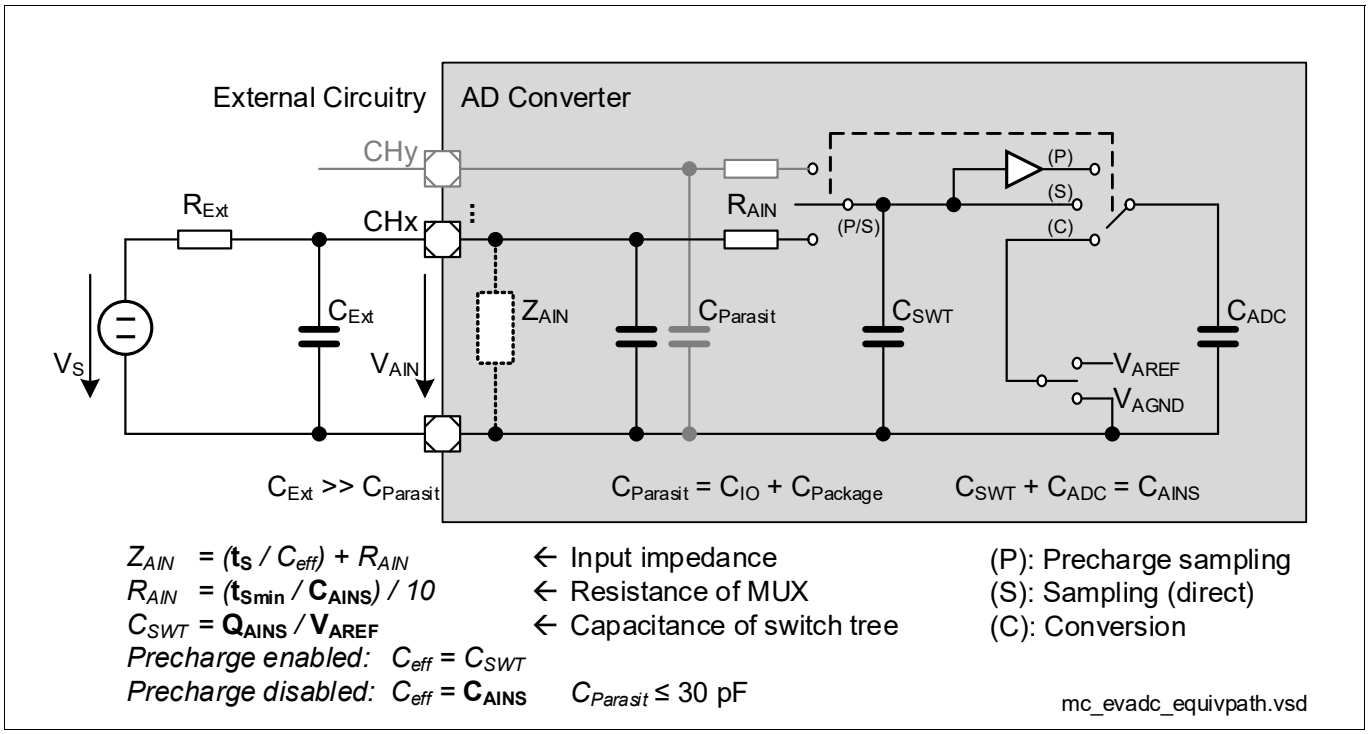


Figure 282 Signal Path Model

A simplified model for the analog input signal path is given in **Figure 282**. An analog voltage source (value V_S) with an internal impedance of R_{Ext} delivers the analog input that should be converted.

During the sample phase the corresponding switch is closed and the capacitor field C_{AINS} is charged. Due to the low-pass behavior of the resulting RC combination, the voltage at C_{ADC} to be actually converted does not immediately follow V_S. The value R_{Ext} of the analog voltage source and the desired precision of the conversion strongly define the required length of the sample phase.

To reduce the influence of R_{Ext} and to filter input noise, it is recommended to introduce a fast external blocking capacitor C_{Ext} at the analog input pin of the ADC. Like this, mainly C_{Ext} delivers the charge during the sample phase. This structure allows a significantly shorter sample phase than without a blocking capacitor, because the low-pass time constant defining the sample time is mainly given by the values of R_{AIN} and C_{AINS}.

The resulting low-pass filter of R_{EXT} (usually a parameter of the signal source) and C_{EXT} should be dimensioned according to the application's requirements:

- For quickly changing dynamic signals, a smaller capacitor allows V_{AINx} to follow V_S between two sample phases of the same analog input channel.
- For high-precision conversions, an external blocking capacitor C_{Ext} in the range of at least $2^n \times C_{AINS}$ keeps the voltage change of V_{AINx} during the sample phase below 1 LSB_n. This voltage change is due to the charge redistribution between C_{Ext} and C_{AINS}.

Leakage current through the analog input structure of the ADC can generate a voltage drop over R_{Ext}, introducing an error. The ADC input leakage current increases at high temperature and if the input voltage level is close to the analog supply ground V_{SSM} or to the analog power supply V_{DDM}. The input leakage current of an ADC channel can be reduced by avoiding input voltages close to the supplies.
An overload condition (input voltage exceeds the supply range) at adjacent analog inputs injects an additional leakage current (defined by a coupling factor) .

The capacitor C_{AINS} is automatically precharged to a voltage of approximately half the standard reference voltage V_{AREF} while the converter is idle and while idle precharge is enabled[1]. Due to varying parameters and parasitic effects, the precharge voltage of C_{AINS} is typically smaller than V_{AREF} / 2.

Note: When conversions are executed in a sequence, or when a conversion cancels a running conversion, the sample phase starts immediately.
The converter does not become idle in this case and C_{AINS} is not precharged!

Calculation Example

The following parameters are assumed as given from the Datasheet:

C_{AINS} = 3.4 pF (max. value), Q_{AINS} = 3.5 pC (primary group, max. value), V_{AREF} = 5.0 V, t_{Smin} = 100 ns (primary group).

The following parameters can be calculated, as indicated in **Figure 282**.

$R_{AIN} = t_{Smin} / C_{AINS}$ = 100 ns / 3.4 pF / 10 = 2.94 kOhm

$C_{SWT} = Q_{AINS} / V_{AREF}$ = 3.5 pC / 5.0 V = 0.7 pF

The input impedance, therefore, can be calculated as:

Z_{AIN} = 100 ns / 0.7 pF + 2.94 kOhm = 145.8 kOhm (with sample precharge enabled)

Z_{AIN} = 100 ns / 3.4 pF + 2.94 kOhm = 32.4 kOhm (without sample precharge, min. sample time)

Z_{AIN} = 200 ns / 3.4 pF + 2.94 kOhm = 61.8 kOhm (without sample precharge, increased sample time)

If no external blocking capacitor C_{Ext} is used, the input channel sees the series resistance ($R_{Ext} + Z_{AIN}$) during the sample time. The analog input voltage, therefore, will be $V_{AIN} = V_S \times Z_{AIN} / (R_{Ext} + Z_{AIN})$.

The first example above then leads to $V_{AIN} = V_S \times 0.936$ (assuming R_{Ext} = 10 kOhm).

An average current that is drawn from the signal source when the corresponding input is converted repeatedly.

This current depends on the configured sampling mode (buffer on/off) and on the actual conversion rate i.e. the conversion cycle time t_{CT}. A conversion cycle time of t_{CT} = 10 µs leads to the following currents:

When sample precharging is disabled, the average current can be calculated as
$I_{avg} = (V_{AIN} / Z_{AIN}) \times (t_S / t_{CT})$ = (5.0 V / 32.4 kOhm) × (100 ns / 10 µs) = 154.3 µA × 0.01 = 1.54 µA (max. input voltage).
When sample precharging is enabled, the average current can be calculated as
$I_{avg} = (V_{AIN} / Z_{AIN}) \times (t_S / t_{CT})$ = (5.0 V / 145.8 kOhm) × (100 ns / 10 µs) = 0.35 µA,
or the average current can simply be calculated as
$I_{avg} = Q_{AINS} / t_{CT}$ = 3.5 pC / 10 µs = 0.35 µA.

1) Fast compare channels do not provide the idle precharge function.

Reference Voltage Path

During the conversion phase, parts of the capacitor field (represented by C_{AREFSW}) are switched to a reference input (V_{AREF} or CH0) or to V_{AGND}. Using CH0 as alternate reference source allows conversions of 5.0 V and 3.3 V based analog input signals with the same ADC kernel.

Stable and noise-free reference and analog supply voltages support accurate conversion results. Because noise can also be introduced from other modules (e.g. switching pins), it is strongly recommended to carefully decouple analog from digital signal domains.

The switching of parts of C_{AREFS} requires a dynamic current at the selected reference input. The impedance R_{RefExt} of the reference voltage source V_R has to be low enough to supply the reference current during the conversion phase. An external blocking capacitor C_{RefExt} can supply the peak currents and minimize the current to be delivered by the reference source.

The reference current I_{AREF} introduces a voltage drop at R_{RefExt} that should not be neglected for the calculation of the overall accuracy. The average reference current during a conversion depends on the reference voltage level and the time t_{CONV} between two conversion starts: $I_{AREF} = C_{AREFS} \times V_{AREF} / t_{CONV}$.

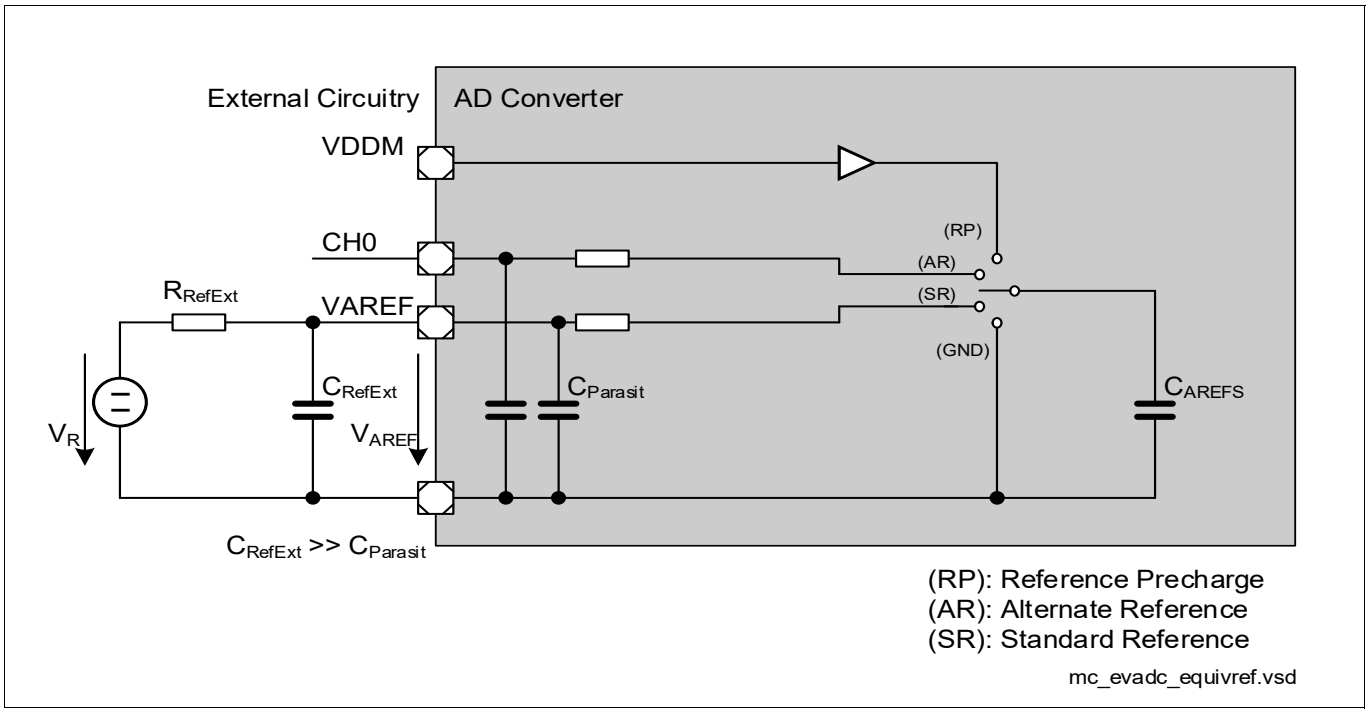


Figure 283 Reference Path Model

Reference Signal Connection

The reference voltages (V_{AREF}, V_{AGND}) have a strong influence on the digital result of a conversion. It is, therefore, recommended to carefully avoid noise on these inputs. Filter structures help to cancel high-frequency noise.

- Capacitors between corresponding reference pins (V_{AREF}, V_{AGND}) provide peak currents for the conversion steps
- Use low-ESR capacitors connected closely to the pins
- Connect reference pins via separate lines to star points close to the sensor supply
- Additional resistors in the reference lines (R_F) must be able to carry the average reference current, to avoid inaccuracy due to voltage drop
- The reference ground lines shall be connected to the common ground plane

Due to the charge redistribution between C_{REFEXT} and C_{AREFSW}, the voltage V_{AREF} decreases during the conversion phase. The error introduced by this effect is limited by the external blocking capacitor.
A blocking capacitor of $C \geq 2^n \times C_{AREFSW}$ limits this error to 1 LSB_n.

Assuming 12-bit conversions and C_{AREFSW} = 6 pF leads to: $C_{REFEXT} > 2^{12} \times 6$ pF = 24.6 nF per active converter.
The total required capacitor value depends on the number of simultaneously active converters:
Assuming 13 converters being active at the same time leads to: 24.6 nF × 13 = 320 nF.

Analog ground (V_{SSA}) and digital ground (V_{SSD}) are internally connected to an on-chip star point. These pins shall, therefore, be connected to a single common ground plane.

Note: A specific application note is available offering hints for the calculation of filter structures for the analog reference and the analog inputs. This application note also provides guidelines for PCB design.

Transfer Characteristics and Error Definitions

The transfer characteristic of the ADC describes the association of analog input voltages to the 2^n discrete digital result values (n bits resolution). Each digital result value (in the range of 0 to 2^n-1) represents an input voltage range defined by the reference voltage range divided by 2^n. This range (called quantization step or code width) represents the granularity (called LSB_n) of the ADC. The discrete character of the digital result generates a system-inherent quantization uncertainty of ±0.5 LSB_n for each conversion result.
The ideal transfer curve has the first digital transition (between 0 and 1) when the analog input reaches 0.5 LSB_n. The quantization steps are equally distributed over the input voltage range. Analog input voltages below or above the reference voltage limits lead to a saturation of the digital result at 0 or 2^n-1.

The real transfer curve can exhibit certain deviations from the ideal transfer curve:

- The **offset error** is the deviation of the real transfer line from the ideal transfer line at the lowest code. This refers to best-fit lines through all possible codes, for both cases.
- The **gain error** is the deviation of the slope of the real transfer line from the slope of the ideal transfer line. This refers to best-fit lines through all possible codes, for both cases.
- The **differential non-linearity error** (DNL) is the deviation of the real code width (variation of the analog input voltage between two adjacent digital conversion results) from the ideal code width.
- The **integral non-linearity error** (INL) is the deviation of the real transfer curve from an adjusted ideal transfer curve (same offset and gain error as the real curve, but equal code widths).
- The **total unadjusted error** (TUE) describes the maximum deviation between a real conversion result and the ideal transfer characteristics over a given measurement range. Since some of these errors noted above can compensate each other, the TUE value generally is much less than the sum of the individual errors.
  The TUE also covers production process variations.
- The **noise-induced error** (RMS) describes internal noise effects that need to be added to the TUE.
  External switching noise generated by the system may add an additional error.

32.17 Summary of Registers and Locations

The EVADC is built from a series of converter blocks that are controlled in an identical way. This makes programming versatile and scalable. The corresponding registers, therefore, have an individual offset assigned (see table below). The exact register location is obtained by adding the respective register offset to the base address (see product-specific appendix) of the corresponding group.

Due to the regular group structure, several registers appear within each group. Other registers are provided for each channel. This is indicated in the register overview table by formulas.

Registers with write access mode "...,M" can additionally be protected from unintended write access by setting the corresponding protection bit in register ACCPROTn. Refer to **"Register Access Control" on Page 21** for more details and an association table.

Table 269 Register Overview - EVADC (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	0000_H	U,SV	SV,E,P	Application Reset	12
ID	Module Identification Register	0008_H	U,SV	BE	PowerOn Reset	11
OCS	OCDS Control and Status Register	0028_H	U,SV	SV,P,OEN	See page 12	12
KRSTCLR	Kernel Reset Status Clear Register	$002C_H$	U,SV	SV,E,P	Application Reset	17
KRST1	Kernel Reset Register 1	0030_H	U,SV	SV,E,P	Application Reset	16
KRST0	Kernel Reset Register 0	0034_H	U,SV	SV,E,P	Application Reset	16
ACCEN0	Access Enable Register 0	$003C_H$	U,SV	SV,SE	Application Reset	14
GLOBCFG	Global Configuration Register	0080_H	U,SV	U,SV,P,M	Application Reset	19
ACCPROT0	Access Protection Register 0	0088_H	U,SV	SV,SE,P	Application Reset	21
ACCPROT1	Access Protection Register 1	$008C_H$	U,SV	SV,SE,P	Application Reset	22
ACCPROT2	Access Protection Register 2	0090_H	U,SV	SV,SE,P	Application Reset	22
GLOBICLASSi	Input Class Register i, Global	$00A0_H$+i*4	U,SV	U,SV,P	Application Reset	61
GLOBBOUND	Global Boundary Select Register	$00B8_H$	U,SV	U,SV,P	Application Reset	67
GLOBEFLAG	Global Event Flag Register	$00E0_H$	U,SV	U,SV,P	Application Reset	140
GLOBEVNP	Global Event Node Pointer Register	0140_H	U,SV	U,SV,P	Application Reset	140

Table 269 Register Overview - EVADC (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
GLOBTF	Global Test Functions Register	0160_H	U,SV	U,SV,P,M	Application Reset	119
GLOBTE	Global Test Enable Register	0164_H	U,SV	U,SV,P,M	Application Reset	121
GLOBRCR	Global Result Control Register	0280_H	U,SV	U,SV,P	Application Reset	95
GLOBRES	Global Result Register	0300_H	U,SV	U,SV,P	Application Reset	95
GLOBRESD	Global Result Register, Debug	0380_H	U,SV	U,SV,P	Application Reset	96
EMUXSEL	External Multiplexer Interface Select Register	$03F0_H$	U,SV	U,SV,P,M	Application Reset	128
GxTRCTR	Trigger Control Register, Group x	0410_H+x *400_H	U,SV	U,SV,P	Application Reset	49
GxARBCFG	Arbitration Config. Register, Group x	0480_H+x *400_H	U,SV	U,SV,P,M	Application Reset	52
GxARBPR	Arbitration Priority Register, Group x	0484_H+x *400_H	U,SV	U,SV,P,M	Application Reset	53
GxANCFG	Analog Fct. Config. Register, Group x	0488_H+x *400_H	U,SV	U,SV,P	Application Reset	27
GxICLASSi	Input Class Register i, Group x	$04A0_H$+x *400_H+i* 4	U,SV	U,SV,P,M	Application Reset	60
GxALIAS	Alias Register, Group x	$04B0_H$+x *400_H	U,SV	U,SV,P	Application Reset	64
GxBOUND	Boundary Select Register, Group x	$04B8_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	67
GxSYNCTR	Synchronization Control Register, Group x	$04C0_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	110
GxQCTRLi	Queue i Source Contr. Register, Group x	0500_H+x *400_H+i* 20_H	U,SV	U,SV,P	Application Reset	38
GxQMRi	Queue i Mode Register, Group x	0504_H+x *400_H+i* 20_H	U,SV	U,SV,P	Application Reset	40
GxQSRi	Queue i Status Register, Group x	0508_H+x *400_H+i* 20_H	U,SV	U,SV,P	Application Reset	41
GxQ0Ri	Queue i Register 0, Group x	$050C_H$+x *400_H+i* 20_H	U,SV	U,SV,P	Application Reset	44

Table 269 Register Overview - EVADC (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
GxQINRi	Queue i Input Register, Group x	0510_H+x *400_H+i* 20_H	U,SV	U,SV,P	Application Reset	42
GxQBURi	Queue i Backup Register, Group x	0514_H+x *400_H+i* 20_H	U,SV	U,SV,P	Application Reset	46
GxREQTMi	Queue i Requ. Timer Mode Reg., Group x	0518_H+x *400_H+i* 20_H	U,SV	U,SV,P	Application Reset	47
GxREQTSi	Queue i Requ. Timer Status Reg., Group x	$051C_H$+x *400_H+i* 20_H	U,SV	U,SV,P	Application Reset	48
GxCEFLAG	Channel Event Flag Register, Group x	0580_H+x *400_H	U,SV	U,SV,P,M	Application Reset	133
GxREFLAG	Result Event Flag Register, Group x	0584_H+x *400_H	U,SV	U,SV,P,M	Application Reset	137
GxSEFLAG	Source Event Flag Register, Group x	0588_H+x *400_H	U,SV	U,SV,P,M	Application Reset	131
GxCEFCLR	Channel Event Flag Clear Register, Group x	0590_H+x *400_H	U,SV	U,SV,P	Application Reset	134
GxREFCLR	Result Event Flag Clear Register, Group x	0594_H+x *400_H	U,SV	U,SV,P	Application Reset	137
GxSEFCLR	Source Event Flag Clear Reg., Group x	0598_H+x *400_H	U,SV	U,SV,P	Application Reset	131
GxCEVNP0	Channel Event Node Pointer Reg. 0, Group x	$05A0_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	135
GxCEVNP1	Channel Event Node Pointer Reg. 1, Group x	$05A4_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	135
GxREVNP0	Result Event Node Pointer Reg. 0, Group x	$05B0_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	138
GxREVNP1	Result Event Node Pointer Reg. 1, Group x	$05B4_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	138
GxSEVNP	Source Event Node Pointer Reg., Group x	$05C0_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	132
GxSRACT	Service Request Software Activation Trigger, Group x	$05C8_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	142
GxEMUXCTR	External Multiplexer Control Reg., Group x	$05F0_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	126
GxEMUXCS	Ext. Multiplexer Channel Select Reg., Group x	$05F4_H$+x *400_H	U,SV	U,SV,P,M	Application Reset	127

Table 269 Register Overview - EVADC (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
GxVFR	Valid Flag Register, Group x	$05F8_H$+x *400_H	U,SV	U,SV,P	Application Reset	97
GxCHCTRy	Group x, Channel y Control Register	0600_H+x *400_H+y* 4	U,SV	U,SV,P,M	Application Reset	58
GxRCRy	Group x Result Control Register y	0680_H+x *400_H+y* 4	U,SV	U,SV,P,M	Application Reset	90
GxRESy	Group x Result Register y	0700_H+x *400_H+y* 4	U,SV	U,SV,P,M	Application Reset	91
GxRESDy	Group x Result Reg. y, Debug	0780_H+x *400_H+y* 4	U,SV	U,SV,P	Application Reset	93
FCxFCCTRL	Fast Compare Control Register, FC Channel x	3400_H+x *100_H	U,SV	U,SV,P,M	Application Reset	74
FCxFCM	Fast Compare Mode Register, FC Channel x	3404_H+x *100_H	U,SV	U,SV,P,M	Application Reset	76
FCxFCRAMP0	Fast Compare Ramp Register 0, FC Channel x	3408_H+x *100_H	U,SV	U,SV,P,M	Application Reset	79
FCxFCRAMP1	Fast Compare Ramp Register 1, FC Channel x	$340C_H$+x *100_H	U,SV	U,SV,P,M	Application Reset	80
FCxFCBFL	Boundary Flag Register, FC Channel x	3420_H+x *100_H	U,SV	U,SV,P,M	Application Reset	84
FCxFCHYST	Fast Comp. Hysteresis Register, FC Channel x	3424_H+x *100_H	U,SV	U,SV,P,M	Application Reset	81

32.18 Revision History

This is a summary of the modifications that have been applied to this chapter.

Table 270 Revision History

Reference	Change to Previous Version	Comment
V3.0.0		
Page 15	Describe details of the module kernel reset.	
Page 9	Update description of configuration changing.	
Page 12	Add info about suspend state to register CLC.	
Page 13	Correct note about suspend mode at register OCS.	
Page 53	Changed description for bitfield 0 in register “GxARBPR (x=0-11)”.	
Page 69	Add detail to note about fast compare channel interval.	
Page 147	Add value for $C_{Parasit}$ to figure.	
Page 1	Line break typos removed.	
V3.0.1		
Page 118	Misleading info removed at the end of the paragraph.	
V3.0.2		
Page 145	Enhanced sequence “Enable a primary/secondary group and prepare it for operation”.	
Page 117	Updated footnote for G10CH15.	
Page 117	Updated description for GxCH28, GxCH30 and GxCH31.	
V3.0.3		
Page 25	Added “Minimum Input Buffering Time 200 ns”.	
V3.0.4		
Page 119, **Page 122**	Register name fixed.	
Page 25, **Page 144**	Value fixed.	
V3.0.5		
Page 100	Section “Result FIFO buffer timing” added.	

33 Enhanced Delta-Sigma Analog-to-Digital Converter (EDSADC)

The Enhanced Delta-Sigma Analog-to-Digital Converter module (EDSADC) provides a series of analog input channels connected to on-chip modulators using the Delta/Sigma (DS) conversion principle. Digital input channels accept data streams from external modulators (section summary on **Page 3**).

The on-chip demodulator channels convert these inputs to discrete digital values.

The number of inputs and EDSADC channels depends on the chosen product type. This is described in the product-specific appendix.

Each converter channel can operate independent of the others, controlled by a dedicated set of registers. The results of each channel can be stored in a dedicated channel-specific result register.

The on-chip filter stages generate digital results from the selected modulator signal.

The EDSADC accepts data from different types of external modulators. Their data streams can be fed through selectable input pins.

Also, on-chip modulators are available that accept differential or single-ended input signals.

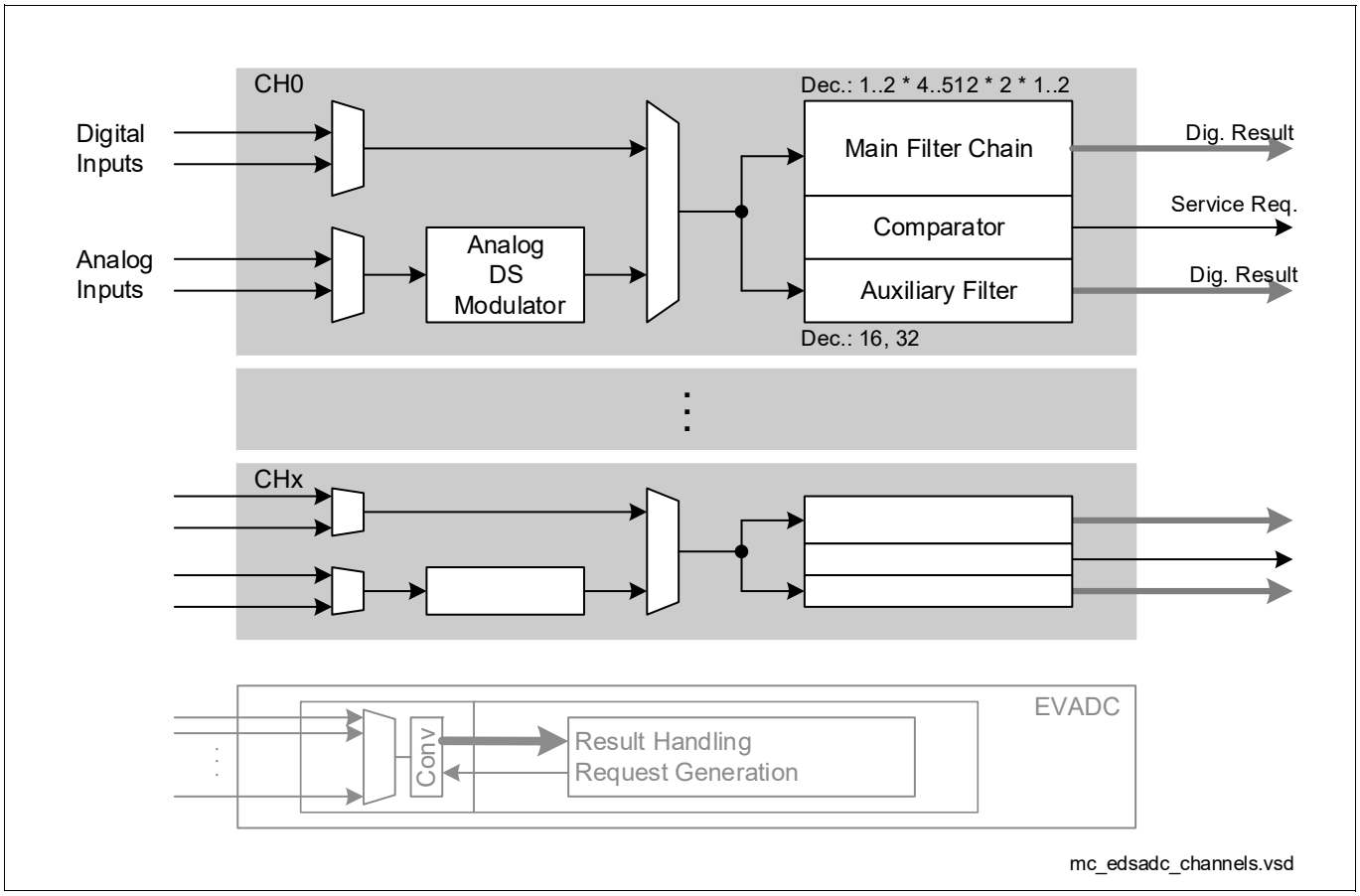


Figure 284 EDSADC Module Overview

33.1 Feature List

The following features describe the functionality of a DS Converter:

- Switched-capacitance input structure
  - Input voltage range 0…5 V
  - Common mode voltage: V_{AREF} / x (x = 2, 4, 8), connection selectable for each pin
  - Programmable gain (1:1 / 1:2 / 1:4)
  - DC Offset <5 mVDC, gain error < ±0.2% (calibrated, on product level)
  - Equivalent input impedance 500 kΩ typ. (for gain = 1, f_{MOD} = 26.67 MHz)
- On-chip modulator, single-bit second-order feed-forward modulator, sample frequency 16 … 40 MHz
- Options to connect external standard DS modulators
  - Selectable data stream inputs
  - Selectable DS clock input or output
- Demodulator (concatenated hardware filter stages)
  - Configurable CIC filter with decimation rates of 4…512
  - Overshoot compensation filter, optional
  - FIR filter with 8 coefficients (10-bit) with decimation rate 2 (FIR0)
  - FIR filter with 28 coefficients (10-bit) with decimation rate 1 or 2 (FIR1)
  - Pass Band 0.723 … 100 kHz, output sampling rate f_d = 2.17 … 300 kHz (decim. rate FIR1 = 2:1),
    Pass Band 0.723 … 10 kHz, output sampling rate f_d = 4.34 … 60 kHz (decim. rate FIR1 = 1:1),
    Pass band ripple: df_d < ±1%
  - Stop band attenuation: 0.5…1 × f_d: >40 dB / 1…1.5 × f_d: >45 dB /
    1.5…2 × f_d: > 50 dB / 2…2.5 × f_d: >55 dB / 2.5…OSR/2 × f_d: > 60 dB
  - Limit checking support
- Optional high-pass filter for DC compensation ($f_{-3dB} = 10^{-5} \times f_d$), configurable
- Hardware offset/gain calibration and compensation
- Parallel auxiliary CIC filter with limit checking for alarm generation
- Support for resolver applications
  - Carrier signal generator (differential sine output)
  - Signal evaluation including rectification, delay compensation and carrier elimination
- Automatic limit checking
  - Two-level boundary comparator
  - Separate indication signals

33.2 Overview

The Delta-Sigma Analog to Digital Converter module provides several channels providing an on-chip modulator with an associated demodulator including a configurable filter chain for demodulation, decimation, and filtering (see **Figure 285**).

You will find the following major sections within this chapter:

Attention: ***This chapter describes the EDSADC of the TC3XX family, encompassing the features and functions of all family members.***
The specific characteristics of a product are described in the product-specific appendix.
This product-specific appendix specifies deviations (e.g. downgrades, etc.) from this family documentation.

Table 271 Abbreviations used in EDSADC

Abbreviation	Meaning
ADC	Analog to Digital Converter
CIC	Cascaded Integrator Comb (filter)
DMA	Direct Memory Access (controller)
DNL	Differential Non-Linearity (error)
DS	Delta-Sigma (conversion principle)
EVADC	Enhanced Versatile Analog-Digital Converter
FIR	Finite Impulse Response (filter)
HDI	Hardware Data Interface
INL	Integral Non-Linearity (error)
LSB_n	Least Significant Bit: finest granularity of the analog value in digital format, represented by one least significant bit of the conversion result with n bits resolution (measurement range divided in 2^n equally distributed steps)

Table 271 Abbreviations used in EDSADC (cont'd)

Abbreviation	Meaning
OSR	Oversampling Ratio
PWM	Pulse Width Modulation

The EDSADC is connected to other on-chip modules to support functions on system level (see **Figure 286**).

The on-chip Single-Bit Second-Order Feed-Forward modulator converts an analog input signal to a datastream.

Several types of external modulators can be connected to the input path. The modulator clock signal can be generated internally or can be fed from an external clock source.

The digital filter chain (see **Figure 285**) builds the demodulator which produces result values at a configurable output rate. The elements of the filter chain can be activated according to the requirements of the application. The filter chain configuration determines the attenuation and delay properties of the filter. The decimation at a configurable rate reduces the modulator's input sampling rate to a lower result data rate that is suitable for the application.

The CIC filter provides the basic filtering and decimation with a selectable decimation rate.

Two FIR filters, each with a decimation rate of 2 (also 1 for FIR1), allow effective signal shaping by attenuating the upper frequencies of the signal spectrum.

The high-pass filter provides offset compensation by removing the DC component of the input signal.

The integrator accumulates a configurable amount of result values. The number of samples is programmable by software or can be controlled by a hardware signal.

Functionality of the integrator:

- further reduction of the data output rate for the application
- calculation of average values with selectable start point e.g. for shunt current measurement applications
- support for resolver applications to get the baseband signal for the motor position calculation

Each channel can generate service requests to trigger DMA transfers or to request CPU service.

Offset calibration and gain calibration are done in hardware. The calibration values are determined automatically by a hardware algorithm. This can be triggered initially after a reset or repeatedly during operation.

The on-chip carrier signal generator produces a selectable output signal (sine, triangle, rectangle) which can be used to drive a resolver. Synchronization of each input signal to the carrier signal ensures correct integration of the resolver input signals.

The basic module clock is the peripheral clock signal f_{ADC}.

A comparator supports limit checking, e.g. for overcurrent detection. Two limit values can be defined to restrict the generation of service requests to result values within a configurable area. This saves CPU performance and/or DMA bandwidth while continuously monitoring the input signal.

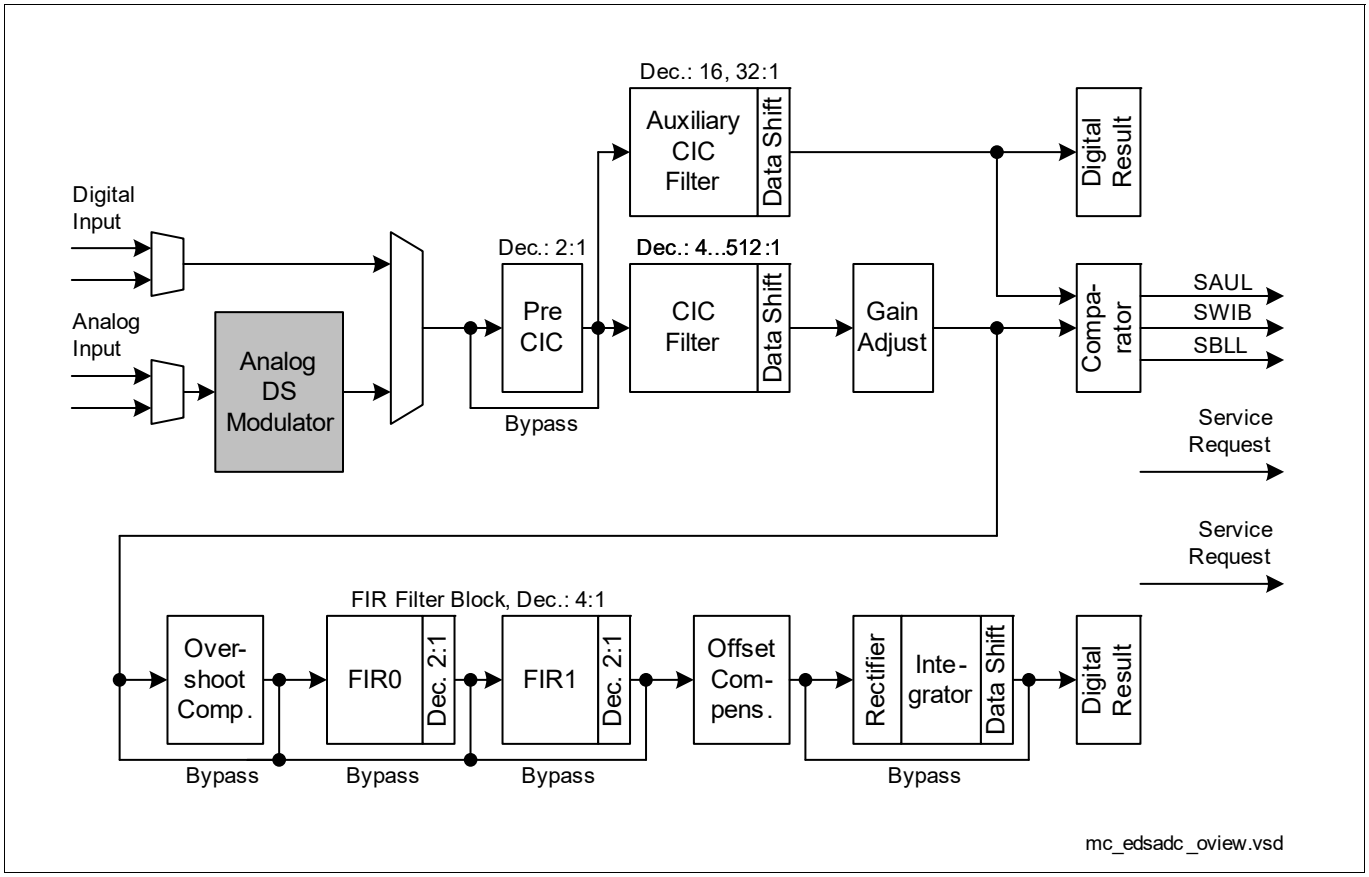


Figure 285 EDSADC Structure Overview

System Integration and Communication

Besides the connection to the peripheral bus, the EDSADC is directly connected to other hardware modules of the AURIX™ TC3xx Platform. **Figure 286** shows an overview of these interconnections. More detailed information can be found within the family documentation of the EDSADC as well as of the connected modules.

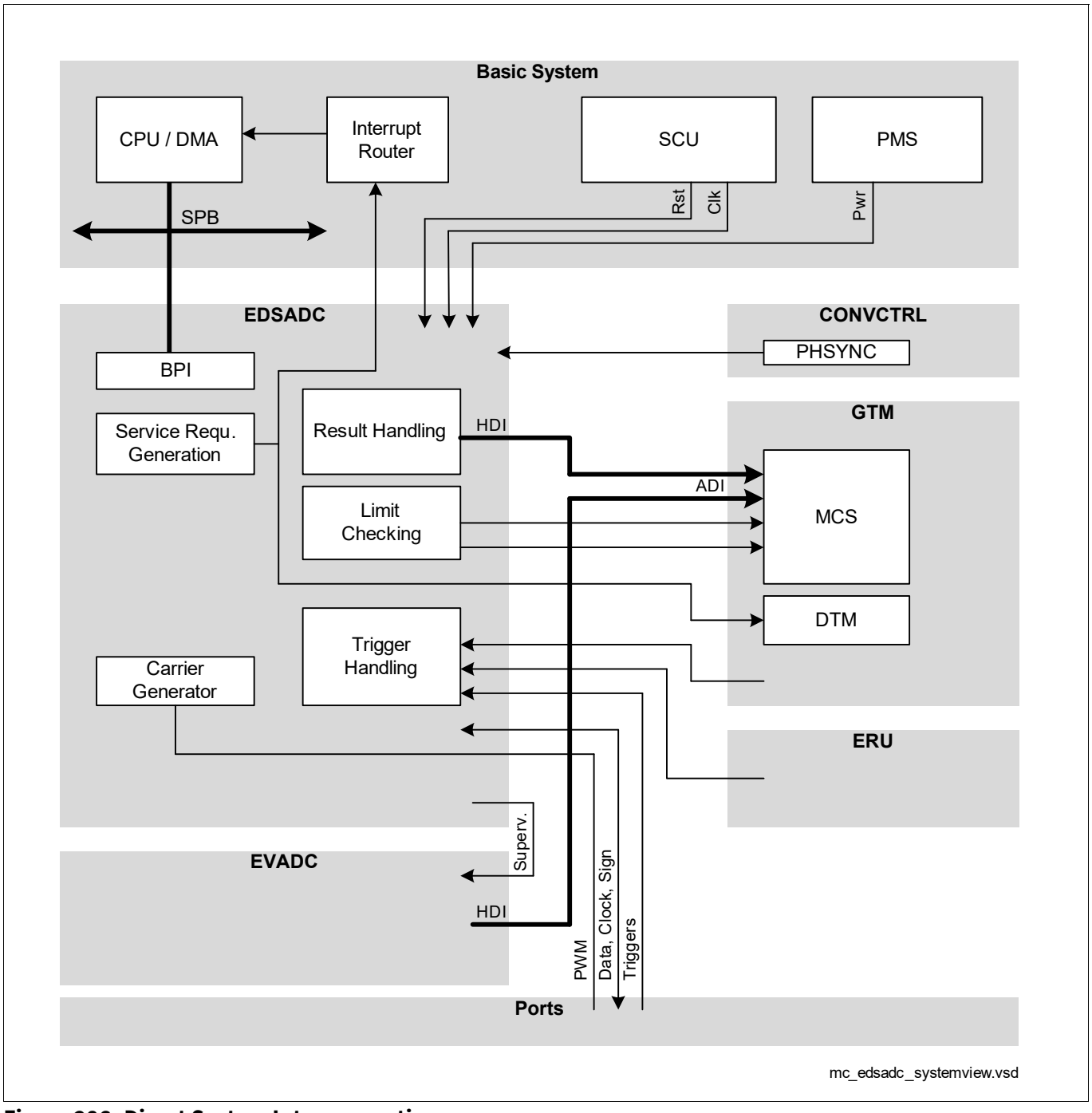


Figure 286 Direct System Interconnections

33.3 Configuration of General Functions

Several parameters can be configured to adapt the functionality of the EDSADC to the requirements of the actual application (see **Section 33.3.5**).

All configurations become effective when the corresponding channel is started, i.e. MxRUN becomes 1 for modulator-specific settings, CHxRUN becomes 1 for demodulator-specific settings.

33.3.1 Changing the Configuration

The configuration bitfields control the operation of the functional blocks of the EDSADC. To ensure proper operation and interaction of these functional blocks, it is recommended to change the configuration parameters of a channel only while this channel is inactive.
This means: MxRUN = CHxRUN = 0_B for channel x.

After reset, this is the default state. When changing the configuration during run-time, be aware of the respective implications listed in the table below.

Table 272 Implications When Changing the Configuration

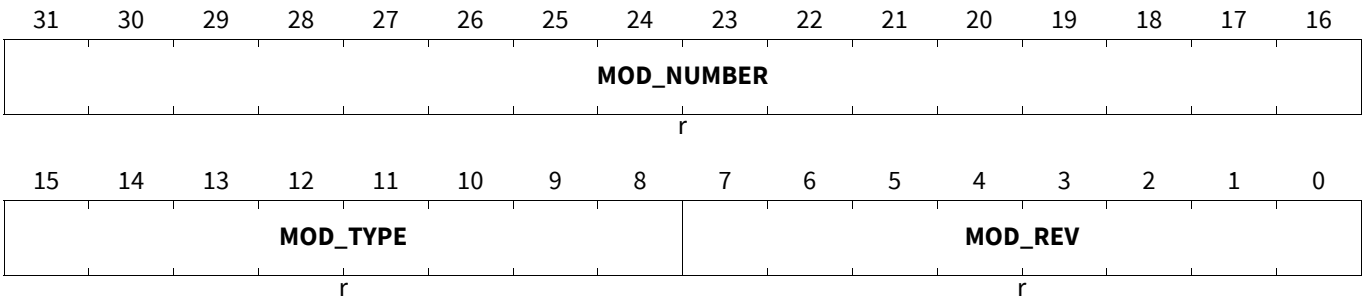
Configuration	Implication
Change during calibration	Wrong calibration or calibration error (to be avoided)
Modulator properties (INCFGP, INCFGN, GAINSEL)	Changes only become active after switching the analog multiplexer setting (if available) or after restarting the modulator (MxRUN = 1). Other modifications become effective immediately.
Change during operation	A change can cause resettling of modulator or filter chain, leading to delayed effect on the result values.

33.3.2 Module Identification

The module identification register indicates the version of the EDSADC module that is used in this product.

Module Identification Register

ID
Module Identification Register **(0008_H)** **PowerOn Reset Value: $00C6\ C007_H$**



Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision** Indicates the revision number of the implementation. This information depends on the design step.
MOD_TYPE	15:8	r	**Module Type** This internal marker is fixed to $C0_H$.

Field	Bits	Type	Description
MOD_NUMBER	31:16	r	**Module Number** Indicates the module identification number ($00C6_H$ = EDSADC)

33.3.3 System Registers

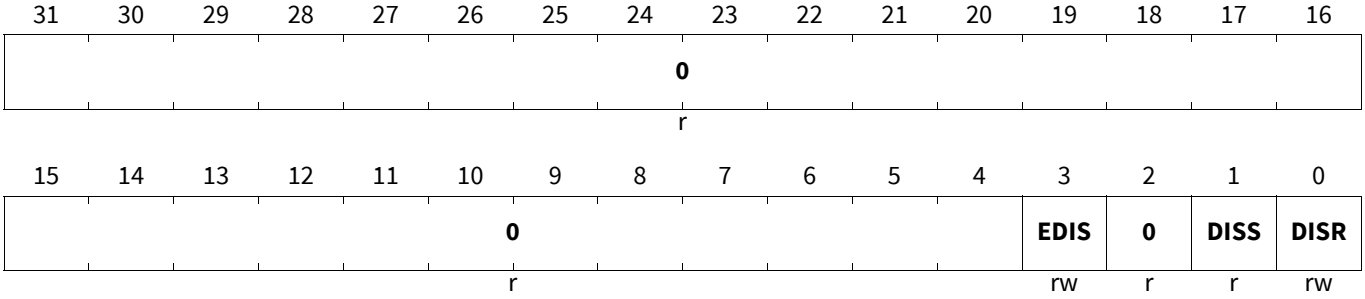
A set of standardized registers provides general access to the module and controls basic system functions.

Clock Control Register

The Clock Control Register **CLC** allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application.

Register **CLC** controls the module clock signal and the reactivity to the sleep signal.

CLC
Clock Control Register **(0000_H)** **Application Reset Value: $0000\ 0003_H$**



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B On request: enable the module clock 1_B Off request: stop the module clock
DISS	1	r	**Module Disable Status Bit** 0_B Module clock is enabled 1_B Off: module is not clocked f_{SPB} and f_{ADC} are disabled
EDIS	3	rw	**Sleep Mode Enable Control** Used to control the module's reaction to sleep mode. 0_B Sleep mode request is enabled and functional 1_B Module disregards the sleep mode control signal
0	2, 31:4	r	**Reserved, write 0**

It is recommended not to write to or read from module registers (except CLC) while the module is disabled. Write operations and read operations from registers that require a clock will generate a bus error.

OCDS Control and Status Register

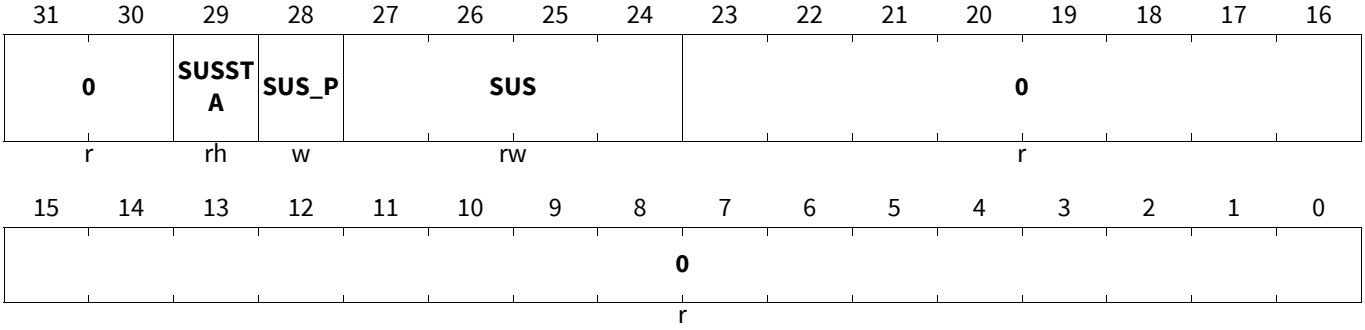
The OCDS Control and Status register OCS controls the module's behavior in suspend mode (used for debugging).

Register OCS is cleared by Debug Reset. It can only be written when OCDS is enabled.

If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective.

OCS

OCDS Control and Status Register **(0028_H)** **Reset Value: Table 274**



Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS). In soft suspend mode, the respective channel is stopped after the next result has been stored. For products with less channels the upper codes are reserved. 0_H Will not suspend 1_H Hard suspend: Clock is switched off immediately. 2_H Soft suspend channel 0 … F_H Soft suspend channel 13
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:0, 31:30	r	**Reserved, write 0, read as 0**

Table 273 Access Mode Restrictions of OCS sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **SUS_P**	rw	SUS	Set SUS_P during write access
(default)	r	SUS	

Table 274 Reset Values of OCS

Reset Type	Reset Value	Note
PowerOn Reset	$0000\ 0000_H$	
Debug Reset	$0000\ 0000_H$	

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions with the on-chip bus master TAG IDs 00 0000_B to 01 1111_B (see On-Chip Bus chapter for the mapping of product TAG ID <-> master peripheral). Register ACCEN0 provides one enable bit for each possible TAG ID encoding. Register ACCEN0 itself is protected by the Safety Endinit feature.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 00 0000_B, EN1 -> TAG ID 00 0001_B , ... , EN31 -> TAG ID 01 1111_B (TAG IDs 1X $XXXX_B$ are not used).

ACCEN0

Access Enable Register 0 **($003C_H$)** **Application Reset Value: FFFF $FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B No write access 1_B Write access will be executed

Individual Module Reset

The Kernel Reset Registers **KRST0**/**KRST1** are used to reset the related module kernel. Kernel Reset Registers 0 and 1 each include bit RST. To reset a module kernel it is necessary to set the RST bits in both Kernel Reset Registers. They will be cleared automatically with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set in the same clock cycle the RST bit is cleared. This bit indicates that a kernel reset was processed. Bit RSTSTAT can be cleared by setting bit CLR in register **KRSTCLR**.

A module kernel reset has the following effects:

Table 275 Effects of a Module Kernel Reset

Register	Executed Action
CLC, OCS, ACCEN0	No influence
ID	No influence
KRST0	Bit RST is cleared automatically after reset execution, RSTSTAT indicates a module kernel reset, cleared via KRSTCLR
KRST1	Bit RST is cleared automatically after reset execution

1) The Access Enable functionality controls only write transactions to registers CLC, OCS, KRSTx, and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

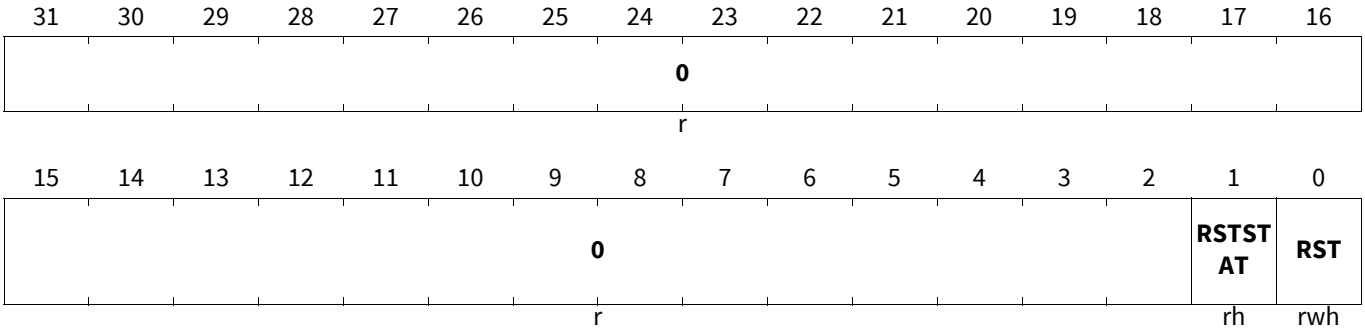
Table 275 Effects of a Module Kernel Reset (cont'd)

Register	Executed Action
KRSTCLR	No influence
Other registers	Reset to their defined reset values

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

Kernel Reset Register 0

KRST0
Kernel Reset Register 0 (0034_H) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** Request a kernel reset. The reset is executed if the reset bits of both kernel reset registers are set. RST is cleared after the kernel reset was executed. 0_B No action 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** Indicates an executed kernel reset. RSTSTAT is set after the execution of a kernel reset in the same clock cycle in which the reset bits are cleared. Clear RSTSTAT by setting bit CLR in register KRSTCLR. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved, write 0, read as 0**

Kernel Reset Register 1

KRST1

Kernel Reset Register 1 **(0030_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															RST
r															rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** Request a kernel reset. The reset is executed if the reset bits of both kernel reset registers are set. RST is cleared after the kernel reset was executed. 0_B No action 1_B A kernel reset was requested
0	31:1	r	**Reserved, write 0**

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **($002C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															CLR
r															w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved, write 0, read as 0**

33.3.4 Register Access Control

Several protection schemes are provided to prevent unintended write access to control bitfields of the EDSADC.

- The registers of the EDSADC are protected by the general access control mechanism that is configured by register **ACCEN0**.
- A specific register access control scheme provides a versatile protection scheme against unintended corruption of register contents. Register ACCPROT allows the restriction of write accesses for several groups of registers. The registers to be protected can be selected by the user. **Table 276** lists the registers that belong to each register group.
  Register ACCPROT itself is protected by the Safety Endinit feature.
- Groups of bitfields within a register may also be protected by an associated write control bit. This write control bit (xxWC) must be written with 1 along with the write access to the intended bitfield(s).

Access Protection Register

ACCPROT
Access Protection Register **(0090_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RGG							**0**								**RG10**
rw							r								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								**RG07**	**RG06**	**RG05**	**RG04**	**RG03**	**RG02**	**RG01**	**RG00**
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RG0x (x=0-7)	x	rw	**Register Group x** 0_B Full access to register group x 1_B Write access to registers of group x is blocked
RG10	16	rw	**Register Group 10** 0_B Full access to register group 10 1_B Write access to registers of group 10 is blocked
RGG	31	rw	**Register Group Global** 0_B Full access to global register group 1_B Write access to global registers is blocked
0	15:8, 30:17	r	**Reserved, write 0, read as 0**

Table 276 Register Protection Groups

Control Bit	Registers<	Notes
RG00	**MODCFGx (x=0-13)**	Modulator control
RG01	**VCMx (x=0-13)**	Common mode voltage control
RG02	**DICFGx (x=0-13)**	Input path control
RG03	**FCFGMx (x=0-13)**, **FCFGCx (x=0-13)**, **OVSCFGx (x=0-13)**	Filter control
RG04	**IWCTRx (x=0-13)**, **RFCx (x=0-13)**, **TSCNTx (x=0-13)**	Integrator and FIFO control
RG05	**CGCFG**, **RECTCFGx (x=0-13)**, **CGSYNCx (x=0-13)**	Carrier generator control
RG06	**GAINCORRx (x=0-13)**, **GAINCALx (x=0-13)**, **GAINCTRx (x=0-13)**, **OFFCOMPx (x=0-13)**	Calibration control
RG07	**EVFLAG**, **EVFLAGCLR**	Service request control
RG10	**BOUNDSELx (x=0-13)**, **FCFGAx (x=0-13)**	Limit checking, aux. filter control
RGG	**GLOBCFG**, **GLOBRC**	Global control

33.3.5 Global Configuration Registers

The EDSADC can operate in several configurations:

- Using the on-chip modulator, running from the internal clock
- Using an external modulator, running on a clock generated on-chip
- Using an external modulator, running on its own clock

The global configuration register GLOBCFG selects several operational or trimming values.

The global run control register GLOBRC controls the general operation of the available channels. For every EDSADC channel, register GLOBRC supports an individual bit for the related modulator (GLOBRC.MxRUN) and the related digital filter chain (GLOBRC.CHxRUN), where x depends on the number of implemented channels in the respective TC3x device. For applications where two or more EDSADC channels have to provide synchronous results, all related channels should be enabled synchronously using a single write access to register GLOBRC. This approach guarantees synchronization between EDSADC channels under all loading conditions of the system peripheral bus (SPB).

Global Configuration Register

GLOBCFG
Global Configuration Register (0080$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SVWC	0	SVSIG		SVCH				0							
w	r	rw		rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPWC	SUPLEV		USC	0	DITRIM			0							
w	rw		rw	r	rw			r							

Field	Bits	Type	Description
DITRIM	10:8	rw	**Trimming Value for the Dithering Function** This trim value is used for all modulators of the device. Not listed combinations are reserved. 000_B Minimum dithering intensity 001_B Low dithering intensity 011_B Medium dithering intensity 111_B High dithering intensity
USC	12	rw	**Unsynchronized Clock Generation** Defines the way the modulator clock is generated. 0_B Synchronized mode Rising clock edge is defined by the phase synchronizer. 1_B Unsynchronized mode The modulator clock is generated independently.

Field	Bits	Type	Description
SUPLEV	14:13	rw	**Supply Voltage Level** Adjusts the analog circuitry to the supply voltage used in the application system. Make sure to keep SUPLEV = 00_B or 01_B in the case of a 5 V supply. 00_B Automatic control: voltage range is controlled by the power supply 01_B Upper voltage range: assume a 5 V power supply is connected 10_B Lower voltage range: assume a 3.3 V power supply is connected 11_B Reserved
CPWC	15	w	**Write Control for Clock Parameters** 0_B No write access to clock parameters 1_B Bitfields SUPLEV, USC, DITRIM can be written
SVCH	27:24	rw	**Supervision Channel Select** Defines the channel for which the supervision signal selected by SVSIG is output. Not listed combinations are reserved. 0_H Supervision signal from channel 0 … D_H Supervision signal from channel 13
SVSIG	29:28	rw	**Supervision Signal Select** Defines the supervision signal of the channel selected by SVCH to be output. 00_B Off, no supervision signal 01_B 1.2 V supply voltage 10_B 3.3 V supply voltage 11_B Reserved
SVWC	31	w	**Write Control for Supervision Parameters** 0_B No write access to supervision parameters 1_B Bitfields SVSIG, SVCH can be written
0	7:0, 11, 23:16, 30	r	**Reserved, write 0, read as 0**

Table 277 Access Mode Restrictions of GLOBCFG sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **CPWC**	rw	DITRIM, SUPLEV, USC	Set CPWC during write access
write 1 to **SVWC**	rw	SVCH, SVSIG	Set SVWC during write access
(default)	r	DITRIM, SUPLEV, SVCH, SVSIG, USC	

Global Run Control Register

The register below shows the maximum configuration.

Other products of the family may have less channels and, consequently, less valid CHxRUN/MxRUN control bits.

GLOBRC

Global Run Control Register **(0088_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		M13RUN	M12RUN	M11RUN	M10RUN	M9RUN	M8RUN	M7RUN	M6RUN	M5RUN	M4RUN	M3RUN	M2RUN	M1RUN	M0RUN
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		CH13RUN	CH12RUN	CH11RUN	CH10RUN	CH9RUN	CH8RUN	CH7RUN	CH6RUN	CH5RUN	CH4RUN	CH3RUN	CH2RUN	CH1RUN	CH0RUN
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CHxRUN (x=0-13)	x	rw	**Channel x Run Control** Each bit (when set) enables the corresponding demodulator channel. When CHxRUN is set, all filter blocks are cleared and the FIFO is flushed. 0_B Stop channel x 1_B Demodulator channel x is enabled and runs
MxRUN (x=0-13)	x+16	rw	**Modulator x Run Control** Each bit (when set) enables the corresponding modulator. 0_B Stop clock for on-chip and external modulator x 1_B Modulator x is enabled and can run[1)]
0	15:14, 31:30	r	**Reserved, write 0, read as 0**

1) Additional run control via the selected gate signal is possible by the automatic power control feature (bitfield APC in register MODCFG).

33.4 Input Channel Configuration

The input data for a channel can be obtained from different sources:

- **On-Chip Modulator**: The on-chip modulator can directly convert a differential or single-ended analog input signal to an input data stream. The on-chip modulator runs on the internal clock.
  The figure below summarizes the data path through the on-chip modulator.
- **External Modulator Without Clock Source**: This type of modulator requires a modulator clock signal. This clock signal is provided by the EDSADC, the data stream produced by the modulator is input as a digital signal.
- **External Modulator With Clock Source**: This type of modulator generates the modulator clock signal along with the data stream. In this case, both the modulator clock and the data stream produced by the modulator are input as digital signals.

Note: An external modulator can be used, in particular, in systems where high voltages are to be sampled. This allows for galvanic decoupling.
Several input pins can be selected.

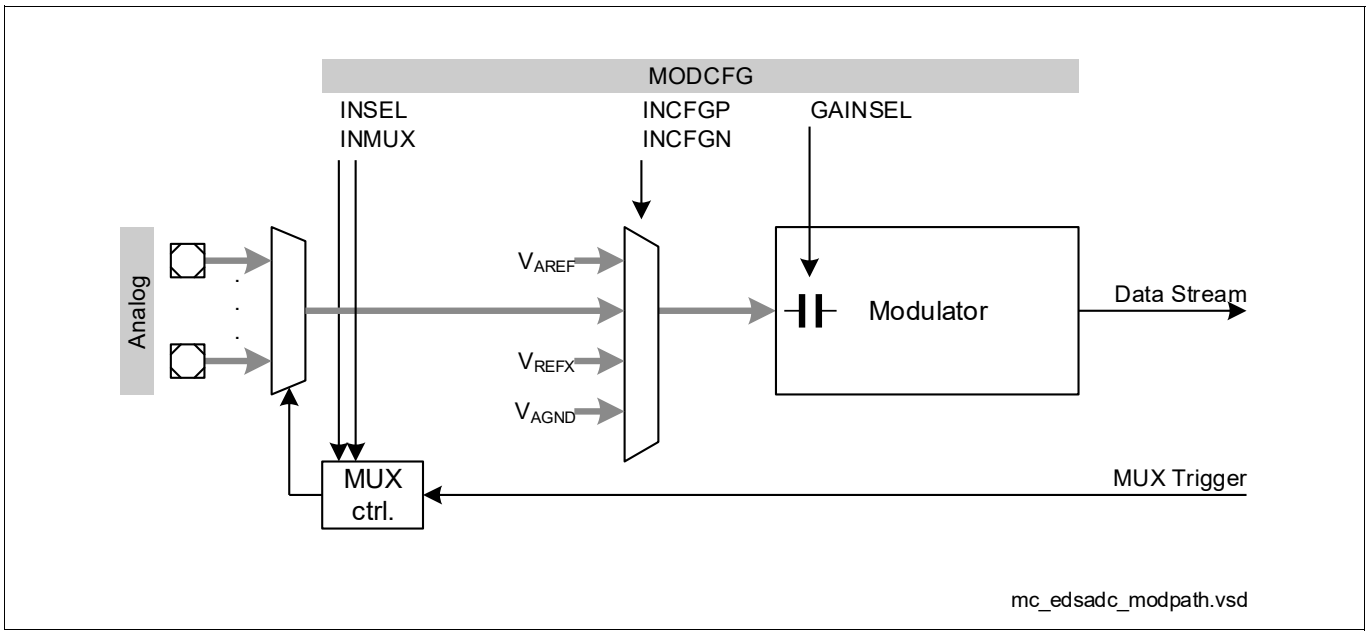


Figure 287 Analog Input Path

Note: To avoid clipping due to overdrive (the data stream contains all 1s) the on-chip modulator is built with an intrinsic gain factor of FM = 0.6945. This allows to handle a certain level of overshoots without entering overdrive.
The behavior of external modulators is described in the respective specification.

The modulator clock can be generated in different ways:

- The modulator clock is derived on-chip from the peripheral clock f_{ADC}. It is used for the on-chip modulators and can be output via a selectable modulator clock pin to be used by an external modulator. It also is the time basis for the timestamp counter.
- The external modulator can generate the clock signal which is then input via one of the modulator clock pins.

A trigger signal can be selected from different inputs. These inputs are connected to on-chip peripherals or to pins. The digital connection table in the product-specific appendix describes the available connections.

This trigger signal can be used for different purposes:

- Integration trigger:
  The external signal defines the integration window, i.e. the timespan during which result values are integrated.
- Timestamp trigger:
  The external signal requests the actualization of the timestamp register.
- Input multiplexer trigger:
  The external signal requests the switching of the analog input multiplexer to the next lower input or to the defined start value, respectively.
- Modulator run gate:
  The on-chip modulator can be controlled by an external gate signal while it is enabled. In this case it can be disabled temporarily to save power.
- Service request gate:
  Service requests for the filter chain can be restricted to the high or low times of the selected trigger signal.

Note: Trigger signals shall be active for at least 1 period of f_{ADC} to properly trigger an action.

The figure below summarizes these three signal paths and indicates the source of the corresponding control information.

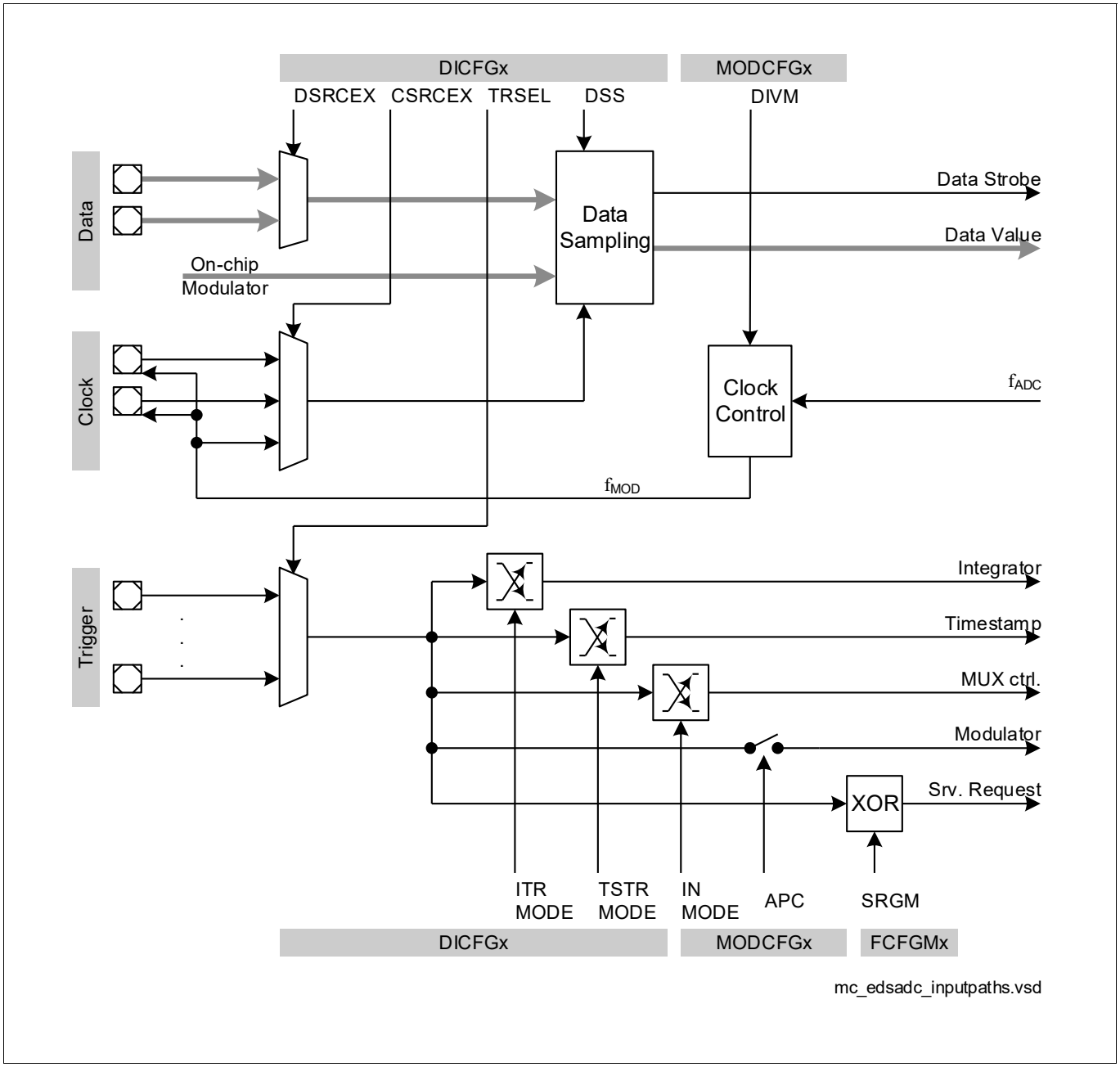


Figure 288 Input Path Summary

33.4.1 Modulator Clock Generation

The modulator clock signal can be generated on-chip or can be generated by an external modulator and be input through a pin (clock input).

The on-chip clock signal is used for the on-chip modulator or can be used for an external modulator through a selectable clock output pin. The on-chip modulator clock is based on the analog phase synchronizer. The phase synchronizer defines the rising edges of the modulator clock, the period is defined by bitfield DIVM (modulator clock period).

For proper operation the duty cycle of the on-chip modulator clock must be 50%. A separate 2:1 prescaler flipflop ensures this while the modulator clock is generated on-chip.

*Note: The EDSADC evaluates the rising edge of the phase synchronizer signal. In synchronized mode, it will operate while the phase synchronizer signal is toggling. In unsynchronized mode (**GLOBCFG**.USC = 1), the modulator clock is generated independent of the phase synchronizer signal.*
For proper operation, only change the configuration of the phase synchronizer while the converters are idle.

The bus interface is clocked with the system clock f_{SPB}.

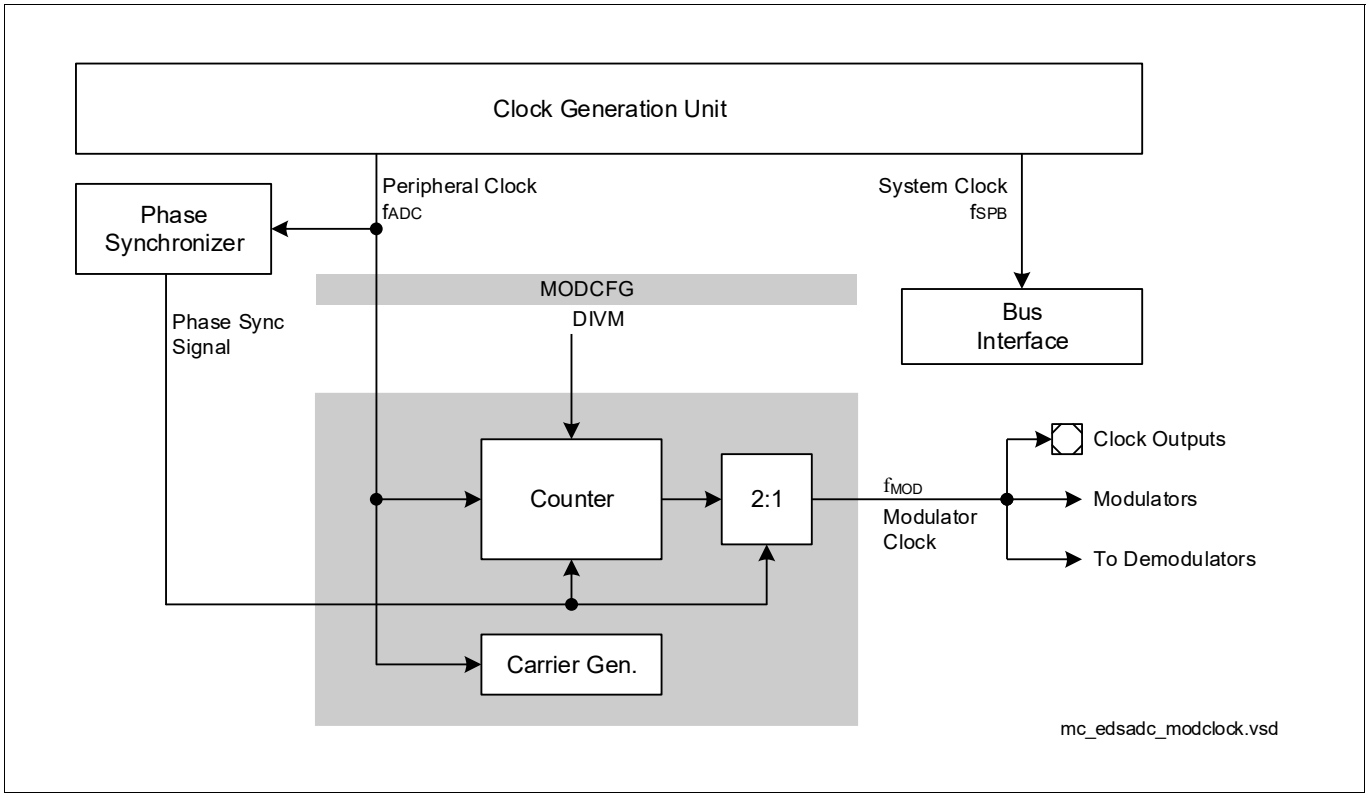


Figure 289 Modulator Clock Configuration

An external clock signal is synchronized to the module clock. A configurable edge detector selects the clock signal edges that generate the data strobes which read the next input data and trigger the demodulator (see **Figure 290**).

Note: To ensure proper synchronization, an external clock signal must have high/low phases of at least two periods of f_{ADC} to be safely synchronized (f_{IN} <= f_{ADC}/4).

When the clock signal is generated on-chip (see **Figure 289**) and is selected as an output signal at a connected pin, this clock signal can drive an external modulator.

Bitfield DSS in register **DICFGx (x=0-13)** selects the data source for each channel. For external modulators it also selects the clocking mode, i.e. the clock edges that put a new input data sample into the filter chain. In this case also the clock source can be selected.

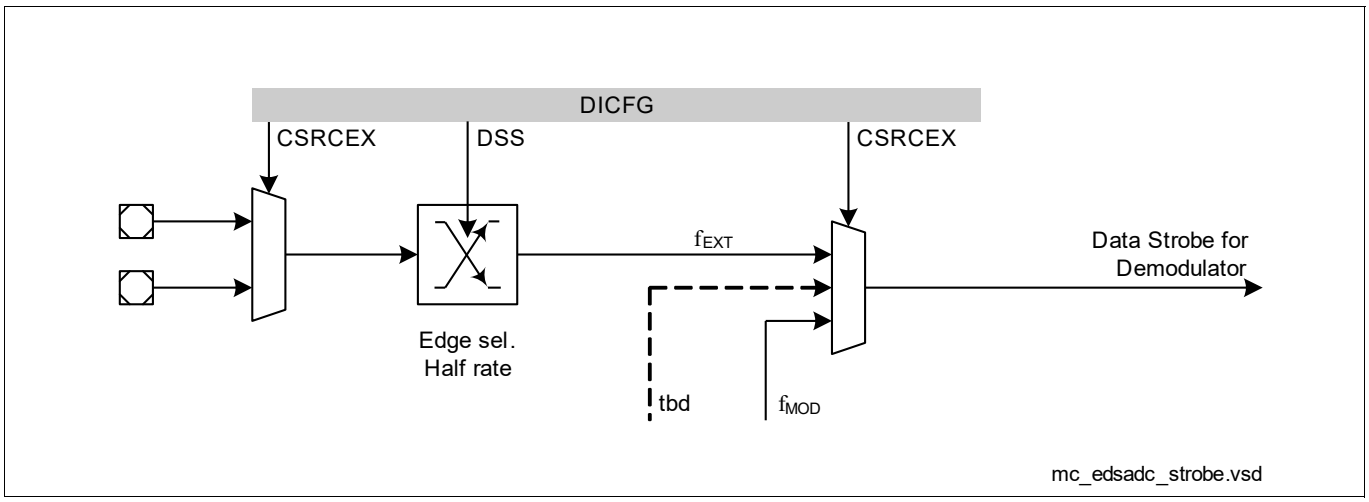


Figure 290 Demodulator Data Strobe Selection

33.4.2 Input Data Selection

The data stream of an external modulator can be input from a selectable pin. This signal can optionally be inverted.

The data stream can also be generated by the on-chip modulator. The selected input datastream is converted to the CIC filter's input format.

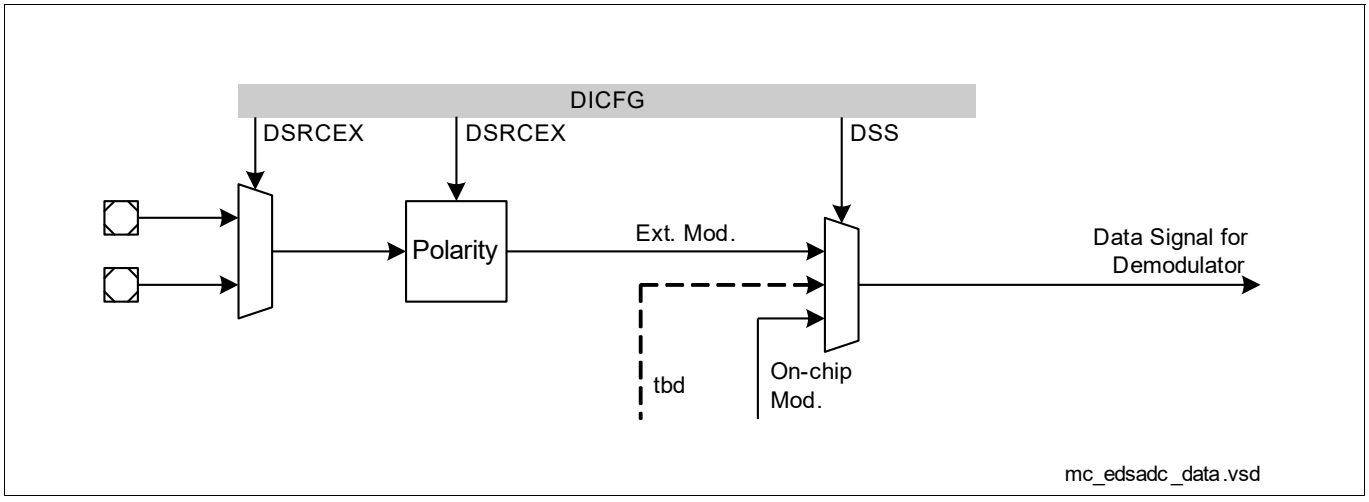


Figure 291 Demodulator Data Signal Selection

All options are configured by the demodulator input configuration register **DICFGx (x=0-13)**.

33.4.3 External Modulator

The input data stream can be obtained from an external modulator via a selectable pin. This modulator may, for example, be connected to a high driving voltage and be galvanically decoupled.

In this case, the modulator's data output is connected to the selected pin and is fed to the filter chain. The data input signal can optionally be inverted.

The sample clock signal can be generated on-chip or can be input from the external modulator. The data strobe that enters a new value into the filter chain can be generated upon configurable clock edges of the modulator clock to support different types of modulators.

33.4.4 On-Chip Modulator

The on-chip modulator is a 2nd order feed-forward modulator. It operates at a sample frequency of 40 MHz, 26.67 MHz, 20 MHz or 16 MHz. This sample frequency is derived from the actual module frequency (see **“Modulator Clock Generation” on Page 22**).

The analog input signal is fed to the sample stage via two input lines (differential input) or via one input line with the other input internally grounded (single-ended input).

Analog input multiplexers connect the input lines to different pins. The product-specific appendix details the available channels and their inputs. The input multiplexers are controlled by register **MODCFGx (x=0-13)**.

The gain factor of the input stage can be programmed to 1, 2, 4. The common mode voltage can be enabled or disabled.

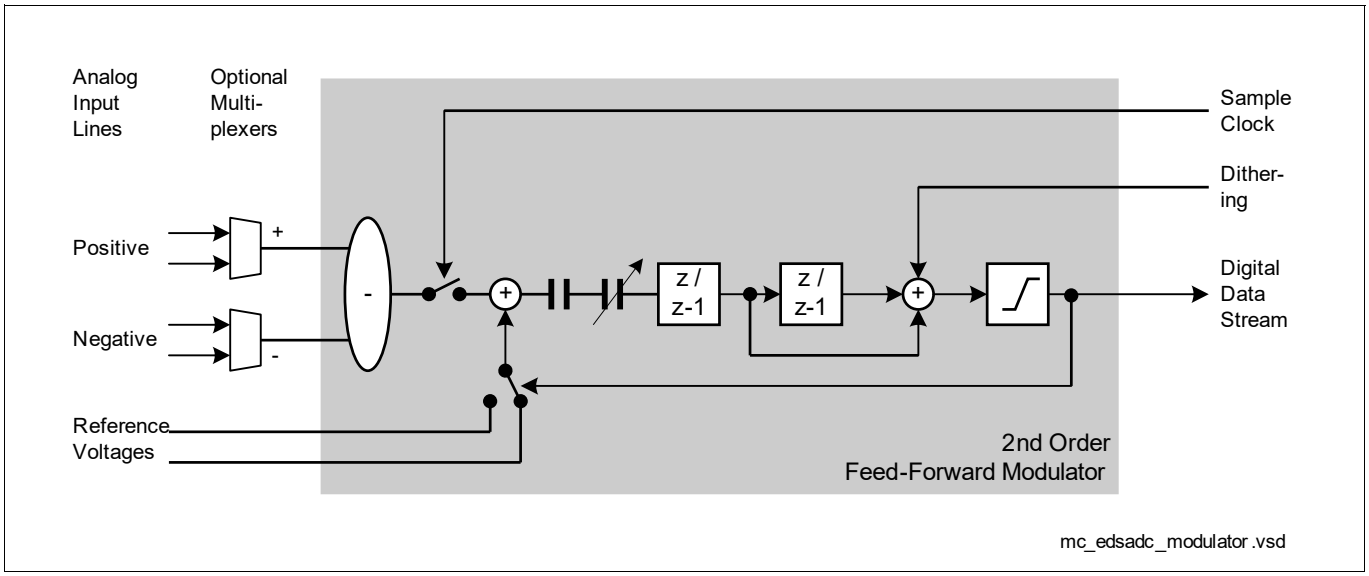


Figure 292 Structure of the On-Chip Modulator

Offset compensation and gain compensation improve the performance of the modulator. The calibration algorithm can be enabled during the initialization phase and also during operation, e.g. to compensate temperature drift.

Power Reduction via External Run Control

If an application does not require permanent activity of a channel, it can disable its modulator intermittently to save energy. This can be done under software control or automatically.

The operating mode is determined by bitfield **MODCFGx (x=0-13)**.APC:

- APC = 00_B: **Normal Operation**: The modulator is controlled by software.
- APC = 01_B: **Slow Standby mode**: Voltage regulator and on-chip modulator are switched off while the gate signal is inactive. They automatically return to normal operation when the gate signal becomes active. Requires the standard wakeup time (see below).
- APC = 10_B: **Fast Standby mode**: The on-chip modulator is switched off while the gate signal is inactive. It automatically returns to normal operation when the gate signal becomes active. Requires the standard wakeup time (see below).

Note: The standard wakeup time is also required after initially enabling the converter. If an external modulator is used, no wakeup time has to be considered. Slow Standby mode and Fast Standby mode are identical.

Wakeup Time from Analog Powerdown

When the modulator is activated, it needs a certain wakeup time (depending on the operating mode) to settle before proper data can be delivered. Make sure to include this settling time in the on-off cycles of the gate signal.

The standard wakeup time is maximum 20 µs.

Internal Dithering

Static or low-frequency input signals can generate so-called idle tones which could degrade the SNR. The internal dithering avoids this phenomenon.

The internal dithering can also reduce the so-called dead-zone.

Internal dithering is disabled by default, see bit DITHEN in register **MODCFGx (x=0-13)**.

Handling of Overload, Overdrive and Overflow Conditions

Operating an EDSADC channel outside its specified operating range requires additional careful handling and precautions. This ensures the generation of useful result values and prevents damaging the device.

An overload condition occurs if the input voltage on the respective pin exceeds the supply voltage. In this case the protection diodes try to limit the input voltage by becoming conducting. The resulting current must be limited to prevent damage to the protection structures. This is specified in the Datasheet.

An overdrive condition occurs if the resulting input voltage to the on-chip modulator becomes too high so the modulator can only generate a bitstream of all 1s. This may happen when an analog gain factor of 2 or 4 is selected. The susceptibility to an increased input voltage (e.g. overshoots) is reduced by the modulator's intrinsic gain factor of FM = 0.6945 (see also section **"CIC Filter" on Page 44**). The overdrive condition is determined by the input voltage, the analog gain factor selected by bitfield MODCFGx.GAINSEL and the reference voltage.

An overdrive condition, therefore, exists if ($V_{AIN} \times 2^{GAINSEL} \times 0.6945$) > V_{AREF}. In this case, the result value is <CALTARGET> × 1/0.6945.

The recovery from an overdrive condition would cause a non-constant group delay for the subsequent filter chain. This can be avoided by clearing the analog integrator stages of the modulator. Integrator clearing is done automatically if enabled by setting bit MODCFGx.IREN = 1[1)].

An overflow condition can, in turn, be generated by an overdrive condition. This can result in an overflow of the digital result that is obtained from the filter chain's output. To avoid this, the calibration target value must be selected below $2^{15} \times 0.6945$ = 22 757 and the data shifter after the CIC filter and the corresponding correction factor be set accordingly (see formula for CICSHIFT in section **"CIC Filter" on Page 44**).

Calibration targets above 22 757 do not cause overdrive if the input voltage V_{AIN} is limited by the system and does not exceed (2^{15} / <CALTARGET>) * (V_{AREF} / $2^{GAINSEL}$). The default value (after reset) of 25 000, for example, provides result values with a resolution of 0.2 mV/LSB.

The subsequent filter chain provides a normalized gain of 1.

Attention: ***Overload conditions while a gain factor of 4:1 is selected lead to an internal stress condition which can reduce the product's life time if the overall duration of this condition exceeds a total of 10 s. Make sure the application software recognizes this stress condition (easiest way is to detect the overdrive condition) and disables the respective modulator or reduces the gain factor within 10 ms. Limit checking can help to detect this situation automatically. This supervision allows to tolerate approx. 1000 occurrences of the internal stress condition over life time.***

1) It is recommended to enable integrator reset as a standard means to deal with on-chip modulator overdrive for applications where overdrive conditions can occur.

33.4.5 Input Path Control

The input for the EDSADC is fed through several stages before being evaluated and filtered. Registers MODCFGx and DICFGx select the available options for these signal stages.

The following features can be configured:

- Signal input pin selection
- Configuration of input stage and on-chip modulator
- Generation method of input data
- Modulator clock source and/or frequency
- Trigger input pin selection

Signal Input Pin Selection

Some modulator inputs provide analog input multiplexers which connect them to several input pins. These input multiplexers can be controlled directly by software or can be switched triggered by an external control signal. Software selects the desired position via bitfield INSEL, bitfield INMUX indicates the actual setting of the input multiplexer. Bitfield INMODE defines the condition for a trigger event, bitfield INMAC selects the way in which the multiplexer is controlled:

- Software control:
  Control bitfield INMUX follows bitfield INSEL, software directly selects the intended input pins.
- Preset mode:
  The (software-written) value in bitfield INSEL is copied to bitfield INMUX upon a trigger event. Software can preselect the next intended multiplexer setting which then becomes active when the next trigger event occurs.
- Single-step mode:
  Bitfield INMUX is decremented upon each trigger event. If INMUX = 00_B when the trigger occurs, it is loaded from bitfield INSEL instead. In this mode, a predefined sequence of input pins can be scanned automatically.

The trigger signal itself is selected by bitfield TRSEL in register **DICFGx (x=0-13)**.

Note: Not all channels provide input multiplexers. Also, the width of the input multiplexers may differ from channel to channel. The product-specific appendix details the available channels and their inputs inputs.

Modulator Configuration Register x

The modulator configuration register selects the operation mode of the on-chip modulator:

- Input line clamping
- Gain factor of input signal path
- Modulator operating mode
- Divider factor for modulator clock

Note: Changes to bitfields INCFGP, INCFGN, GAINSEL only become active after switching the analog multiplexer setting (if available) or after restarting the channel (MxRUN = 1).

MODCFGx (x=0-13)
Modulator Configuration Register x **(0100_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MMWC	0	APC		IREN	DITHEN	0			ACSD			0	DIVM		
w	r	rw		rw	rw	r			rw			r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INCWC	INMAC	INMODE		INMUX		INSEL		GAINSEL				INCFGN		INCFGP	
w	rw	rw		rh		rw		rw				rw		rw	

Field	Bits	Type	Description
INCFGP	1:0	rw	**Configuration of Positive Input Line** Defines the internal connection of the positive input. 00_B Input pin 01_B Reference voltage V_{AREF} 10_B V_{REFX} (see VREFXSEL setting in register VCM) 11_B Reference ground V_{AGND}
INCFGN	3:2	rw	**Configuration of Negative Input Line** Defines the internal connection of the negative input. 00_B Input pin 01_B Reference voltage V_{AREF} 10_B V_{REFX} (see VREFXSEL setting in register VCM) 11_B Reference ground V_{AGND}
GAINSEL	7:4	rw	**Gain Select of Analog Input Path** Not listed combinations are reserved. 0_H Gain factor 1 1_H Gain factor 2 2_H Gain factor 4
INSEL	9:8	rw	**Input Pin Selection** Defines the initial or permanent setting for the input multiplexer (bitfield INMUX) depending on the selected operating mode (bitfield INMODE).
INMUX	11:10	rh	**Input Multiplexer Setting** Indicates the current setting of the input multiplexer connecting the input pins to the buffer inputs. The product-specific appendix details the available channels and their inputs. 00_B Input pin position A 01_B Input pin position B 10_B Input pin position C 11_B Input pin position D

Field	Bits	Type	Description
INMODE	13:12	rw	**Input Multiplexer Control Mode** Defines the condition for a trigger event to control the input multiplexer. Bitfield INMAC selects the action upon a trigger event. 00_B Software control (INMUX follows INSEL) 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B Trigger event upon any edge
INMAC	14	rw	**Input Multiplexer Action Control** Defines the mechanism by which the input multiplexer is controlled. 0_B Preset mode (load INMUX upon a trigger) 1_B Single-step mode (decrement INMUX upon a trigger, wrap around to <INSEL>)
INCWC	15	w	**Write Control for Input Parameters** 0_B No write access to input parameters 1_B Bitfields INCFGP, INCFGN, GAINSEL, INSEL, INMODE, INMAC can be written
DIVM	18:16	rw	**Modulator Clock Period** Defines the period of the modulator clock (on-chip/external), derived from the peripheral clock: $t_{MOD} = t_{ADC} \times CP$ ($f_{MOD} = f_{ADC} / CP$), with CP = 4 + DIVM × 2. 000_B CP = 4 … 111_B CP = 18
ACSD	22:20	rw	**Analog Clock Synchronization Delay** Defines the delay in clocks after the sync signal. *Note: Valid only if the phase synchronizer is selected (USC = 0_B).* 000_B 0, no delay 001_B 1 clock cycle delay 010_B 2 clock cycles delay … 111_B 7 clock cycles delay
DITHEN	26	rw	**Dithering Function Enable** Controls the dithering function for each modulator separately. 0_B Disable dithering 1_B Dithering is enabled
IREN	27	rw	**Integrator Reset Enable** Controls the modulator overdrive handling 0_B No integrator reset 1_B Integrators are reset in case of an overdrive

Field	Bits	Type	Description
APC	29:28	rw	**Automatic Power Control** 00_B Off: Modulator active while its associated bit MxRUN is set 01_B Slow standby mode on-chip modulator and voltage regulator are deactivated, external modulator clock is disabled, while the gate signal (selected trigger) is inactive 10_B Fast standby mode:on-chip modulator is deactivated, external modulator clock is disabled, while the gate signal (selected trigger) is inactive 11_B Reserved
MMWC	31	w	**Write Control for Modulator Mode Settings** 0_B No write access to mode settings 1_B Bitfields APC, IREN, DITHEN, ACSD, DIVM can be written
0	19, 25:23, 30	r	**Reserved, write 0, read as 0**

Table 278 Access Mode Restrictions of MODCFGx (x=0-13) sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **INCWC**	rw	GAINSEL, INCFGN, INCFGP, INMAC, INMODE, INSEL	Set INCWC during write access
write 1 to **MMWC**	rw	ACSD, APC, DITHEN, DIVM, IREN	Set MMWC during write access
(default)	r	ACSD, APC, DITHEN, DIVM, GAINSEL, INCFGN, INCFGP, INMAC, INMODE, INSEL, IREN	

Demodulator Input Config. Register x

The demodulator input configuration register selects input signal sources for each channel:

- Source of the data stream
- Input for an external data stream
- Input for an external clock
- Trigger signal source and modes
- Read mode for the result register

Note:

DICFGx (x=0-13)

Demodulator Input Config. Register x **(0108_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSWC	**0**	**RDM**	**TSM**	**DRM**		**0**		**TSTRMODE**		**ITRMODE**		**TRSEL**			
w	r	rw	rw	rw		r		rw		rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSWC	**0**				**CSRCEX**			**0**	**DSRCEX**			**0**	**DSS**		
w	r				rw			r	rw			r	rw		

Field	Bits	Type	Description
DSS	2:0	rw	**Data Stream Select** 000_B On-chip modulator 001_B Reserved 010_B Reserved 011_B External modulator, use rising and falling clock edge (double data) 100_B External modulator, use each falling clock edge (direct clock) 101_B External modulator, use each rising clock edge (direct clock) 110_B External modulator, use every 2nd falling clock edge (dbl. clock) 111_B External modulator, use every 2nd rising clock edge (dbl. clock)
DSRCEX	6:4	rw	**Data Source for External Modulator** 000_B External, from input A, direct 001_B External, from input A, inverted 010_B External, from input B, direct 011_B External, from input B, inverted 100_B External, from input C, direct 101_B External, from input C, inverted 110_B External, from input D, direct 111_B External, from input D, inverted
CSRCEX	10:8	rw	**Clock Source for External Modulator** 000_B Internal clock 001_B Reserved 010_B Reserved 011_B External, from input A 100_B External, from input B 101_B External, from input C 110_B Reserved 111_B Reserved
DSWC	15	w	**Write Control for Data Stream Selection** 0_B No write access to data parameters 1_B Bitfields CSRCEX, DSRCEX, DSS can be written

Field	Bits	Type	Description
TRSEL	19:16	rw	**Trigger Select** Selects an input for the trigger signal used for the following features (see also **Figure 288**): [1] integrator control, timestamp, multiplexer control, modulator control (APC), service request gating. The product-specific appendix details the connected trigger input signals.
ITRMODE	21:20	rw	**Integrator Trigger Mode** To ensure proper operation, ensure that bitfield ITRMODE is 00_B before selecting any other trigger mode.[2] Bit INTEN is set when ITRMODE = 11_B or when the selected trigger signal transition occurs. Bit INTEN is cleared when ITRMODE = 00_B, after REPVAL+1 integration cycles (IWS = 0) or when the inverse trigger signal transition occurs (IWS = 1). 00_B No integration trigger, integrator bypassed 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B No trigger, integrator active all the time
TSTRMODE	23:22	rw	**Timestamp Trigger Mode** The timestamp trigger mode controls capturing the timestamp information to register **TSTMPx**. 00_B No timestamp trigger 01_B Trigger event upon a falling edge 10_B Trigger event upon a rising edge 11_B Trigger event upon each edge
DRM	27:26	rw	**Data Read Mode** Selects the data that is returned when register RESMx is read (see **Table 290**). 00_B Single: Issue one 16-bit value per read access (sign on high bits) 01_B Single: Issue one 16-bit value per read access (timestamp or zero on high bits) 10_B Double: Issue two 16-bit values per read access 11_B Reserved
TSM	28	rw	**Time-Stamp Mode** See **Table 290**. 0_B No timestamp, only issue result values 1_B Insert timestamp upon the trigger (when the gate opens)
RDM	29	rw	**Result Display Mode** 0_B Signed mode result values range from -2^{15} to $+2^{15}-1$ 1_B Unsigned mode result values range from 0 to $+2^{16}-1$ (shifted by 2^{15})
MSWC	31	w	**Write Control for Mode Settings** 0_B No write access to mode settings 1_B Bitfields RDM, TSM, DRM, TSTRMODE, ITRMODE, TRSEL can be written

Field	Bits	Type	Description
0	3, 7, 14:11, 25:24, 30	r	**Reserved, write 0, read as 0**

1) To avoid unintended triggers, select the trigger source first before enabling the corresponding function.
2) The integration trigger mode controls bit INTEN in register **IWCTRx** and hence the operation of the integrator:

Table 279 Access Mode Restrictions of DICFGx (x=0-13) sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **DSWC**	rw	CSRCEX, DSRCEX, DSS	Set DSWC during write access
write 1 to **MSWC**	rw	DRM, ITRMODE, RDM, TRSEL, TSM, TSTRMODE	Set MSWC during write access
(default)	r	CSRCEX, DRM, DSRCEX, DSS, ITRMODE, RDM, TRSEL, TSM, TSTRMODE	

33.4.6 Common Mode Voltage

Some applications require the assertion of a common mode voltage e.g. to support passive differential sensors. The common mode voltage V_{CM} is derived from the analog reference voltage V_{AREF}: $V_{CM} = V_{AREF} / X$ and is enabled via bit VXON and configured via bitfield VREFXSEL in register **VCMx (x=0-13)**.

Bits INyVCz in register **VCMx (x=0-13)** select the connection to V_{CM} for each of the possible input pins. Two nibbles are assigned to each channel (positive and negative inputs) and contain the control bits for each possible input of the respective channel.

While the input pins are not connected to the input lines of the channel (e.g. while the analog input multiplexer selects different pins or while the modulator is off), each individual analog input pin that is intended to be used for the EDSADC can remain connected to the common mode voltage V_{CM}. This ensures that the sensor input is biased and does not float away while the respective input line is disconnected or the modulator is disabled.

Note: The available channels provide different numbers of input pins. The product-specific appendix details the available channels and their inputs inputs. Only those INyVCz bits in VCMx are valid that correspond to an existing input pin.

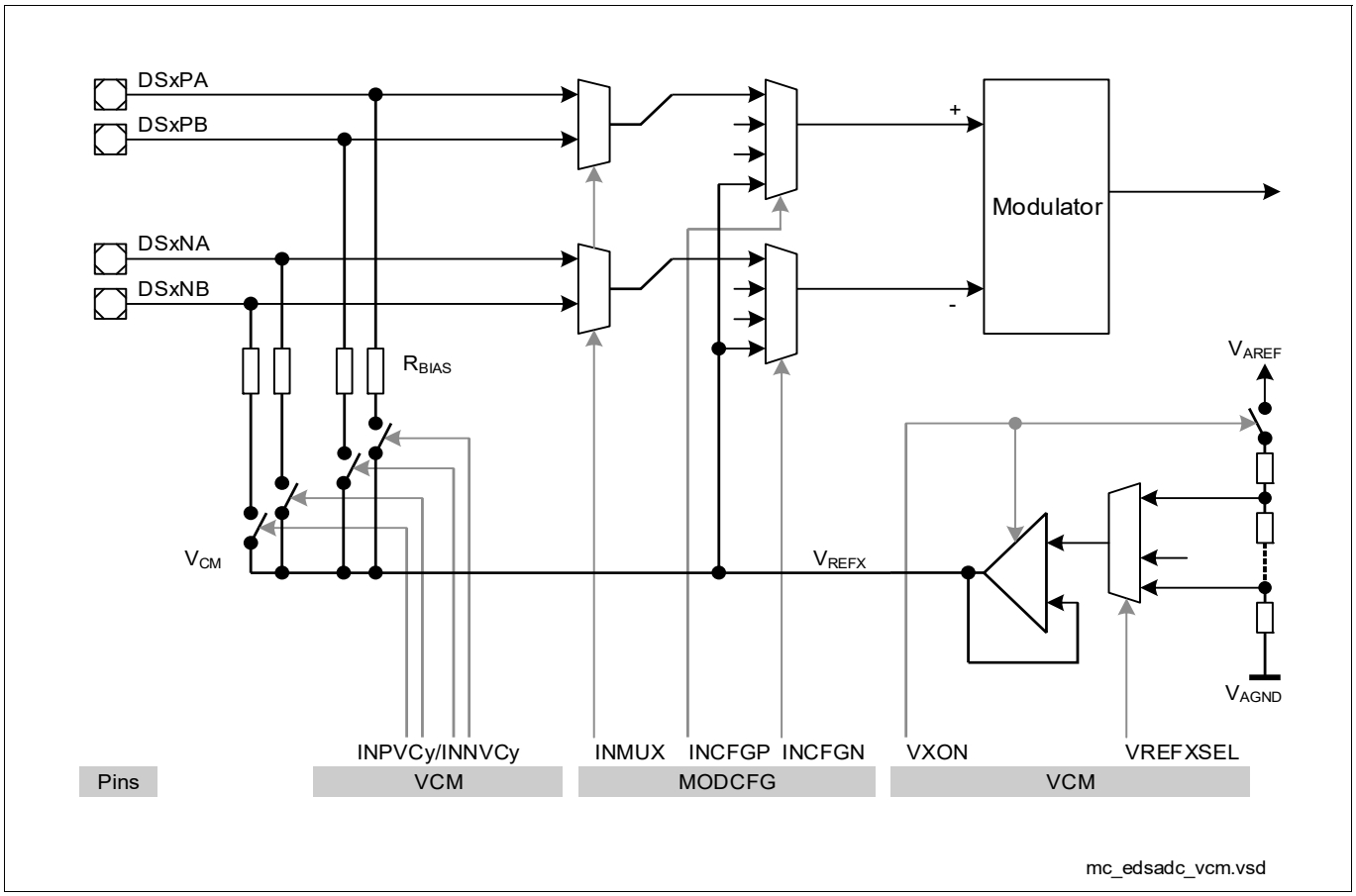


Figure 293 Control of VCM Switches

Common Mode Voltage Register x

The common mode voltage registers enable the voltage generators and select which input pins are connected to the common mode hold voltage.

Only those bits are valid that correspond to an existing input pin (see product-specific appendix).

VCMx (x=0-13)

Common Mode Voltage Register x **($01B0_H$+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								INNVC3	INNVC2	INNVC1	INNVC0	INPVC3	INPVC2	INPVC1	INPVC0
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													VXON	VREFXSEL	
r													rw	rw	

Field	Bits	Type	Description
VREFXSEL	1:0	rw	**Fractional Reference Voltage Selection** 00_B $V_{REFX} = V_{AREF} / 2$ 01_B $V_{REFX} = V_{AREF} / 4$ 10_B $V_{REFX} = V_{AREF} / 8$ 11_B Reserved
VXON	2	rw	**Fractional Reference Voltage Enable** 0_B V_{REFX} is not connected 1_B V_{REFX} is connected, value according to VREFXSEL
INPVCy (y=0-3)	y+16	rw	**Voltage Control of Positive Inputs y of CHx** Defines the connection of the respective positive input y to the common mode voltage. y indicates the input of the analog multiplexers (if available). 0_B No connection to common mode voltage 1_B This pin is connected to the common mode voltage
INNVCy (y=0-3)	y+20	rw	**Voltage Control of Negative Inputs y of CHx** Defines the connection of the respective negative input y to the common mode voltage. y indicates the input of the analog multiplexers (if available). 0_B No connection to common mode voltage 1_B This pin is connected to the common mode voltage
0	15:3, 31:24	r	**Reserved, write 0, read as 0**

33.4.7 Calibration Support

The performance of the EDSADC can be improved by applying some calibration techniques.

The calibration algorithm for gain and offset is executed automatically. Since it takes a certain time during which no conversions can be executed, two modes can be used:

- Software can determine when the algorithm is executed by setting bit CALIB in register **FCFGMx (x=0-13)**.
- In the case of gated operation the calibration algorithm can also be started automatically when the gate closes (set bit AUTOCAL in register **FCFGMx (x=0-13)**).

The calibration algorithm supports differential and single-ended mode with gain settings of 1 and 2. The respective calibration mode is derived from registers **MODCFGx (x=0-13)** and **VCMx (x=0-13)** as shown in the table below:

Table 280 Supported Calibration Modes

Selected Mode	Corresponding Bitfields in MODCFG
Differential, gain 1 / gain 2	INCFGP = INCFGN = 00_B, GAINSEL = 0000_B or 0001_B
Single-ended, gain 1	INCFGx = 00_B, INCFGy = 11_B, GAINSEL = 0000_B
Single-ended, gain 2	INCFGx = 00_B, INCFGy = 10_B, GAINSEL = 0001_B, VCMx.VREFXSEL = 00_B

Note: An initial calibration must be triggered once after a reset.
Calibration for a gain factor of 4 is executed with gain setting 2.

Repeated Calibration During Operation

To compensate temperature effects it is recommended to repeat the calibration sequence when the device temperature has changed by approximately 20°C. This can be done by simply setting bit CALIB in register **FCFGMx (x=0-13)** or by enabling automatic calibration (AUTOCAL = 1). After calibration, the channel will resume its normal operation.

Calibration Timing

The calibration uses a sequential search algorithm to determine the correct calibration factor. The duration of the calibration algorithm, therefore, depends on the deviation to be calibrated (algorithm), and also on the configuration of the module, i.e. on the modulator frequency ($f_{MOD} = f_{ADC} / (4 + 2\times DIVM)$) and on the decimation rate used for calibration ($DC = 8 \times 2^{CICDEC}$).

The calibration time, therefore, is determined by $t_{ADC} = 1 / f_{ADC}$ and $t_{MOD} = 1 / f_{MOD}$.

The following formula describes the required maximum calibration time: $t_{CAL} = 16 \times DC \times t_{MOD} + 4\,200 \times t_{ADC}$.
If gain factor 2 is selected the calibration algorithm is extended by: $t_{2nd} = 12 \times DC \times t_{MOD} + 8\,300 \times t_{ADC}$,
so the total maximum calibration time will be: $t_{CAL2} = 28 \times DC \times t_{MOD} + 12\,500 \times t_{ADC}$.

Table 281 Calibration Times [µs] for f_{ADC} = 160 MHz, Gain1 (Gain2)

DIVM [f_{MOD}] --->	0 [40]	1 [26.7]	2 [20]	3 [16]
CICDEC = 4, DC = 128	78 (168)	104 (213)	129 (258)	155 (303)
CICDEC = 5, DC = 256	129 (258)	180 (347)	232 (437)	283 (527)
CICDEC = 6, DC = 512	232 (437)	334 (616)	436 (795)	539 (975)

Note: Higher decimation rates require more calibration time but offer a higher calibration precision.
*The end of calibration is indicated in bitfield CAL in register **FCNTCx (x=0-13)**.*

Range Adaptation

The application can select the digital full-scale result value that corresponds to an input of $V_{AIN} = V_{AREF}$.

The intended full-scale value (CALTARGET) is reached by configuring the decimation rate of the CIC filters, selecting the corresponding position of the data shifter and configuring the resulting multiplication factor for the gain adjust unit, written to register **GAINCORRx (x=0-13)** (see links below).

The signal to noise ratio defines the effective number of result bits: ENOB = (SNR - 1.76) / 6.02.

Since (80 dB - 1.76) / 6.02 = 13 bit, selecting a minimum 14-bit target value is recommended.

When choosing the full-scale value, note, however, that using the full 15-bit range may lead to overflows in case of overshoots. See **"Handling of Overload, Overdrive and Overflow Conditions" on Page 26**.

Summary of Calibration Parameters

During calibration the settings of CIC filter and gain correction are replaced by specific values which are valid for the calibration algorithm. The application needs to compute and configure these values in addition to the values used during operation.

The calibration algorithm will trim the channel according to the configured calibration target value (CALTARGET).

Note: The formula to determine the GAINFACTOR is given in section ***"Data Shifter and Decimation Factor" on Page 45***.

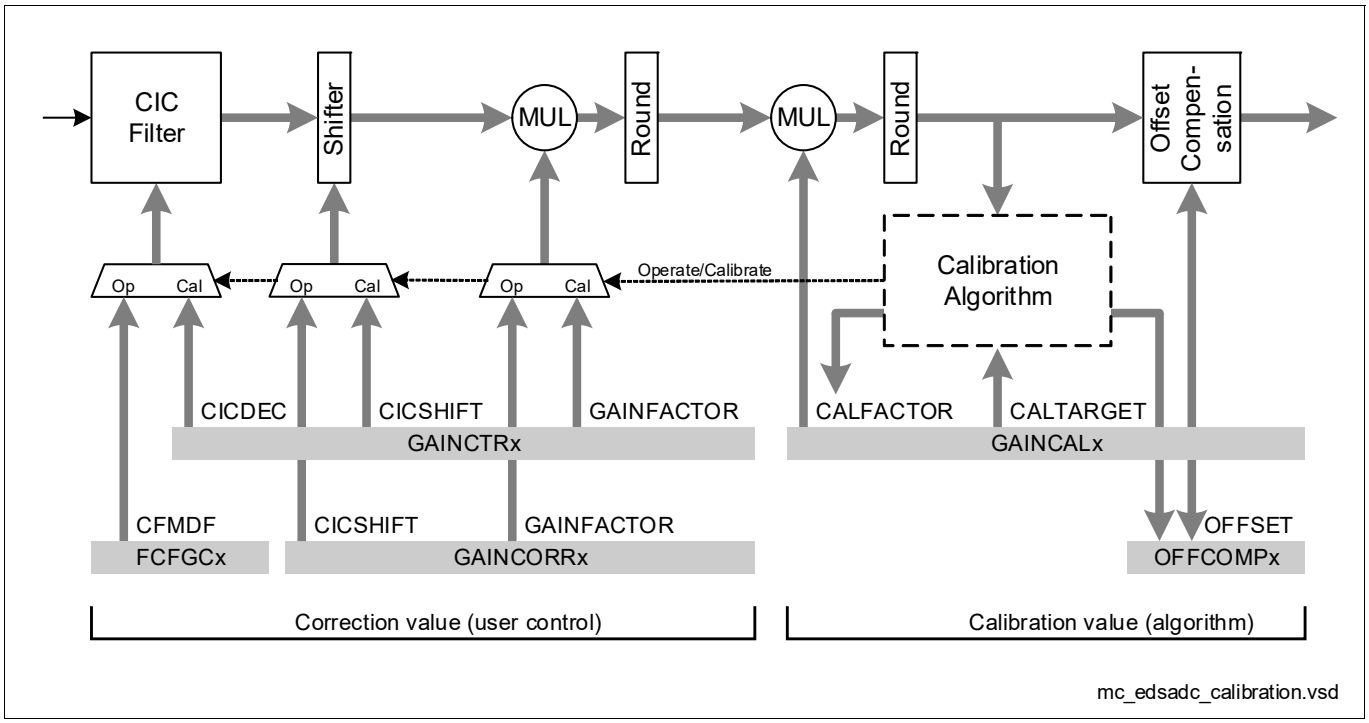


Figure 294 Calibration Overview

Gain Calibration Register x

GAINCALx (x=0-13)

Gain Calibration Register x **($013C_H$+x*100_H)** **Application Reset Value: 61A8 1000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								CALTARGET							
r								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			CALFACTOR												
r			rwh												

Field	Bits	Type	Description
CALFACTOR	12:0	rwh	**Multiplication Factor for Gain Calibration** The resulting factor is (<CALFACTOR> / 4 096) *Note: The initial value of 4 096 (1000_H) corresponds to a factor of 1.000.*
CALTARGET	30:16	rw	**Target Value for Calibrated Fullscale** Defines the target value for the calibration algorithm. *Note: The initial value of 25 000 ($61A8_H$) corresponds to 0.2 mV per LSB.*
0	15:13, 31	r	**Reserved, write 0, read as 0**

Gain Control Register x

GAINCTRx (x=0-13)

Gain Control Register x **(0140_H+x*100_H)** **Application Reset Value: 0000 1000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					CICDEC			0			CICSHIFT				
r					rw			r			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			GAINFACTOR												
r			rw												

Field	Bits	Type	Description
GAINFACTOR	12:0	rw	**Multiplication Factor for Gain Correction During Calibration** The resulting factor is (<GAINFACTOR> / 4 096)

Field	Bits	Type	Description
CICSHIFT	20:16	rw	**Position of the CIC Filter Output Shifter During Calibration** Selects the valid outputs bits from the CIC filter, depending on the chosen decimation factor (see data shifter formula), $1D_H$ … $1F_H$ are reserved. 00_H Use bits 0 ... 16 … $1C_H$ Use bits 28 ... 44
CICDEC	26:24	rw	**Decimation Rate of the CIC Filter During Calibration** Factor = 2 ^ (CICDEC + 3) 000_B 8 … 110_B 512 111_B Reserved
0	15:13, 23:21, 31:27	r	**Reserved, write 0, read as 0**

33.4.8 Correction for External Circuitry

The built-in calibration support is optimized for low-impedance inputs. Operation using higher impedances (>100 Ohm) incurs gain errors due to the external circuitry (series resistance and buffer capacitor.

- Single-ended operation (see **Figure 295**):
  One input line usually internally connected to ground.
  An additional compensation factor can be calculated to compensate the error caused by the average input current I_{RMS} flowing through the resistance of the external path R_{EXT}:
  $F_{GR} = 1 + (I_{RMS} \times R_{EXT}) / 5\ V$
- Differential operation (see **Figure 296**):
  Both input lines connected to signal source.
  An additional compensation factor can be calculated to compensate the error caused by the average input current I_{RMS} flowing through the resistances of the external path R_{EXTP} and R_{EXTN}:
  $F_{GR} = 1 + (I_{RMS} \times (R_{EXTP} + R_{EXTN})) / 5\ V$

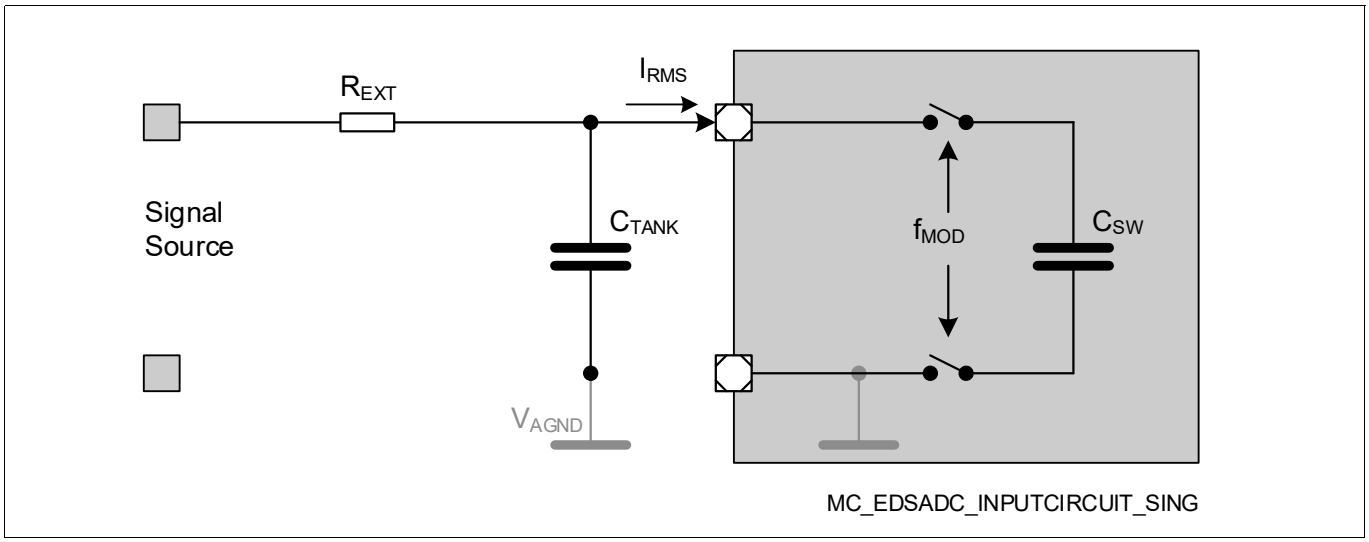


Figure 295 Single-Ended Input Circuitry of a Channel

*Note: This description example assumes the single-ended sensor signal to be connected to the positive input and the negative input to be grounded internally (indicated in grey in **Figure 295**).*

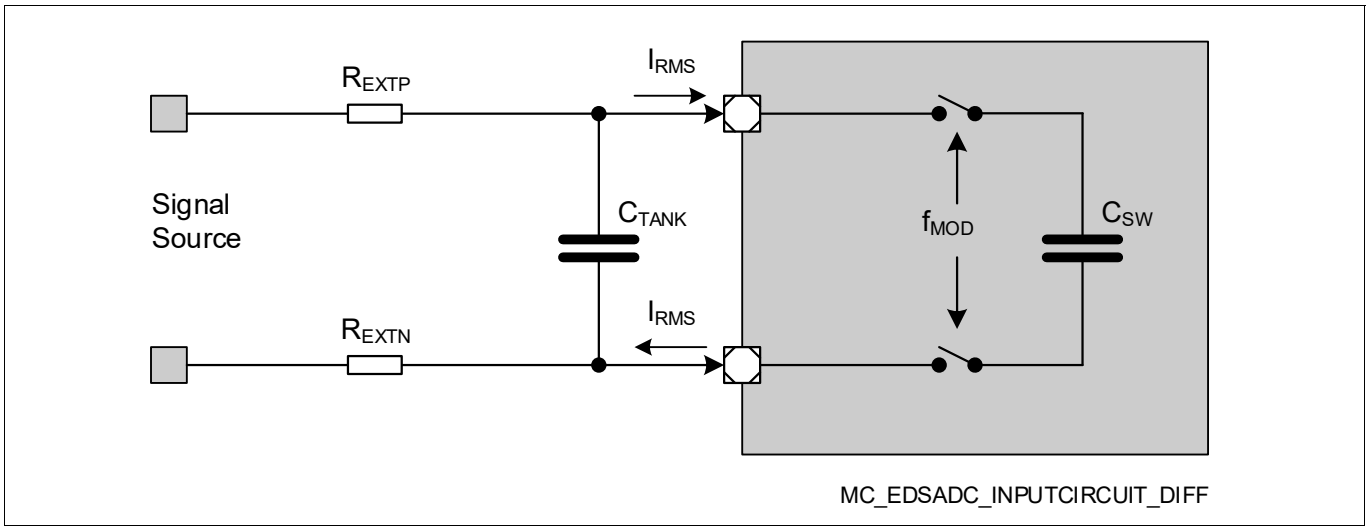


Figure 296 Differential Input Circuitry of a Channel

The buffer capacitor C_{TANK} not only is a part of the anti-alias filter but also minimizes the gain error of the complete input path. The maximum value for C_{TANK} is determined by the maximum signal frequency that shall be converted by the respective channel.

Also the buffer capacitor itself causes a small additional gain error. This can be compensated with the following formula:

$F_{GC} = 1 + I_{RMS} / (f_{MOD} \times C_{TANK} \times 5\ V)$

These factors can be combined with the range adaptation factor and CIC correction factor and be written into register **GAINCORRx (x=0-13)**.

Stored Calibration Values

The average input current is measured during production and the value is stored in the on-chip Flash (value IRMS, see below). The application can, therefore, calculate a device-specific correction factor.

The stored measurement value is referenced to a voltage of 5.0 V, a modulator frequency of 26.67 MHz and a gain factor of 1:1. So the actual input current is $I_{RMS} = (IRMSx / 100) \times (f_{MOD} / 26.67) \times 2^{GAINSEL}$.

The device-specific calibration values are stored in the User Configuration Block (UCB_USER) within the on-chip Flash memory. The area assigned to the EDSADC begins at offset 0080_H and provides records of two 32-bit words per channel. The channel-specific records, therefore, can be read from offset [0080_H + 8×CHNr.] within block UCB_USER. The table below shows the structure of a record.

Table 282 Structure of EDSADC Calibration Records (Generated at f_{MOD} = 26.67 MHz)

	Halfword 3	Halfword 2	Halfword 1	Halfword 0
Location CH0	0086_H	0084_H	0082_H	0080_H
Location CH1	$008E_H$	$008C_H$	$008A_H$	0088_H
:	:	:	:	:
Stored value	IRMS	Reserved	Reserved	Reserved

The parameters are stored in the following format:

Current = IRMSx × 0.01 µA, i.e. $03E8_H$ = 10 µA.

Example:

For R_{EXT} = 10 kOhm, IRMS = $042E_H$ (10.7 µA), C_{TANK} = 330 pF, f_{MOD} = 40 MHz, GAINSEL = 0:

The real current at 40 MHz, therefore, is I_{RMS} = 10.7 µA × 40 / 26.67 × 2^0 = 16.05 µA:

F_{GR} = 1 + (16.05 µA × 10 kOhm) / 5 V = 1.0321

F_{GC} = 1 + 16.05 µA / (40 MHz × 330 pF × 5 V) = 1.00024

The total factor will be $F_G = F_{GR} \times F_{GC}$ = 1.0321 × 1.00024 = 1.03235

Assuming a range adaptation factor of 1.2 (25 000 to 30 000), the overall factor would be 1.2 × 1.03235 = 1.2388, so the value for bitfield GAINFACTOR is 5 074 = $13D2_H$.

33.5 Filter Chain

The result data words are generated by feeding the input data stream through a chain of filter elements and decimating it by a selectable ratio.

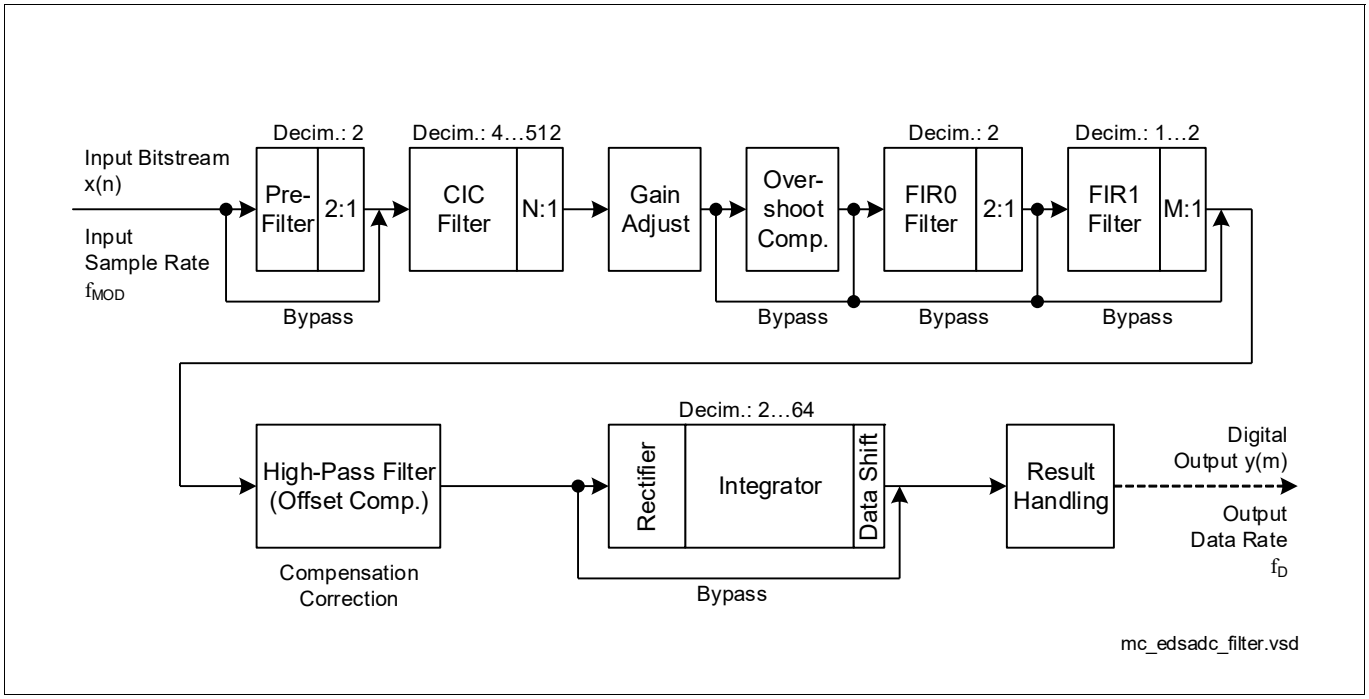


Figure 297 Structure of the Filter Chain

Most elements of the filter can be bypassed, i.e. the filter chain is configurable and its behavior can be adapted to the requirements of the actual application. This comprises the frequency attenuation as well as the total decimation rate.

The filter chain consists of the following elements:

- **CIC Filter**
- **Gain Correction**
- **Overshoot Compensation**
- **FIR Filters**
- **Offset Compensation**
- **Integrator Stage**

*Note: Reconfiguring filter parameters while the channel is active incurs some implications. Refer to **Section 33.3.1**.*

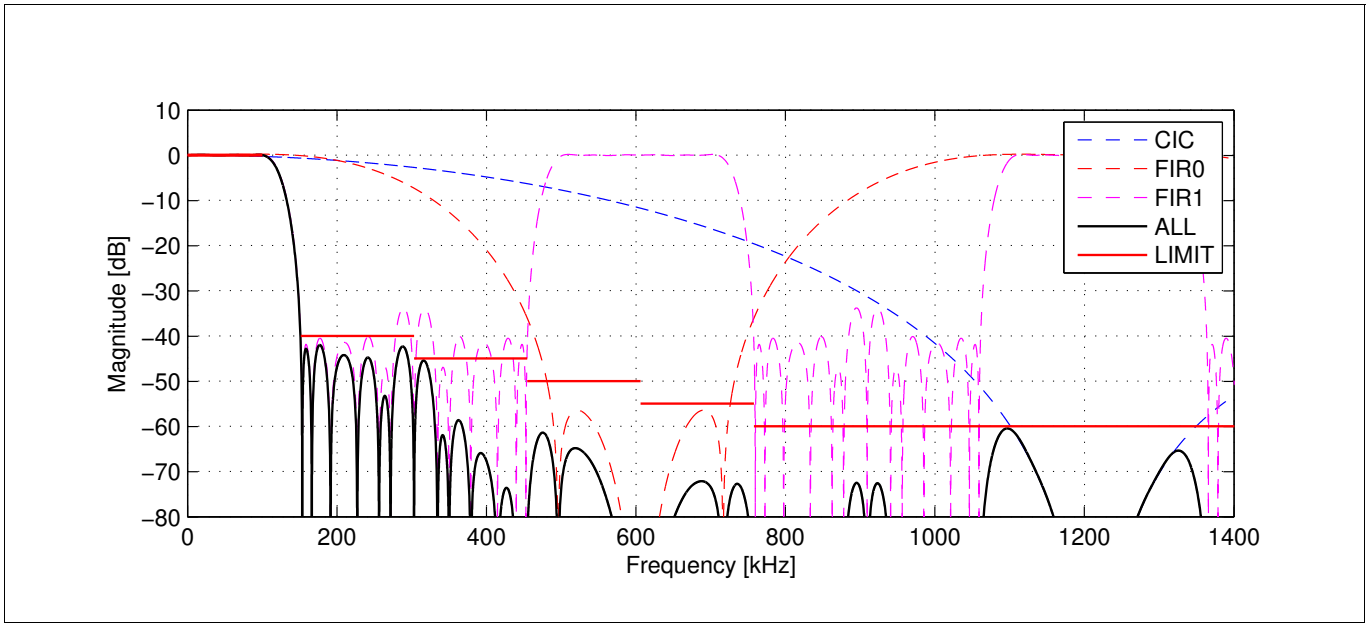


Figure 298 Frequency Response of the Complete Filter Chain (Example for 100 kHz Passband Setup)

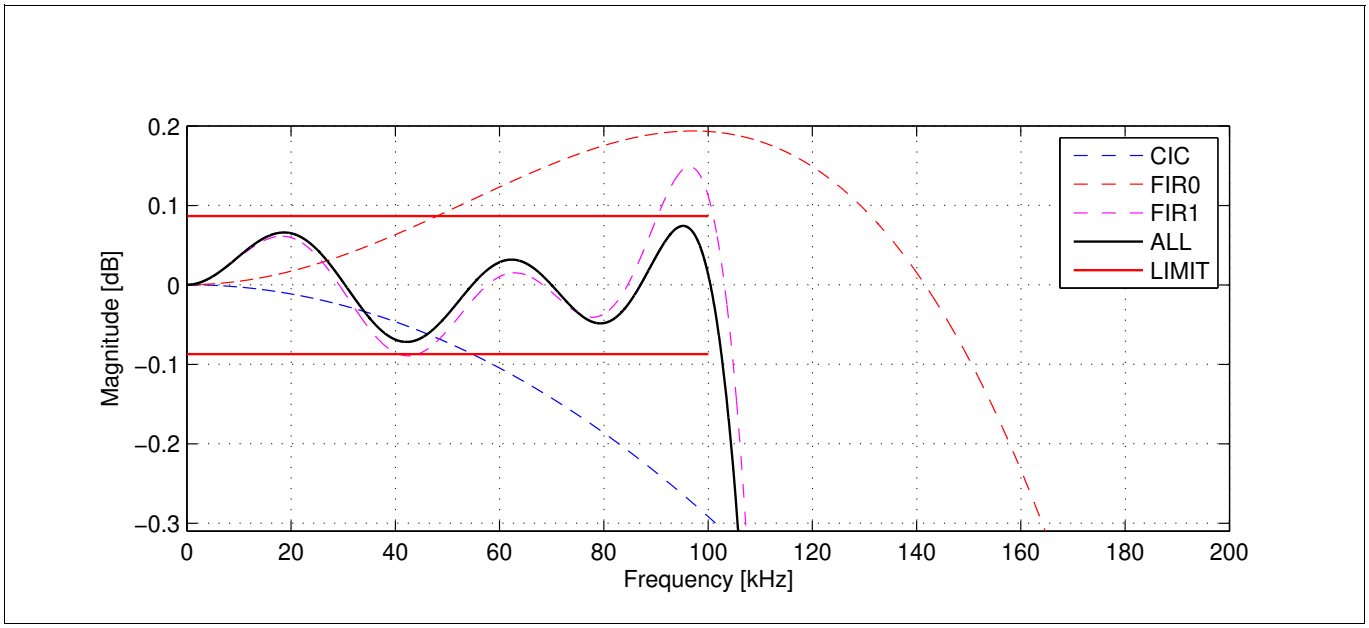


Figure 299 Passband Ripple (Example for 100 kHz Passband Setup)

33.5.1 CIC Filter

The Cascaded Integrator Comb filter (CIC filter, a.k.a. SINC filter) is a simple but very efficient low-pass filter. Three comb filter stages are cascaded to improve the frequency characteristics (CIC3). The decimation rate is programmable within a wide range from 4 to 512.

A prefilter reduces the data rate for modulator frequencies above 20 MHz. The prefilter is a CIC3 filter with a fixed decimation rate of 2.

To synchronize filters of different channels with fine granularity, the decimation counter starts from an arbitrary start value. This start value can be different from the decimation factor and is loaded only once when the counter is started.

The decimation counter is also restarted (i.e. loaded with the start value) when the selected integration trigger event occurs (see **Section 33.5.6**).

Table 283 CIC Filter Properties (N = Decimation Rate)

Stages	Data Width	Maximum Delay	Peak Output Value
CIC3	25 bits signed	3N	[-N^3, +N^3]

The CIC3 filter consists of 3 cascaded CIC filter stages.

Frequency domain, frequency response function:

$$H(z) = \left(\frac{1 - z^{-N}}{1 - z^{-1}} \right)^{k}$$

$$\left| H(j\omega) \right| = \left(\frac{\sin(\omega N / 2)}{\sin(\omega / 2)} \right)^{3}$$

Time domain:

The figure below illustrates the response of the CIC3 filter:

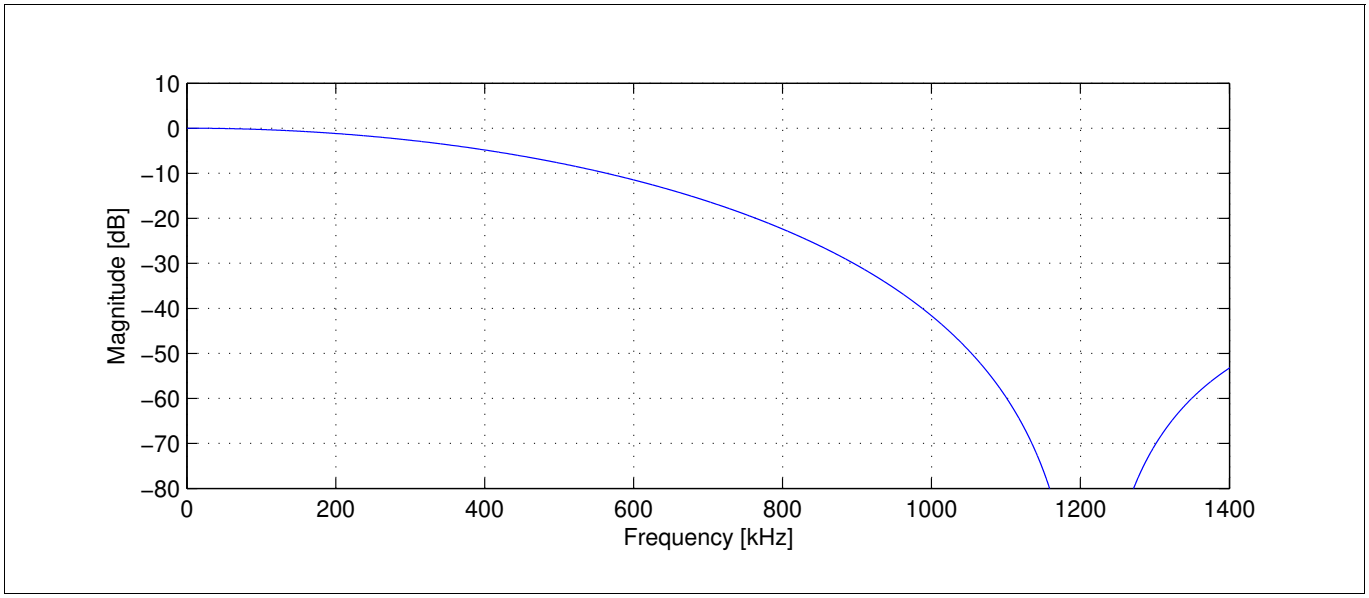


Figure 300 CIC3 Filter Frequency Response (Example for 1.212 MHz Output Data Rate)

Data Shifter and Decimation Factor

At the CIC filter output the valid data bits are selected for the subsequent blocks. The selected portion of the filter output is determined by the selected decimation factor and the employed modulator. These parameters define the maximum possible data value and, hence, the proper position for the extracted output value. During operation, the data shifter is controlled by bitfield GAINCORRx.CICSHIFT, during calibration by bitfield GAINCTRx.CICSHIFT.

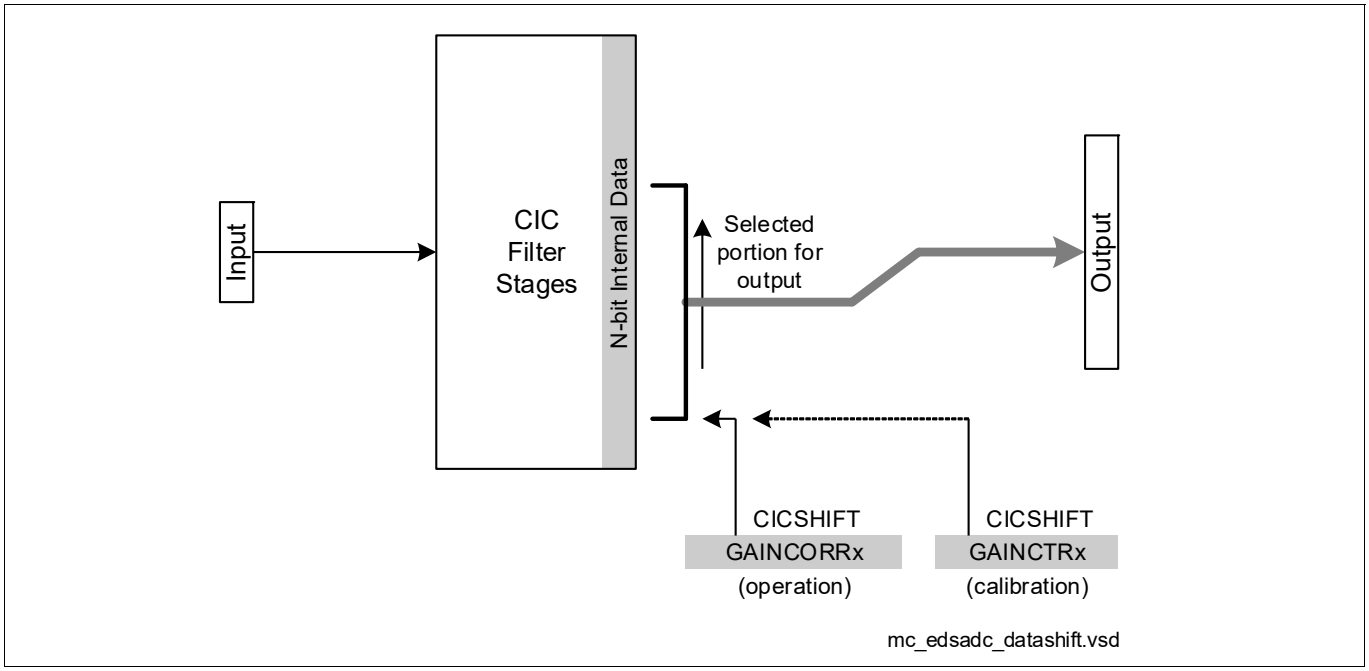


Figure 301 Data Shift Unit

The value for bitfield CICSHIFT is determined with this formula:

<CICSHIFT> = roundup(14 - ld(2 × AFS / (N^3 × 4 × FM)))

The following parameters are used:

- N = selected decimation factor
- AFS = calibrated full-scale value (25 000 after reset), refers to the analog full-scale ($V_{IN} = V_{AREF}$)
- FM = modulator gain factor
  - on-chip modulator: FM = 0.6945
  - external modulator: FM depends on used type

The gap that comes from the rounding in above formula can be closed by computing a corresponding gain correction factor:

<GAINFACTOR> = truncate(((2 × AFS / (N^3 × 4 × FM)) × 2^(<CICSHIFT>-14)) × 4096)

Example:

For N = 50, on-chip modulator:

N^3 × 4 × 0.6945 = 347 250

<CICSHIFT> = roundup(14 - ld(2 × 25 000 / 347 250)) = roundup(14 - (-2.796)) = 17 (11_H)

The correction factor will be:

<GAINFACTOR> = (2 × 25 000 / 347 250) × 2^3 = 1.1519

The bitfield value will then be truncate(1.1519 × 4 096) = truncate(4 718.1824) = 4 718 = $126E_H$.

33.5.2 Gain Correction

Gain calibration is done by multiplying the raw results with a correction value and a calibration value.

The application correction value is determined by the application based on the selected decimation factor etc., and is used to compensate the reduced digital gain that results from decimation factors in the CIC filter that are not 2^N (see **"Data Shifter and Decimation Factor" on Page 45**).

The calibration value is determined during the calibration sequence by the calibration algorithm. The automatic calibration algorithm normalizes the overall gain factor of the filter chain to 1.000, independent of their configuration. The full-scale value of the result is adapted to <CALTARGET> (25 000 after reset), representing the reference voltage.

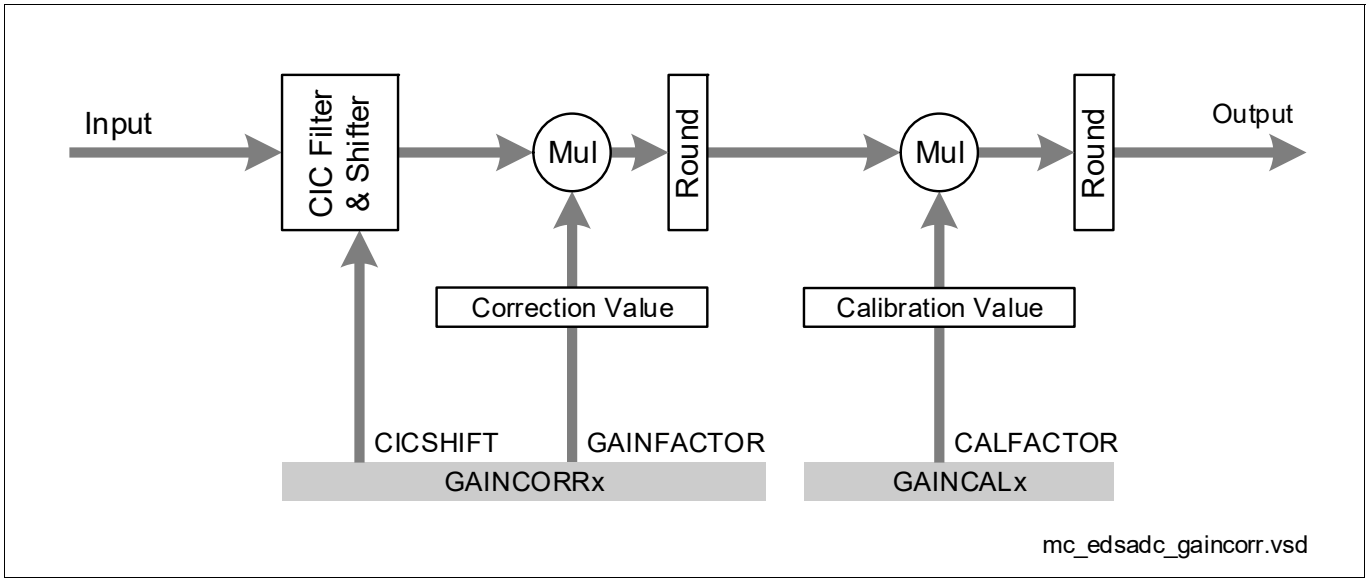


Figure 302 Gain Correction

Note: The calibration algorithm ensures that the configured target value (CALTARGET) is reached at the output of the gain correction unit within the specified precision when $V_{IN} = V_{AREF}$.

The correction value in register GAINCORR is calculated from application parameters either during compile time for static configurations, or during run time for dynamic configurations.

The calibration value in register GAINCAL is generated automatically by the calibration algorithm. This value is generated during the initial startup calibration and is adapted by subsequent calibration cycles that are triggered either by software or automatically.

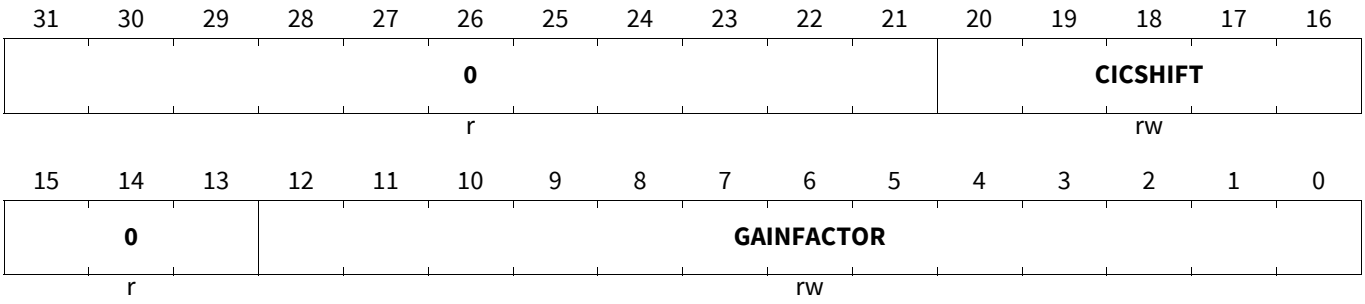
Both factors can be in the range of 0.000 to 1.999, with an initial value of 1.000.

While the calibration sequence is executing, the configuration selected for operation is replaced with the configuration selected for calibration (in register **GAINCTRx (x=0-13)**). Configure both settings before starting a channel.

*Note: An overview of the calibration mechanism is given in **"Calibration Support" on Page 36**. The formula to determine the GAINFACTOR is given in section **"Data Shifter and Decimation Factor" on Page 45**.*

Gain Correction Register x

GAINCORRx (x=0-13)
Gain Correction Register x **(0144_H+x*100_H)** **Application Reset Value: 0000 1000_H**



Field	Bits	Type	Description
GAINFACTOR	12:0	rw	**Multiplication Factor for Gain Correction** The resulting factor is (<GAINFACTOR> / 4 096)
CICSHIFT	20:16	rw	**Position of the CIC Filter Output Shifter** Selects the valid outputs bits from the CIC filter, depending on the chosen decimation factor (see formula above) $1D_H$ … $1F_H$ are reserved. 00_H Use bits 0 ... 16 … $1C_H$ Use bits 28 ... 44
0	15:13, 31:21	r	**Reserved, write 0, read as 0**

33.5.3 Overshoot Compensation

Due to their properties, the FIR filters tend to produce overshoots when the input signal changes rapidly. The overshoot compensation block monitors the data flow, detects a step in the input signal (rapid change), and recontours the signal so the FIR filters do not produce the unwanted overshoot.

Several aspects of the overshoot compensation are configurable via register **OVSCFGx (x=0-13)**:

- **Step Detection Threshold**
  Defines the numeric difference between samples that activates the overshoot compensation (SDTH). The threshold is compared to the magnitude of the difference, i.e. it is valid for both directions signal change.
- **Step Detection Mode**
  Defines if the current sample is compared to the last sample or to the second-last sample (SDM).
- **Slew Rate Filter Run Time**
  Defines the duration of the overshoot compensation (SRFRT).
- **Slew Rate Filter Strength**
  Defines the signal recontouring effect (SRFS).

The figure below shows how overshoot compensation modifies the output signal generated by the FIR filters.

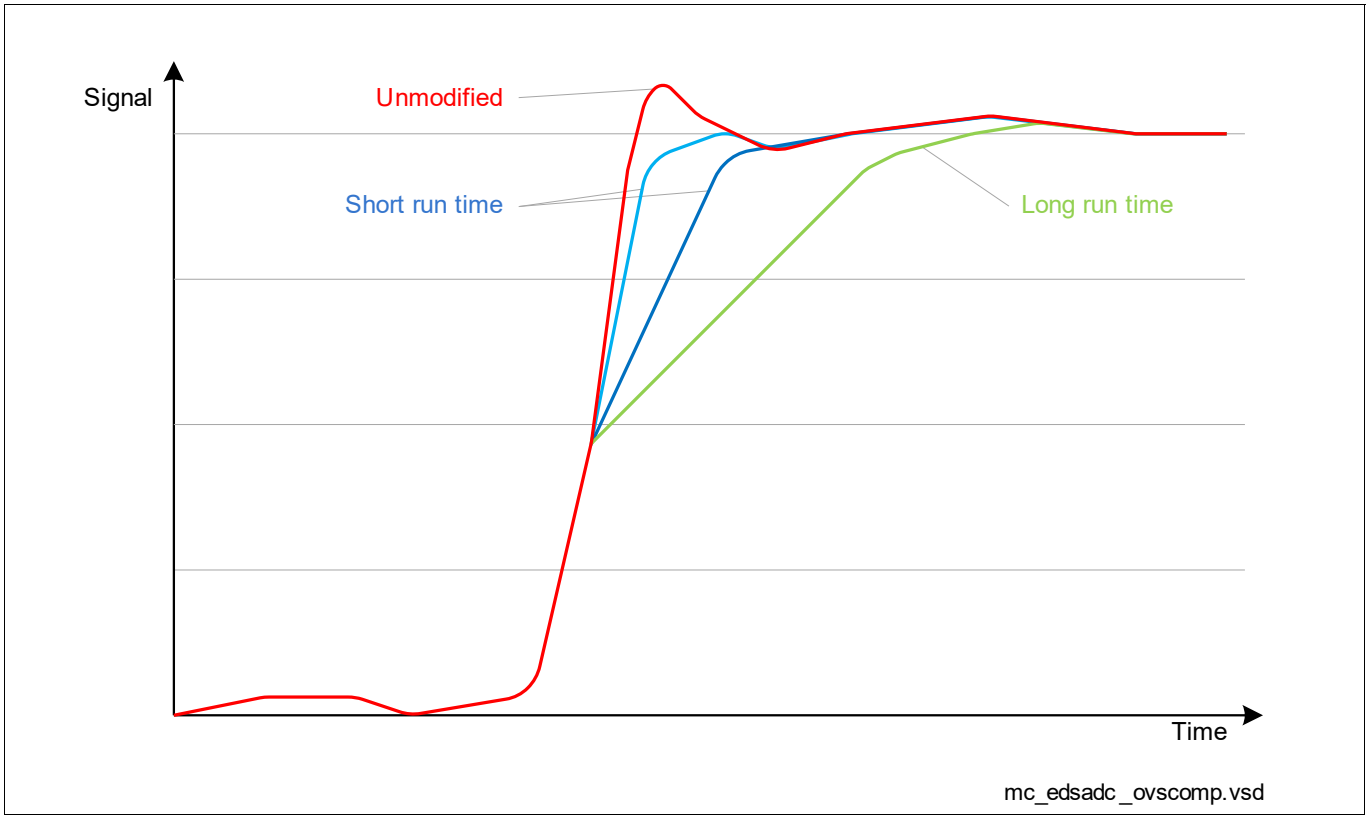


Figure 303 Overshoot Compensation Effects

Overshoot Compensation Cfg. Register x

OVSCFGx (x=0-13)

Overshoot Compensation Cfg. Register x (011C$_H$+x*100$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					SDTH										
r					rw										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											SDM	SRFRT		SRFS	
r											rw	rw		rw	

Field	Bits	Type	Description
SRFS	1:0	rw	**Slew Rate Filter Strength** Defines the time constant for the slew rate filter. 00_B Minimum filter effect, early attenuation, linear operation 01_B Weak filter effect 10_B Medium filter effect 11_B Maximum filter effect, steep beginning, smooth end
SRFRT	3:2	rw	**Slew Rate Filter Run Time** Defines the time constant for the slew rate filter. 00_B 2 input cycles 01_B 4 input cycles 10_B 8 input cycles 11_B 16 input cycles
SDM	4	rw	**Step Detection Mode** Defines when the slew rate filter is activated. 0_B Compare threshold to difference of current and last input 1_B Compare threshold to difference of current and second-last input
SDTH	26:16	rw	**Step Detection Threshold** Defines the threshold value (magnitude) used for step detection. The threshold value is <SDTH> × 32 000_H 0 (slew rate filter active all the time) 001_H 32 … $7FF_H$ 65504
0	15:5, 31:27	r	**Reserved, write 0, read as 0**

33.5.4 FIR Filters

The FIR filter further attenuates the higher frequency bands that pass the CIC filter. This improves the overall frequency response of the filter chain.

The FIR filter is realized as two subsequent FIR blocks. FIR0 adds a decimation factor of 2, FIR1 adds a decimation factor of 1 or 2, so the total decimation factor (i.e. the oversampling rate, including the CIC filter) is 2×N or 4×N.

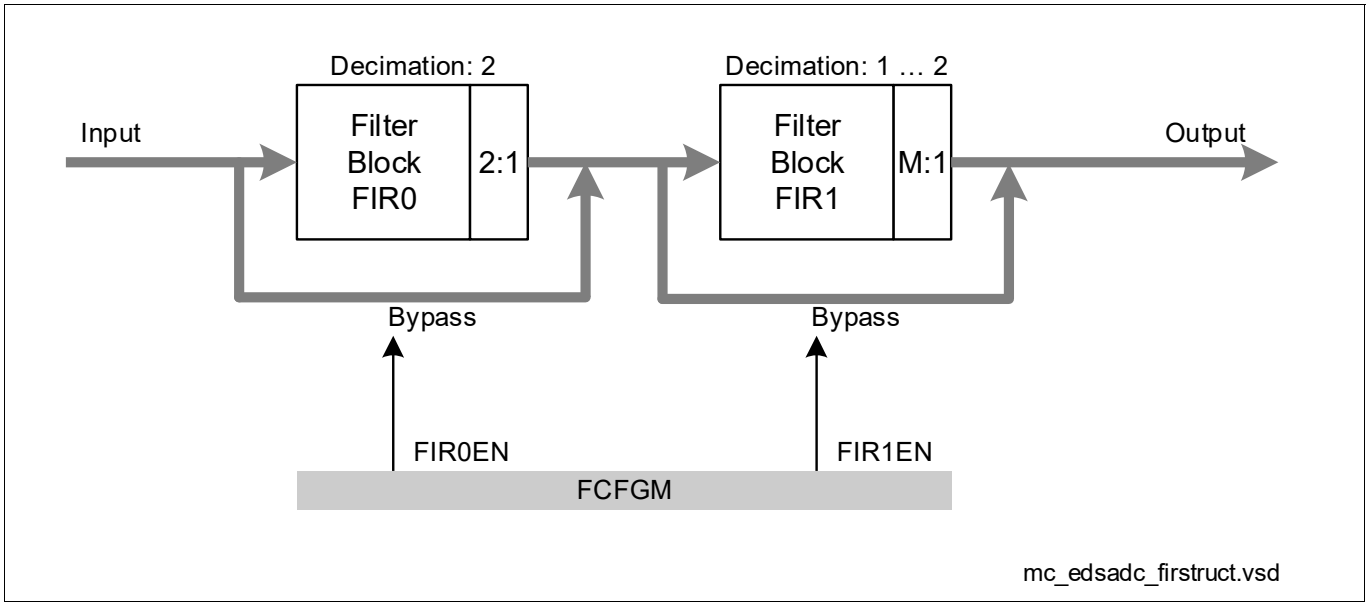


Figure 304 FIR Filter Blocks

The FIR filters can be described as:

$$y_1(m_1) = \sum_{i=0}^{N_1-1} a_1(i)\,x(2m_1 - i)$$

$$y(m) = \sum_{i=0}^{N_2-1} a_2(i)\,y_1(2m - i)$$

The coefficients of both FIR filters are fixed (see **Table 284** and **Table 285**). The filters are symmetric and have linear phase.

FIR0 has N1 = 8 coefficients, and generates the following response:

Table 284 Coefficients of FIR0

a0	a1	a2	a3	a4	a5	a6	a7
-29	-43	143	441	441	143	-43	-29

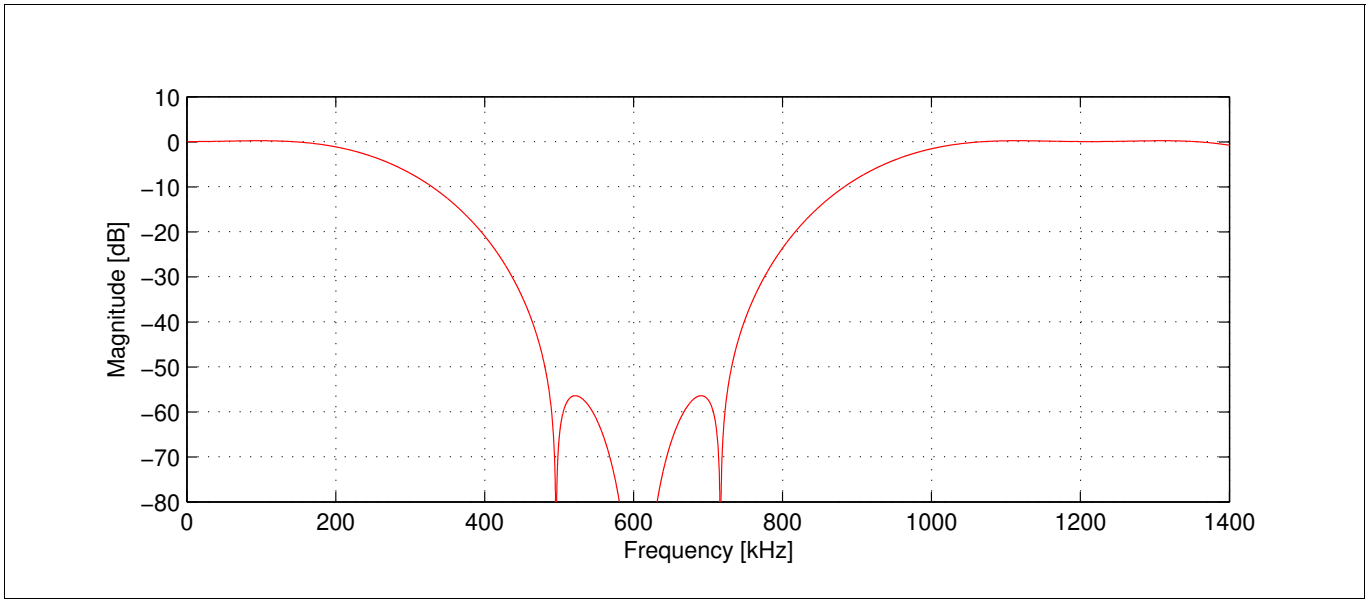


Figure 305 FIR0 Filter Frequency Response (Example for 606.06 kHz Output Data Rate)

FIR1 N2 = 28 coefficients and provides the following response:

Table 285 Coefficients of FIR1

a0, a27	a1, a26	a2, a25	a3, a24	a4, a23	a5, a22	a6, a21	a7, a20	a8, a19	a9, a18	a10, a17	a11, a16	a12, a15	a13, a14
-5	-3	3	14	-2	-22	-12	30	39	-20	-86	-19	196	399

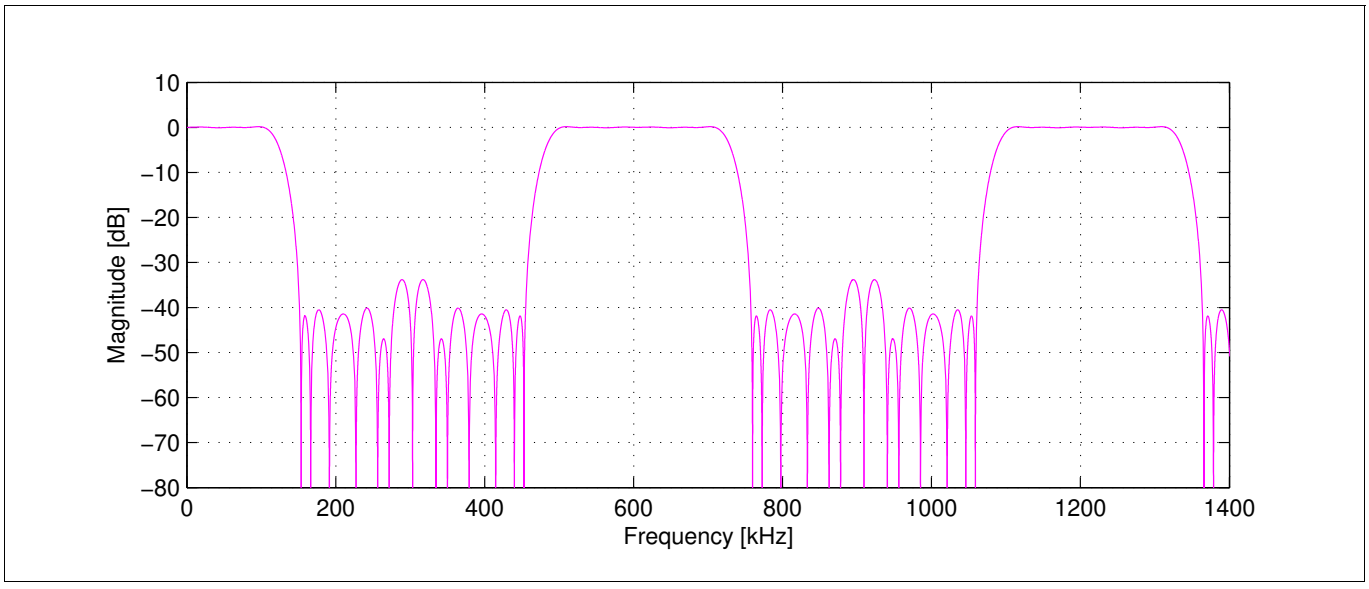


Figure 306 FIR1 Filter Frequency Response (Example for 303.03 kHz Output Data Rate)

33.5.5 Offset Compensation

The offset component that is present in most input signals can be removed automatically in order to receive the original sensor signal. The offset compensation stage can operate in two basic modes:

- **Highpass Filter**
  in this mode, the offset compensation can automatically remove the static component of a cyclic differential signal. This is achieved by the implemented IIR filter.
- **Offset Correction**
  in this mode, the offset compensation can remove the offset component of a static signal by subtracting a predefined value from the raw results. The IIR filter is disabled in this mode.

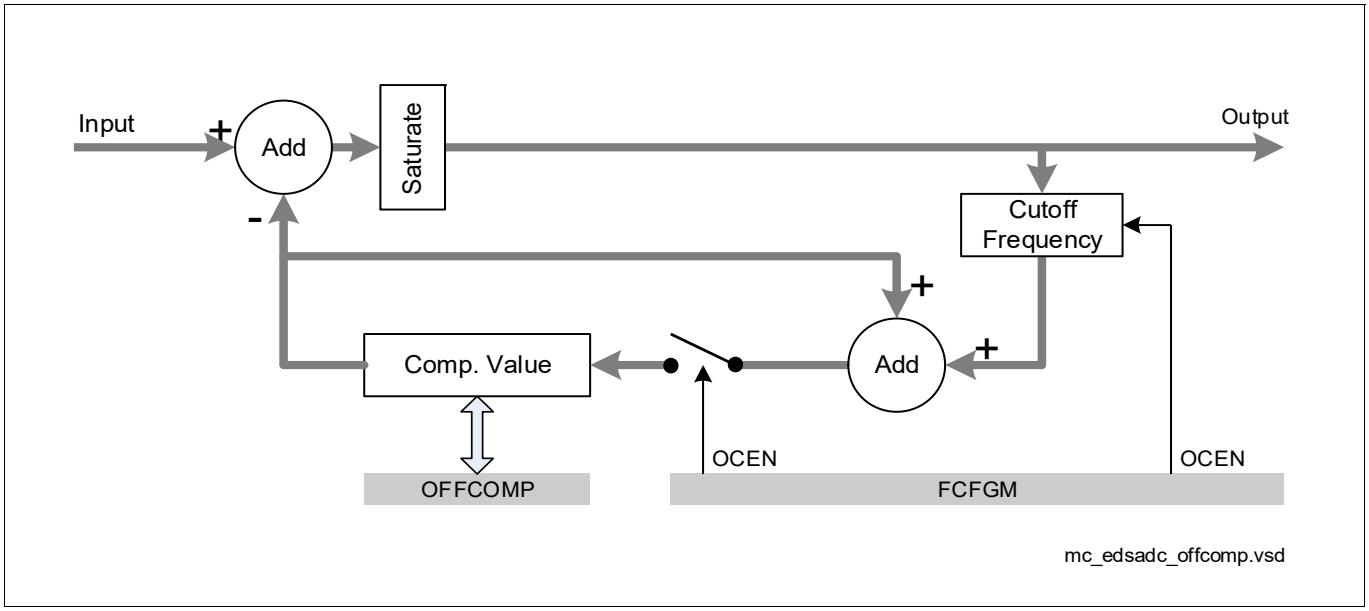


Figure 307 Offset Compensation

- **When the highpass filter is active**, the compensation value in register **OFFCOMPx (x=0-13)** is automatically updated by the filter. This attenuates the signal by a factor of at least DCF = -3 dB for frequencies below the selected cutoff frequency ($\geq 10^{-5} \times f_d$).
  The settling time of the highpass filter can be reduced by writing an initial compensation value instead of using the default value of zero. The general settling time is also adjustable by changing the filter properties, see bitfield **FCFGMx (x=0-13)**.OCEN and **Figure 308**.
- **When the highpass filter is off**, half of the current value in register **OFFCOMPx (x=0-13)** is subtracted from each raw value. The correction value can be determined and updated by the calibration sequence. Alternatively, it can be defined by the application or the calibrated value can be adjusted.
  Updating the offset value by the calibration sequence can be prohibited by setting bit OFFPROT in register **FCFGMx (x=0-13)**. This preserves the value written by the user.

*Note: An overview of the calibration mechanism is given in **"Calibration Support" on Page 36**.*

Since the filter chain internally uses more than 16 bits, the value in bitfield OFFSET is used shifted by one. This adds an additional bit to the LSB, which reduces the quantization error when several result values are accumulated. This, for example, is done when using the integrator.

A correction value of 233, therefore, is stored as $01D2_H$, $01D3_H$ would equal 233.5.

The high-pass filter can be described as (α = OCEN for OCEN > 000_B):

$$H(z) = \frac{1 - z^{-1}}{1 + (2^{2\alpha-16} - 1)z^{-1}}$$

The high-pass filter provides the following frequency responses:

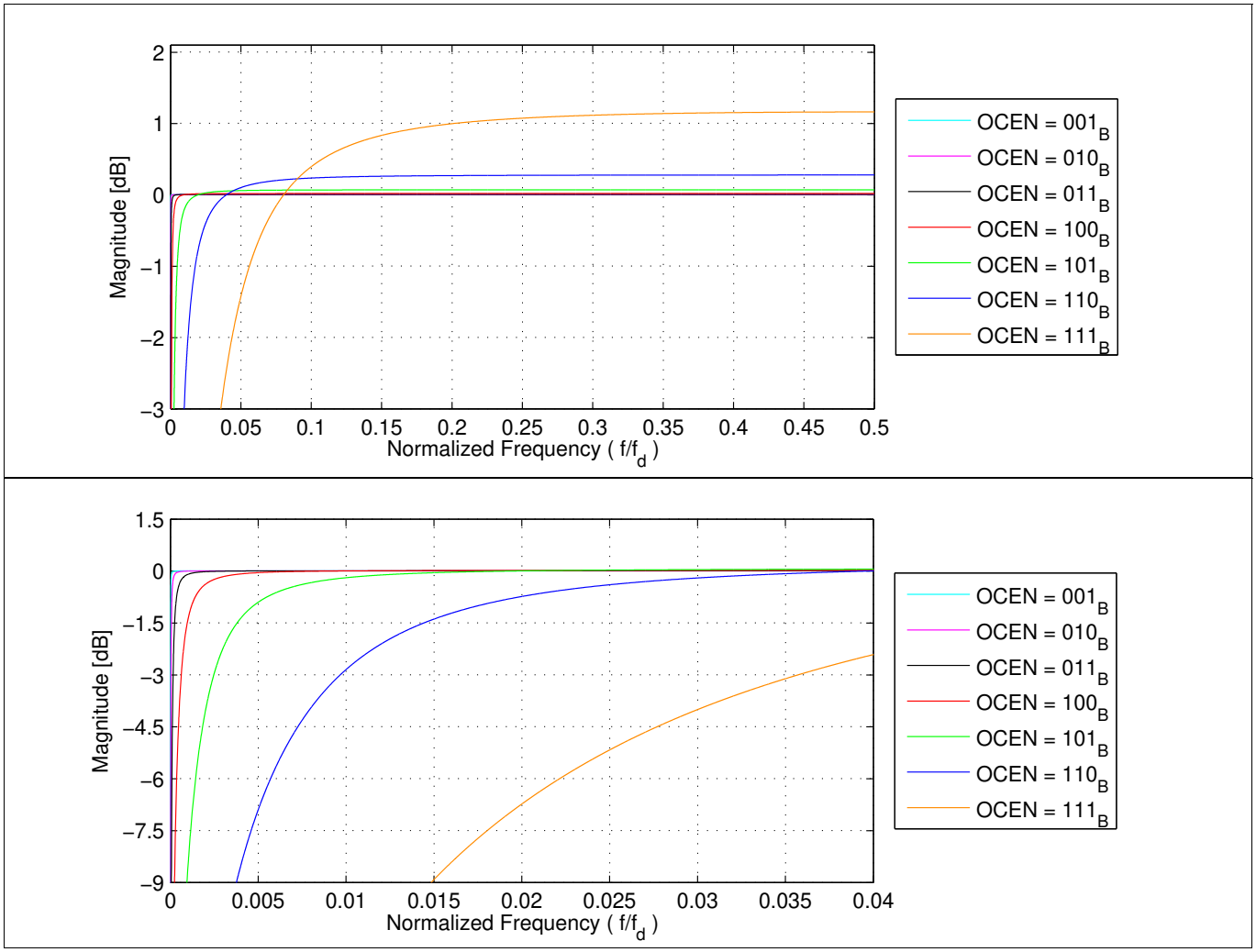


Figure 308 High-Pass Filter Frequency Responses

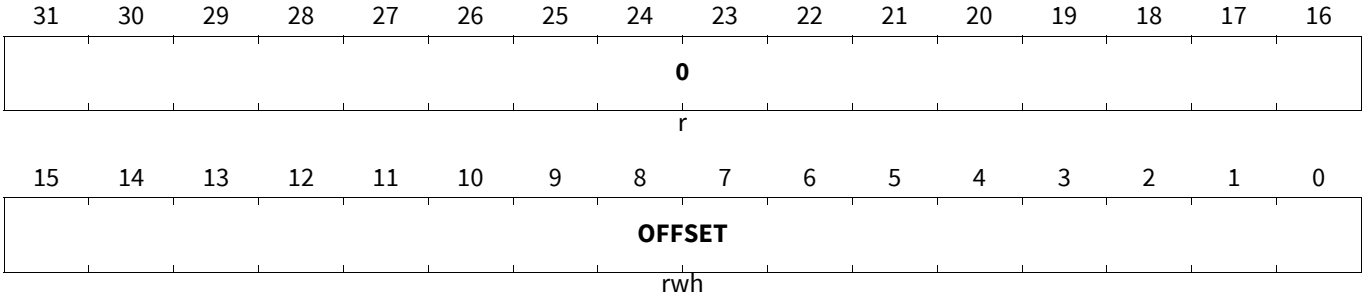
As shown in the figure above, bitfield **FCFGMx (x=0-13)**.OCEN selects different filter parameters:

Table 286 Offset Compensation Filter Parameters

OCEN	Cutoff Frequency (-3 dB)	Note
000_B	---	Offset compensation filter off
001_B	$1 \times 10^{-5} \times f_d$	Lowest cutoff frequency
010_B	$4 \times 10^{-5} \times f_d$	
011_B	$16 \times 10^{-5} \times f_d$	
100_B	$62 \times 10^{-5} \times f_d$	
101_B	$25 \times 10^{-4} \times f_d$	
110_B	$1 \times 10^{-2} \times f_d$	
111_B	$4 \times 10^{-2} \times f_d$	Highest cutoff frequency

Offset Compensation Register x

OFFCOMPx (x=0-13)
Offset Compensation Register x **(0138_H+x*100_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
OFFSET	15:0	rwh	**Offset Value** Half of this signed value is subtracted from each result produced by the filter chain. *Note: Bit 0 represents 1/2 LSB. This increases the precision in case of accumulated result values, e.g. in the integrator.*
0	31:16	r	**Reserved, write 0, read as 0**

33.5.6 Integrator Stage

The integrator integrates the result values generated during the defined integration window by adding a configurable number of values to build the final result value. The integration window can be started triggered by an internal or external signal.

A configurable number of values can automatically be discarded after the trigger before the integration window is started. This positions the integration window exactly into a timeframe where the filter is stable or where the signal to be measured is free of system-generated noise.

Integration can be used to measure currents through shunt resistors at defined positions in the signal waveform. It also can remove the carrier signal component in resolver applications. In this case, the values to be integrated can be rectified to yield the maximum amplitude of the receiver signal. The delay between the carrier signal (generated by the on-chip carrier generator) and the received position signals can be compensated automatically. also refer to **Section 33.12**.

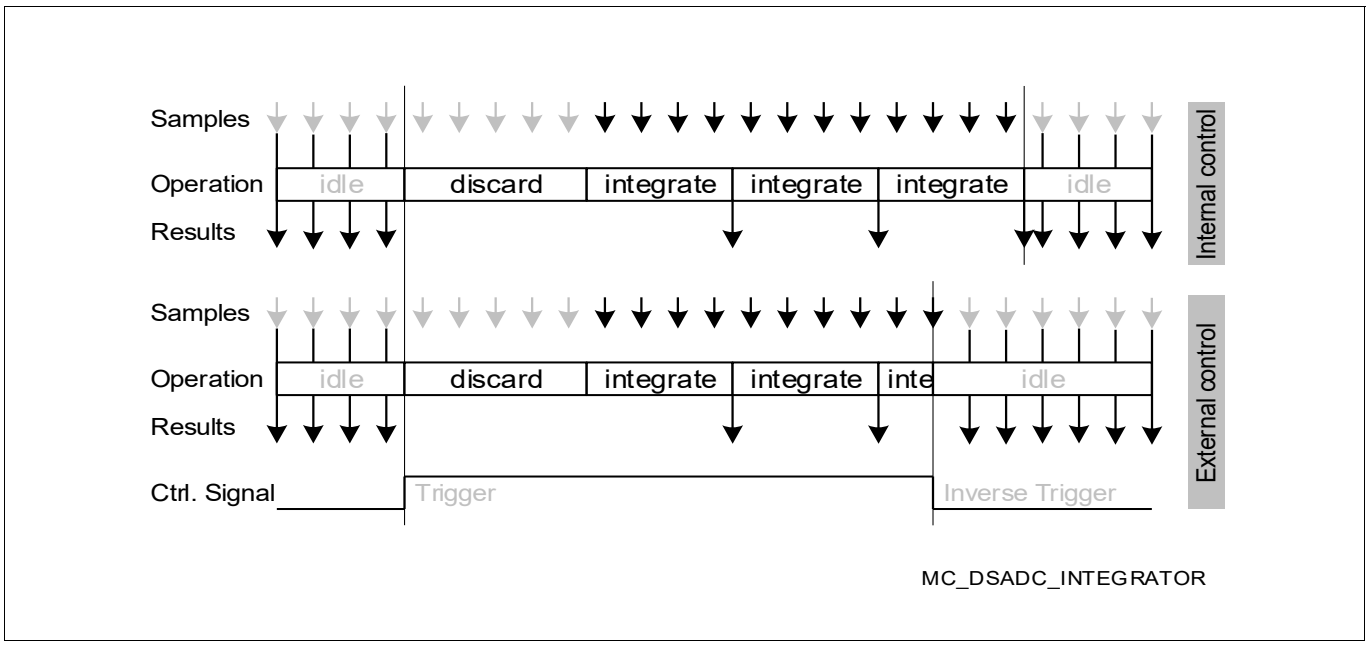


Figure 309 Integrator Operation

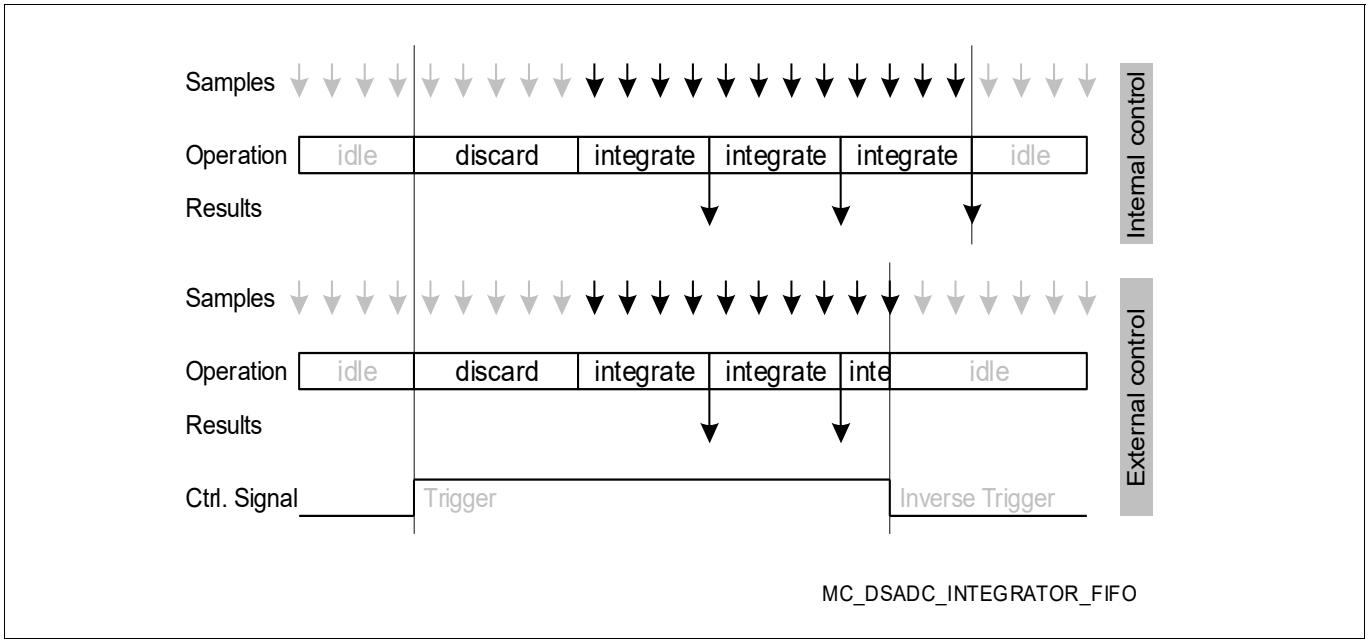


Figure 310 Integrator Operation with FIFO

Starting the Integration Window

The integration window starts when bit INTEN becomes 1:

- Software-Controlled Mode:
  Select software-controlled integration mode by setting bitfield ITRMODE = 11_B
- Trigger-Controlled Mode:
  Select trigger-controlled integration mode (ITRMODE = 01_B or 10_B)
  and generate the selected trigger edge at the configured trigger input

The external integration trigger signal is selected by bitfield TRSEL in register **DICFGx (x=0-13)**. Bitfield ITRMODE selects the transition of the selected signal to generate the trigger event. Select the trigger source first before enabling it.

When the integration window is started (INTEN becomes 1) the filter chain can be restarted (FRC = 0). This means, every non recursive filter element of the Filter Chain (CIC3, FIR0, FIR1, Integrator Stage) is cleared to zero and the related decimation counters are loaded with their start value (see **Section 33.5.1**). This ensures a deterministic start of an integration sequence. Simultaneously, the CIC filter of the Auxiliary Filter is cleared to zero and the related decimation counter is set to its initial value (see **Section 33.6**). This allows a synchronous operation of Filter Chain and Auxiliary Filter.
Keeping the filter chain running (FRC = 1) avoids the group delay until the first valid result values. In this case, the delay between start of integration window and the first result value depends on the current status of the filter chain.

After this the integration counter starts counting. After <NVALDIS> values the counter is reset and the integrator is started (if NVALDIS is zero the integration starts immediately).

Executing Integration Cycles

During an integration cycle <NVALINT+1> input values are integrated. After that, the integration result is stored in the result register and the integrator and the counter are cleared. If bit INTEN = 1 the next integration cycle is started.

Stopping the Integration Window

The integration window stops when bit INTEN becomes 0:

- Software-Controlled Mode:
  Disable software-controlled integration mode by setting bitfield ITRMODE = 00_B
- Trigger-Controlled Mode:
  - Internally controlled end of integration (IWS = 0, see upper part of **Figure 309**):
    After <REPVAL+1> integration cycles INTEN is cleared and the integration window closes after (<NVALINT+1> × <REPVAL+1>) input values
  - Externally controlled end of integration (IWS = 1, see lower part of **Figure 309**):
    Generate the selected inverse trigger edge at the configured trigger input (ITRMODE = 01_B or 10_B)
  - Integration may also be stopped by software by setting bitfield ITRMODE = 00_B
  - The source for the result register (RESMx) changes from the integrator output to the upstreamed filter chain. This means, the last result of the trigger-controlled integration is available in the result register (RESMx) only for the timeframe that is defined by the data rate period of the upstreamed filter chain. Using the result FIFO (DICFGx.DSM=0B and DICFGx.TSM=0_B), the integration result is kept in the result register up to the time where another trigger-controlled integration is initiated. For this purpose, the result FIFO has to use one of the following configuration:
    DICFGx.ITRMODE=10_B, FCFGMx.SRGM=10_B, RFCx.SRLVL=00_B
    DICFGx.ITRMODE=01_B, FCFGMx.SRGM=10_B, RFCx.SRLVL=00_B
    These configurations have the effect that service requests are only generated during integration window. In case of disabled integrator stage (ISTATx.INTEN=0_B), results generated by the upstreamed filter chain will not trigger service requests. However, results from the upstreamed filter chain will be stored in higher stages of the result FIFO. When the FIFO stages are fully loaded, all other results from the upstreamed filter chain are discarded.

After the integration has stopped, the current integration value can be read from register **IIVALx (x=0-13)**.

Integrator Data Output Format

Since the integrator accumulates a series of result values, the magnitude of its output increases depending on the size of the selected integration interval. A data shifter (controlled by bitfield ISC) compensates this for intervals of 2^N. If intervals other than 2^N are selected, the magnitude of the result value will be decreased accordingly.

Integration Window Control Register x

IWCTRx (x=0-13)
Integration Window Control Register x **(0120_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		NVALINT						0		NVALDIS					
r		rw						r		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				REPVAL				0		FRC	IWS	0	ISC		
r				rw				r		rw	rw	r	rw		

Field	Bits	Type	Description
ISC	2:0	rw	**Integrator Shift Control** Controls the data shifter after the integrator that selects the portion of the integrator data for the result register. 110_B … 111_B are reserved. *Note: ISC selects the respective bits in register IIVAL. The lowest selected bit is used for rounding and is then removed.* 000_B Select bits 4 … 20 (integration of 2 values) 001_B Select bits 5 … 21 (integration of 3 … 4 values) … 101_B Select bits 9 … 25 (integration of 33 … 64 values)
IWS	4	rw	**Integration Window Size** 0_B Internal control: stop integrator after REPVAL+1 integration cycles 1_B External control: stop integrator upon the inverse trigger event
FRC	5	rw	**Filter Chain Restart Control** 0_B Restart the filter chain when an integration window starts 1_B No influence on filter chain when an integration window starts, except for the integrator itself
REPVAL	11:8	rw	**Number of Integration Cycles** Defines the number of integration cycles to be counted by REPCNT if activated (IWS = 0). The number of cycles is REPVAL+1.
NVALDIS	21:16	rw	**Number of Values Discarded** Start the integration cycle after NVALDIS values
NVALINT	29:24	rw	**Number of Values to be Accumulated** Stop the integration cycle after NVALINT+1 values *Note: Use intervals of 2 minimum, so no data is lost due to the data shifter.*
0	3, 7:6, 15:12, 23:22, 31:30	r	**Reserved, write 0, read as 0**

Integrator Status Register x

ISTATx (x=0-13)

Integrator Status Register x **(0128_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTEN	0			REPCNT				0		NVALCNT					
rh	r			rh				r		rh					

Field	Bits	Type	Description
NVALCNT	5:0	rh	**Number of Values Counted** Counts the number of integrated values until integration is started (NVALDIS) or completed (NVALINT)
REPCNT	11:8	rh	**Integration Cycle Counter** Counts the number of integration cycles if activated (IWS = 0). This number is selected via bitfield REPVAL.
INTEN	15	rh	**Integration Enable** Indicates the activity of the integrator. 1) 0_B Integration stopped. INTEN is cleared at the end of the integration window, i.e. after REPVAL cycles or upon the inverse trigger event transition of the external trigger signal. 1_B Integration enabled. INTEN is set when the channel is started while permanent integration is selected (ITRMODE = 11_B) or upon the defined trigger event.
0	7:6, 14:12, 31:16	r	**Reserved, write 0, read as 0**

1) For the control of bit INTEN, see also bitfield ITRMODE in register **DICFGx**.

Intermediate Integration Value

IIVALx (x=0-13)

Intermediate Integration Value **(0124_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						IVAL									
r						rh									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVAL															
rh															

Field	Bits	Type	Description
IVAL	25:0	rh	**Result of most recent accumulation**
0	31:26	r	**Reserved, write 0, read as 0**

33.5.7 Group Delay and Settling Time

Data that enter a digital filter need a certain number of filter clocks before they appear at the filter's output and can be used by the system. When the digital filter chain is in steady state, the corresponding delay is defined by the group delay. The effective group delay depends on the configuration of the filter chain. **Table 287** summarizes the delays incurred by the elements of the filter chain. "N" indicates the selected oversampling rate (decimation factor) of the CIC filter. Delays are listed in modulator clock cycles.

Steady state condition is reached as soon the digital filter chain is settled. The effective settling time depends on the configuration of the filter chain. Due to the cascaded topology of the filter chain, the configuration specific settling time is defined by the step response of the terminating filter chain element. Generally, the filter chain characteristics like passband frequency (f_{PB}) and settling time are normalized to the data output rate (f_D). The data output rate (f_D) depends on the selected oversampling rate of the CIC filter and the enabled filter chain elements. **Table 288** summarizes the settling time for the different filter chain elements.

Table 287 Group Delay Summary

Filter Chain Element	Delay [t_{MOD}]	Notes
On-chip modulator	1	Delay of analog frontend
Input select/adjust unit	0.5	Synchronization of input signal
CIC3	$3 \times (2^{PRE} \times N - 1) / 2$	PRE = 0 or 1 if the prefilter is off (0) or on (1)
FIR0	$3.5 \times 2^{PRE} \times N$	To be added to CIC delay
FIR1	$13.5 \times 2^{PRE} \times N \times 2^{F0}$	F0 = 0 or 1 if FIR0 decimation is off (0) or on (1)
Offset correction/compensation	0	
Integrator	IWCTRx.NVALINT × $2^{PRE} \times N \times 2^{F0} \times 2^{F1} / 2$	Depends on the number of integrated values. F1 = 0 or 1 depending if FIR1 decimates (1) i.e. FIR1DEC = 0 or not (0) i.e. FIR1DEC = 1

Table 288 Settling Time Summary

Filter Chain Element	Settling Time [t_D]	Notes
CIC3	3 + 1	Settling time is defined by 3rd order of the CIC filter
FIR0	8 / 2 + 1	Settling time is defined by the 8 taps of FIR0 and the decimation of 2
FIR1	$28 / 2^{1\text{-FCFGMx.FIR1DEC}} + 1$	Settling time is defined by the 28 taps of FIR1 and the configurable decimation (FCFGMx.FIR1DEC)
Offset correction/compensation	$1 / (5 \times f_{-3dB})$	Cutoff frequency (f_{-3dB}) can be configured by bit-field FCFGMx.OCEN
Integrator	1 + 2	Mathematically, the integrator behave like a 1st order CIC filter

Example

f_{MOD} = 40 MHz, t_{MOD} = 25 ns,

prefilter active, OSR(CIC3) = 32, FIR0 active, FIR1 active and decimating, OC off, integration of 10 values.

$f_D = f_{MOD} / (2 \times 32 \times 2 \times 2 \times 10) = 15.625$ kHz

$t_D = 1 / f_D$

Group delay:
$[1 + 0.5 + (3 \times 63 / 2) + (3.5 \times 2 \times 32) + (13.5 \times 2 \times 32 \times 2) + (9 \times 2 \times 32 \times 2 \times 2 / 2)] \times t_{MOD} =$
$(1.5 + 94.5 + 224 + 1\,728 + 1\,152) \times t_{MOD} = 3\,200 \times 25$ ns = 80.0 µs
At the given oversampling rate of 2×32×2×2×10 = 2 560 this equals a delay of $1.25 \times t_D$ (data output rate cycles).
Without integration we get $(1.5 + 94.5 + 224 + 1\,728) \times t_{MOD} / 2{\times}32{\times}2{\times}2 = 2\,048 / 256 = 8 \times t_D$.
Settling time:
$[1 + 2] \times t_D$ = 192 µs
Without integration, data output rate (f_D) is defined by $f_D = f_{MOD} / (2 \times 32 \times 2 \times 2) = 156.25$ kHz, tD = 1/ f_D = 6.4 µs
Corresponding settling time is defined by
[28 / 2 + 1] × 6.4 µs = 96 µs

Note: When changing the input source (e.g. by switching the analog input multiplexer, if available), the second result value will safely belong to the newly selected input. Result values that correspond to the new input level, however, can only be retrieved after the corresponding group delay.

Note: With a configuration for the filter chain where only the CIC3 filter is used and the calibration is triggered (FCFGM.CALIB) immediately after modulator enabling (GLOBRC.MxRUN), the settling time of the filter chain reduces to a single cycle of t_D. The calibration algorithm uses the CIC3 filter and the settling occurs during execution of the calibration algorithm. When calibration is running, the filter chain provides no data.

33.5.8 Filter Configuration Options

The filter chain can be configured according to the requirements of the intended application. The following pages describe the available bitfields in the control registers.

Filter Configuration Register x, Main

FCFGMx (x=0-13)
Filter Configuration Register x, Main **(0110_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSRWC	0	AUTOCAL	CALIB	0			EGT	ESEL		SRGA		0		SRGM	
w	r	rw	w	r			rw	rw		rw		r		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMWC	0			OFFPROT	OCEN			0		PFEN	CICMOD	FIR1DEC	OVCEN	FIR1EN	FIR0EN
w	r			rw	rw			r		rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
FIR0EN	0	rw	**FIR0 Filter Enable** 0_B FIR0 disabled and bypassed 1_B FIR0 filter enabled
FIR1EN	1	rw	**FIR1 Filter Enable** 0_B FIR1 disabled and bypassed 1_B FIR1 filter enabled
OVCEN	2	rw	**Overshoot Compensation Enable** 0_B Disabled, feed FIR filter directly 1_B Attenuate response to fast edges
FIR1DEC	3	rw	**FIR1 Filter Decimation Rate** 0_B Decimation 2:1 for FIR1 1_B FIR1 filter does not decimate
CICMOD	4	rw	**CIC Filter Mode** 0_B CIC3 1_B Reserved
PFEN	5	rw	**Prefilter Enable** 0_B Off 1_B Prefilter enabled
OCEN	10:8	rw	**Offset Compensation Filter Enable** 000_B Offset compensation filter disabled, register OFFCOMP not changed 001_B Enable offset compensation filter, set cutoff frequency to rate 1 … 111_B Enable offset compensation filter, set cutoff frequency to rate 7

Field	Bits	Type	Description
OFFPROT	11	rw	**Offset Protection** Controls the influence of the calibration sequence on register OFFCOMP. 0_B Unprotected, calibration sequence updates bitfield OFFSET 1_B Protected, bitfield OFFSET is locked and not modified by calibration
FMWC	15	w	**Write Control for Filter Modes** 0_B No write access to filter modes 1_B Bitfields OFFPROT, OCEN, PFEN, CICMOD, FIR1DEC, OVCEN, FIR1EN, FIR0EN can be written
SRGM	17:16	rw	**Service Request Generation for Main Service Request** 00_B Never, service request disabled 01_B While gate (selected trigger signal) is high 10_B While gate (selected trigger signal) is low 11_B Always, as selected by bitfield SRLVL
SRGA	21:20	rw	**Service Request Generation for Alternate Service Request** *Note:* 00_B Never, service request disabled 01_B Comparator event, as selected by bitfield ESEL/EGT 10_B Timestamp event 11_B Alternate source: Capturing of a sign delay value to register CGSYNC
ESEL	23:22	rw	**Event Select** Defines when a comparator event is generated. 00_B Always, for each new result value 01_B If result is inside the boundary band 10_B If result is outside the boundary band 11_B Reserved
EGT	24	rw	**Event Gating** Defines if alarm events are coupled to the integration window. 0_B Separate: generate events according to ESEL 1_B Coupled: generate events only when the integrator is enabled and after the discard phase defined by bitfield NVALDIS[1)]
CALIB	28	w	**Calibration Trigger** 0_B No action 1_B Start the calibration algorithm now
AUTOCAL	29	rw	**Automatic Calibration Control** 0_B Calibration algorithm started by software (set bit CALIB) 1_B Automatically start the calibration algorithm when the selected service request gate closes (trailing edge, see bitfield SRGM)
CSRWC	31	w	**Write Control for Calibration and Service Request Modes** 0_B No write access to calibration and service request modes 1_B Bitfields AUTOCAL, CALIB, EGT, ESEL, SRGA, SRGM can be written

Field	Bits	Type	Description
0	7:6, 14:12, 19:18, 27:25, 30	r	**Reserved, write 0, read as 0**

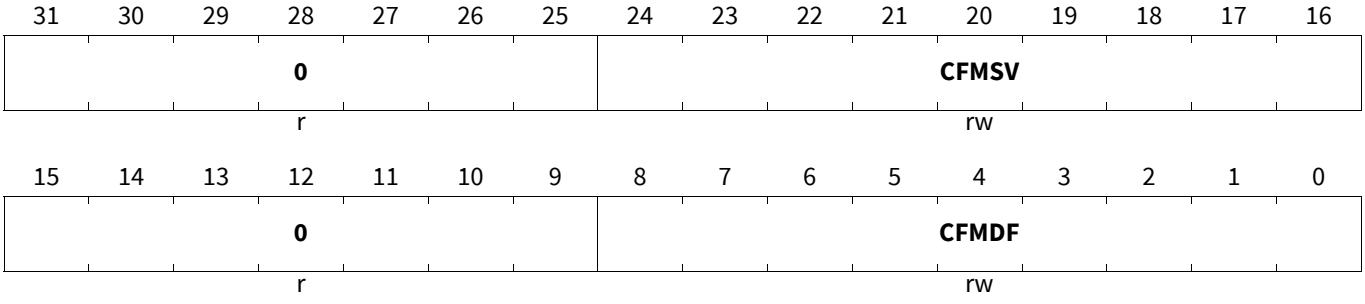
1) While the integrator is bypassed, event gating suppresses alarm service requests, result values are still generated and stored.

Table 289 Access Mode Restrictions of FCFGMx (x=0-13) sorted by descending priority

Mode Name	Access Mode		Description
write 1 to **FMWC**	rw	CICMOD, FIR0EN, FIR1DEC, FIR1EN, OCEN, OFFPROT, OVCEN, PFEN	Set FMWC during write access
write 1 to **CSRWC**	rw	AUTOCAL, EGT, ESEL, SRGA, SRGM	Set CSRWC during write access
write 1 to **CSRWC**	w	CALIB	Set CSRWC during write access
otherwise	r	AUTOCAL, CICMOD, EGT, ESEL, FIR0EN, FIR1DEC, FIR1EN, OCEN, OFFPROT, OVCEN, PFEN, SRGA, SRGM	
(default)	rX	CALIB	

Filter Configuration Register x, CIC Filter

FCFGCx (x=0-13)
Filter Configuration Register x, CIC Filter (0114_H+x*100_H) Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
CFMDF	8:0	rw	**CIC Filter Decimation Factor** Defines the oversampling rate of the CIC filter: OSR = CFMDF + 1. Valid values are 003_H to $1FF_H$ (OSR = 4 to 512).
CFMSV	24:16	rw	**CIC Filter Start Value** The decimation counter begins counting at value CFMSV, when started or restarted. Valid values are 003_H to CFMDF (4 to selected OSR).[1)]

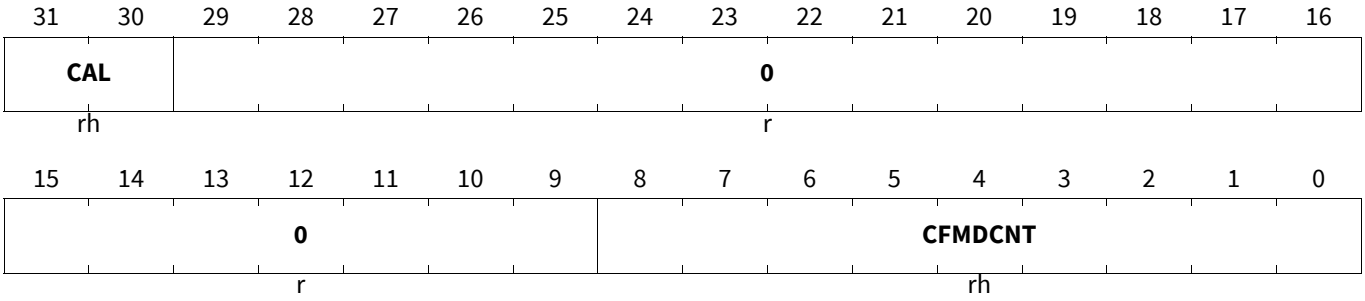
Field	Bits	Type	Description
0	15:9, 31:25	r	**Reserved, write 0, read as 0**

1) Start values above the selected oversampling rate may lead to overflows in the CIC filter!

Filter Counter Register x, CIC Filter

FCNTCx (x=0-13)

Filter Counter Register x, CIC Filter **(0118_H+x*100_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CFMDCNT	8:0	rh	**CIC Filter Decimation Counter** The decimation counter counts the filter cycles until an output is generated, i.e. the oversampling rate. CFMDCNT counts down from the respective start value.
CAL	31:30	rh	**Calibration Status Flag** *Note: Bitfield CAL is set to 01_B in the next clock cycle after setting bit CALIB or after detecting the selected trigger (if auto-calibration is activated).* 00_B Uncalibrated, initial state after reset 01_B The calibration algorithm is currently running 10_B Calibrated, normal operation is possible 11_B Calibration terminated incorrectly
0	29:9	r	**Reserved, write 0, read as 0**

33.6 Auxiliary Filter

The parallel auxiliary filter uses a simple CIC filter for decimation. The decimation rate is restricted to 16 or 32. This reduces the filter delay, so the auxiliary filter can be used to supervise the input signal and detect abnormal input values, such as overcurrent, earlier than the main filter chain.

Note: Limit checking using the parallel auxiliary filter at a low decimation rate produces results with a reduced resolution, but generates an alarm earlier than the threshold values are seen at the output of the regular filter chain.

Depending on the application, alarms can be generated either from the main filter chain or from the auxiliary filter. This is selected via register **FCFGAx (x=0-13)**.
For the functionality of the comparator, refer to **"Limit Checking" on Page 79**.

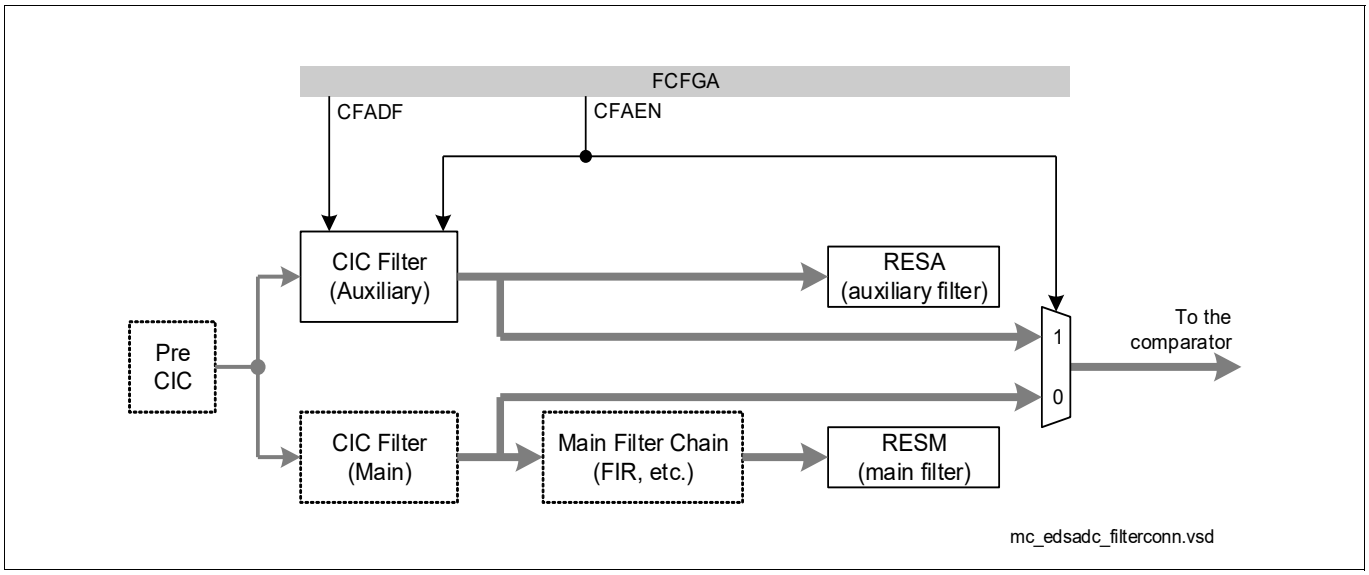


Figure 311 Connection of the Auxiliary Filter

Result Register x Auxiliary

RESAx (x=0-13)
Result Register x Auxiliary **(0180_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
rh															

Field	Bits	Type	Description
RESULT	15:0	rh	**Most Recent Result of Auxiliary Filter**
0	31:16	r	**Reserved, write 0, read as 0**

Auxiliary Filter Configuration Register x

FCFGAx (x=0-13)

Auxiliary Filter Configuration Register x **(0170_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											CFACNT				
r											rh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														CFADF	CFAEN
r														rw	rw

Field	Bits	Type	Description
CFAEN	0	rw	**CIC Filter (Auxiliary) Enable** 0_B Off: Auxiliary filter is not active 1_B Auxiliary filter is active and generates results and alarm events
CFADF	1	rw	**CIC Filter (Auxiliary) Decimation Factor** 0_B OSR = 16 1_B OSR = 32
CFACNT	20:16	rh	**CIC Filter (Auxiliary) Decimation Counter** The decimation counter counts the filter cycles until an output is generated, i.e. the oversampling rate.
0	15:2, 31:21	r	**Reserved, write 0, read as 0**

33.7 Time-Stamp Support

Some applications need to determine the result value at certain points of time inbetween two regular output values. The interpolation algorithm needs to determine the position of the required point of time in relation to the regular results.

This interpolation is supported by providing a timestamp that marks the delay since the last regular output value. This timestamp is the number of modulator clock cycles that have passed since the last result value has been generated. Depending on the configured decimation/integration factors this defines the fraction of an output cycle and, therefore, enables the interpolation.

The timestamp information is generated by the timestamp counter. This 16-bit counter is clocked with the optionally prescaled modulator clock (see **TSCNTx (x=0-13)**.TSCLK). It covers the following cycles:

- CIC filter decimation: up to 512
- FIR filter decimation: up to 4
- Integrator decimation: up to 64

The timestamp information is captured into the timestamp register upon a trigger event. For this purpose, the trigger signal is used, which is selected by bitfield TRSEL in register **DICFGx (x=0-13)**. Bitfield TSTRMODE selects the edge(s) that capture(s) timestamp information.

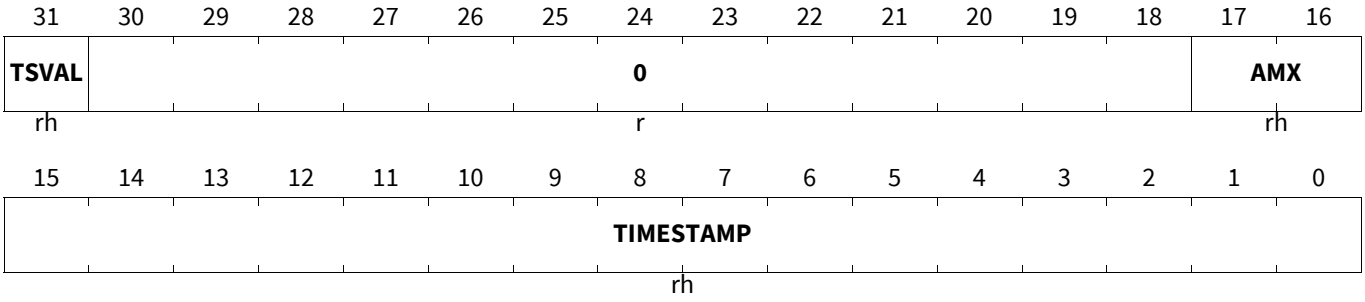
A timestamp trigger can generate a service request if selected by bitfield SRGA in register **FCFGMx (x=0-13)**.

The valid flag TSVAL is set when timestamp information is stored and is cleared when register TSTMP is read. This indicates to the application software if or not a timestamp trigger has occurred before the corresponding result value was generated.

When timestamp information is stored, a service request can optionally be generated. This allows application software to operate on all result data, even if multiple timestamp triggers occur before a regular result service request. The timestamp data is also available via the standard result register (see **Table 290** and **"Result Service Request Generation and Read Sequencing" on Page 76**).

Time-Stamp Register x

TSTMPx (x=0-13)
Time-Stamp Register x **(0150_H+x*100_H)** **Application Reset Value: 0000 0000_H**



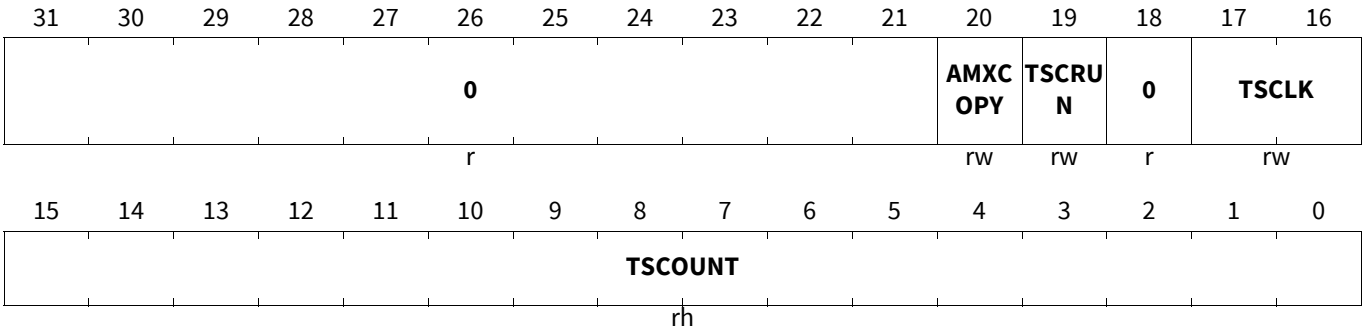
Field	Bits	Type	Description
TIMESTAMP	15:0	rh	**The Most Recent Captured Timestamp Value** This value is copied from the timestamp counter TSCOUNT *Note: If bit AMXCOPY in register **TSCNTx** is 1, bits TIMESTAMP[15:14] are replaced with a copy of bitfield AMX.*
AMX	17:16	rh	**Analog Multiplexer Setting** This value is copied from bitfield INMUX in register MODCFGx

Field	Bits	Type	Description
TSVAL	31	rh	**Timestamp Valid** Indicates valid timestamp information. 0_B No timestamp trigger occurred since last read access 1_B Timestamp information has been stored after a timestamp trigger
0	30:18	r	**Reserved, write 0, read as 0**

Time-Stamp Counter x

TSCNTx (x=0-13)

Time-Stamp Counter x **(0154_H+x*100_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TSCOUNT	15:0	rh	**Timestamp Counter Value** TSCOUNT is clocked with the modulator clock and is cleared when a new result value has been generated.
TSCLK	17:16	rw	**Timestamp Counter Clock Selection** 00_B $f_{TSTMP} = f_{MOD}$ 01_B $f_{TSTMP} = f_{MOD} / 2$ 10_B $f_{TSTMP} = f_{MOD} / 4$ 11_B $f_{TSTMP} = f_{MOD} / 8$
TSCRUN	19	rw	**Timestamp Counter Run Control** 0_B Timestamp counter is off 1_B Timestamp counter is counting at the rate selected by bitfield TSCLK
AMXCOPY	20	rw	**Analog MUX Setting Copy Enable** Allows copying of bitfield AMX into bitfield TIMESTAMP (in register **TSTMPx**). 0_B Do not copy, timestamp uses all 16 bits 1_B Copy AMX to bits TIMESTAMP[15:14], timestamp uses lower 14 bits
0	18, 31:21	r	**Reserved, write 0, read as 0**

33.8 Conversion Result Handling

The EDSADC preprocesses the conversions result data before storing them for retrieval by the CPU or a DMA channel.

Conversion result handling comprises the following functions:

- **Filtering and Post-Processing**
- **Storage of Conversion Results** to FIFO and result register
- **Result Service Request Generation and Read Sequencing**
- **Hardware Data Interface** provides result values to other modules

33.8.1 Filtering and Post-Processing

The result data words are generated by feeding the input data stream through a chain of filter elements and decimating it by a selectable ratio. The selectable integrator can further reduce the output data rate while executing accumulation and averaging.

Several elements of the filter chain can be bypassed, i.e. the filter chain is configurable and its behavior can be adapted to the requirements of the actual application.

The result values are multiplied by a factor that serves for gain calibration and data format normation. An offset can be subtracted automatically from each result value before being fed to the integrator or being written to the result register.

For differential mode applications, the offset can alternatively be removed automatically by the high-pass filter.

Due to the differential input stage, the results are signed values. Usually, these results are stored in a 16-bit two's-complement format. For the specific quasi-differential operating modes (single-ended input using common mode voltage) the results can be stored as 16-bit unsigned integer values (see **DICFGx (x=0-13)**).

The CIC filter's output value depends on the selected filter parameters and decimation factor. A data shifter extracts the most significant bits from this filter result. A multiplier adjusts the magnitude of the result values to the result range required by the application.

33.8.2 Storage of Conversion Results

The conversion result values are stored in a FIFO-structure which is accessible via the result register RESMx.

This enables the following features:

- Buffering increases the allowable latency for retrieving result values
- Two result values can be retrieved by one single read access
- Two subsequent read accesses enable the efficient transfer of up to 4 result values[1)]
- The previous result value is still available to calculate interpolation values for timestamp operation

Also timestamp information can be accessed through the result register (either directly or via the FIFO). This enables access to all relevant data using a single DMA channel.

To optimize access to result values, the result register can be read in several modes:

Table 290 Result Register Read Modes

Read Mode[1)]	RESMx[31:16] (high)	RESMx[15:0] (low)	Notes
Single-word read mode, sign-extended ($DRM = 00_B$) (SSSS'RRRR)	Extended sign value (only valid for result values)	Next result value (from FIFO stage 1)	TSM = 0, 16-bit read access[2)] ($SRLVL = 00_B, 01_B, 10_B, 11_B$)
		Timestamp[3)], then initial result value, then next result value (from FIFO stage 1)	TSM = 1, gate-controlled timestamp mode, 16-bit read access[2)] ($SRLVL = 10_B, 11_B$)
Single-word read mode ($DRM = 01_B$) (0000'RRRR, TTTT'RRRR)	0000_H	Recent result value (FIFO not used, overwrite RESULTLO)	TSM = 0, 16-bit read access
	Timestamp	Recent result value (FIFO not used, overwrite RESULTLO)	TSM = 1, 32-bit read access
Double-word read mode ($DRM = 10_B$) (NNNN'RRRR)	Subsequent res. value (from FIFO stage 2)	Next result value (from FIFO stage 1)	TSM = 0 32-bit read access[4)], (low bus load) ($SRLVL = 01_B, 11_B$)
	Initial result value, then subsequent value (from FIFO stage 2)	Timestamp[3)], then next result value (from FIFO stage 1)	TSM = 1, gate-controlled timestamp mode, 32-bit read access[4)] ($SRLVL = 11_B$)

1) Selected by bitfields DRM and TSM in register **DICFGx (x=0-13)**.
2) Due to the sign extension, result values can also be read as signed 32-bit values.
3) The timestamp is inserted once when the selected gate opens. The FIFO is flushed when the selected gate closes.
4) In double-word read mode, a service request shall only be generated when the result double buffer holds 2 values.

FIFO Control

Result values are only written to the FIFO ($DRM = X0_B$) while service requests are enabled. This enables the application e.g. to read stored values after the service request gate has closed.The FIFO is flushed when the service request gate opens, so it provides an actual set of result values.

In gate-controlled timestamp mode, the FIFO is flushed when the gate closes, so it can store the current result value in stage 2.

1) Also refer to **"Result Service Request Generation and Read Sequencing" on Page 76**.

Result Handling Without FIFO

In a special single-word read mode (DRM = 01_B) the result values are stored in RESMx directly.

The low word returns the latest result value, the high word either is cleared or returns the current timestamp counter value.

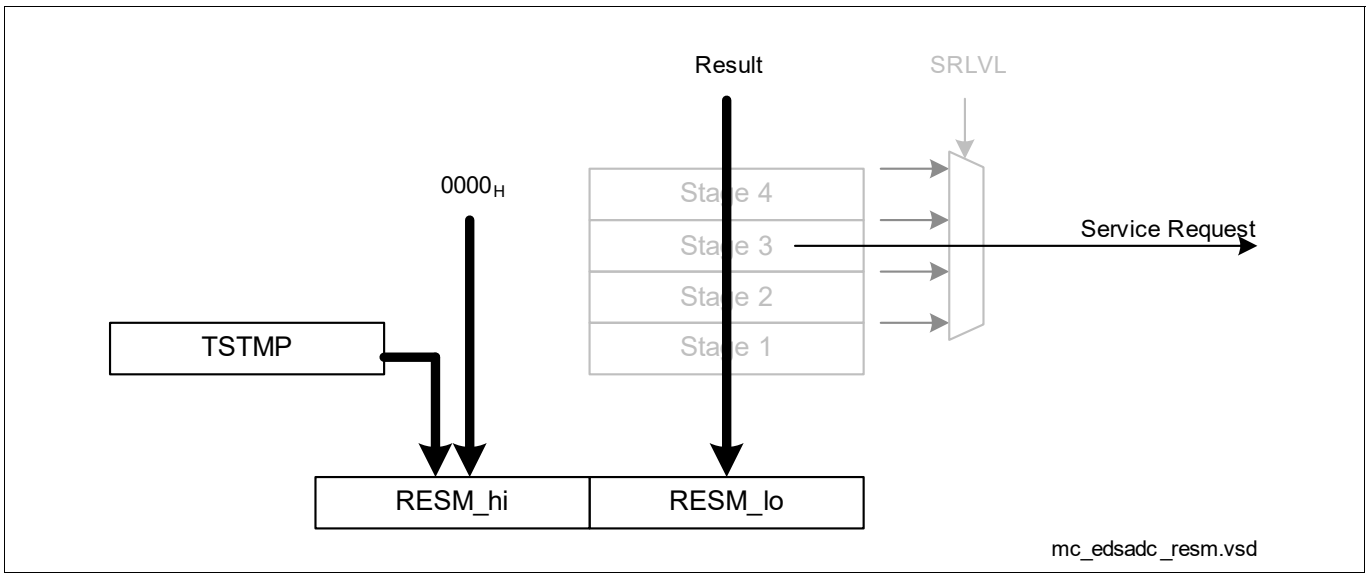


Figure 312 Direct Result Storage

Result Handling Via FIFO

The result values are not directly written to RESMx but are stored in a FIFO structure. From there they are retrieved when RESMx is read. In timestamp mode, the FIFO is transparent while the gate signal is inactive, i.e. all input values are directly forwarded to stage 2 of the FIFO. When the gate opens (this is the trigger event) a timestamp is generated and inserted to stage 1 of the FIFO. Subsequent input values are then piled into the FIFO.

A service request is generated when a certain number of values has been stored in the FIFO. The respective FIFO fill level is selected in bitfield SRLVL.

FIFO control and status bitfields are available in register **RFCx (x=0-13)**.

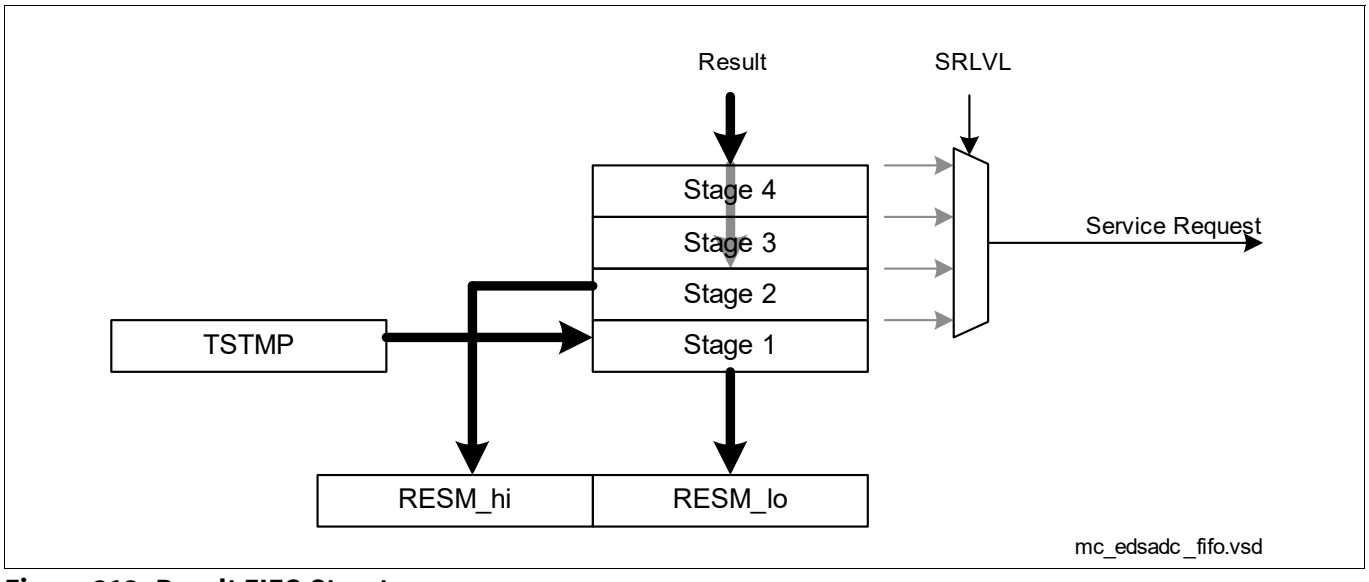


Figure 313 Result FIFO Structure

Result FIFO Control Register x

RFCx (x=0-13)

Result FIFO Control Register x **($012C_H$+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										WRERR	RDERR	0	FILL		
r										rh	rh	r	rh		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									FIFL	WREC	RDEC	0		SRLVL	
r									w	w	w	r		rw	

Field	Bits	Type	Description
SRLVL	1:0	rw	**Service Request FIFO Level** 00_B Generate service request when FIFO fill level reaches 1 value 01_B Generate service request when FIFO fill level reaches 2 values 10_B Generate service request when FIFO fill level reaches 3 values 11_B Generate service request when FIFO fill level reaches 4 values
RDEC	4	w	**Read Error Flag Clear** 0_B No action 1_B Clear flag RDERR
WREC	5	w	**Write Error Flag Clear** 0_B No action 1_B Clear flag WRERR
FIFL	6	w	**FIFO Flush** 0_B No action 1_B Remove all entries from result FIFO
FILL	18:16	rh	**FIFO Fill Level** Not listed combinations are reserved. 000_B Result FIFO is empty 001_B Result FIFO contains 1 valid value 010_B Result FIFO contains 2 valid values 011_B Result FIFO contains 3 valid values 100_B Result FIFO contains 4 valid values
RDERR	20	rh	**Read Error Flag** 0_B No problem encountered 1_B A read access occurred while the FIFO was empty A read error is also indicated when a read access occurs during the FIFO's synchronization stall phase (4 clock cycles after a read access). Clear this sticky flag by writing 1 to bit RDEC.
WRERR	21	rh	**Write Error Flag** 0_B No problem encountered 1_B A write access occurred while the FIFO was full Clear this sticky flag by writing 1 to bit WREC.

Field	Bits	Type	Description
0	3:2, 15:7, 19, 31:22	r	**Reserved, write 0, read as 0**

Result Register x Main

RESMx (x=0-13)

Result Register x Main **(0130_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESULTHI															
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULTLO															
rh															

Field	Bits	Type	Description
RESULTLO	15:0	rh	**Result Value Lower Part** Returns the next value from the result FIFO (Result or timestamp, see **Table 290**)
RESULTHI	31:16	rh	**Result Value Higher Part** Returns an additional value (Sign extension, result from FIFO, timestamp, or zero, see **Table 290**)

33.8.3 Result Service Request Generation and Read Sequencing

The generated result values (including timestamp values, if enabled) are retrieved in a defined sequence. This sequence depends on the selected result register read mode, the availability of result/timestamp values, and the system's response to service requests.

Result events are generated for each available result value. Result service requests are generated depending on the configured read mode and the FIFO fill level. Each data transfer (result data values or timestamp information) is initiated by a service request. This service request is issued when a defined number of values becomes available.

Figure 314 shows different situations for standard data transfers:

- Single Read:
  The service request is generated when one result value is available
  and is serviced by 16-bit read access (D0, D1)
- Double Read:
  The service request is generated when two result values are available
  and is serviced by 32-bit read access (D3/D2, D5/D4)
- Single Transfer Mode:
  The service request is answered with a single transfer.
  This transfer either transfers 16 or 32 bits.
- Double Transfer Mode:
  The service request is answered with two subsequent transfers[1].
  These transfers either transfer 16 or 32 bits.

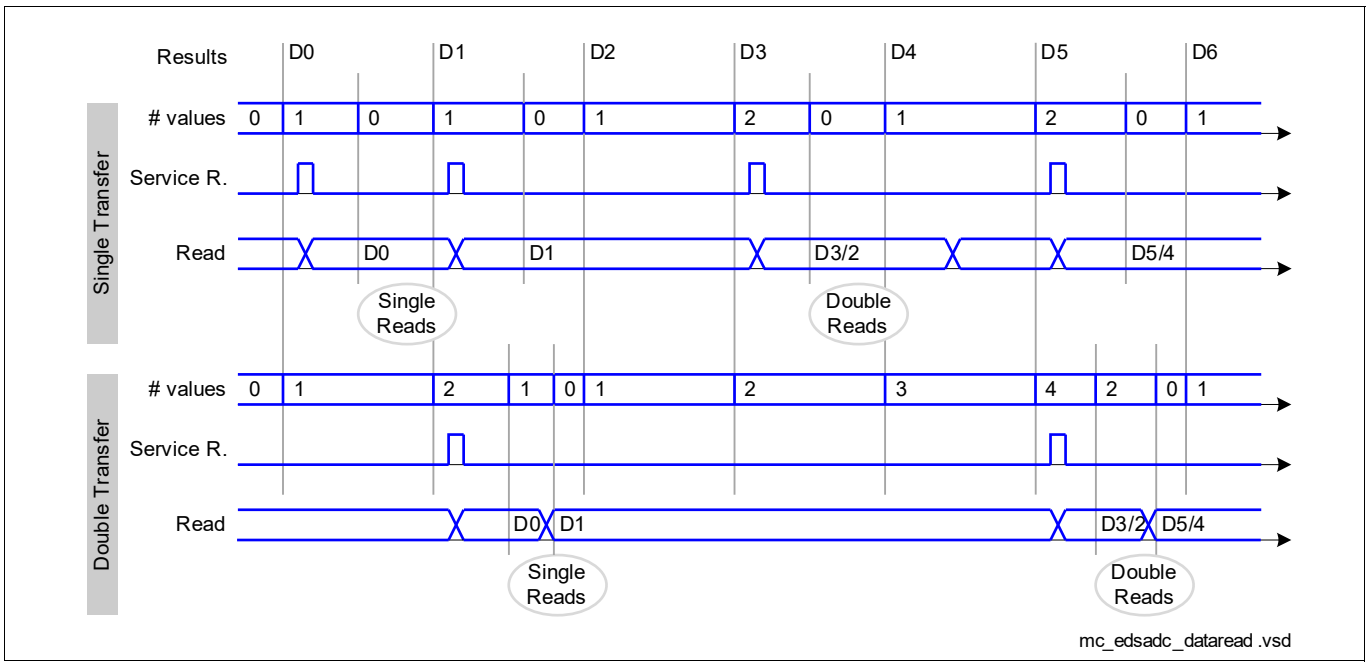


Figure 314 Standard Read Sequences

1) Synchronizing the result values between the two clock domains and controlling the result FIFO takes up to 8 cycles of f_{SPB}. Read accesses, therefore, must have at least 8 cycles of f_{SPB} in between them.

Timestamp mode supports applications using service request gating. A timestamp is generated when the gate opens (defined timestamp trigger). When the configured number of values (including the timestamp) is available, a service request is generated.

Figure 315 shows different situations for timestamp usage:

- Single Read:
  A timestamp trigger is generated when the service request gate opens. Read accesses return the timestamp value, the result before the timestamp (D0), and then subsequent result values. Note that D1 may be generated shortly after the timestamp event.
- Double Read:
  A timestamp trigger is generated when the service request gate opens. Read accesses return the result before the timestamp (D0) and the timestamp value, and then subsequent result value pairs. Accumulating 4 values and transferring them with two subsequent transfers provides the most efficient way to store data.

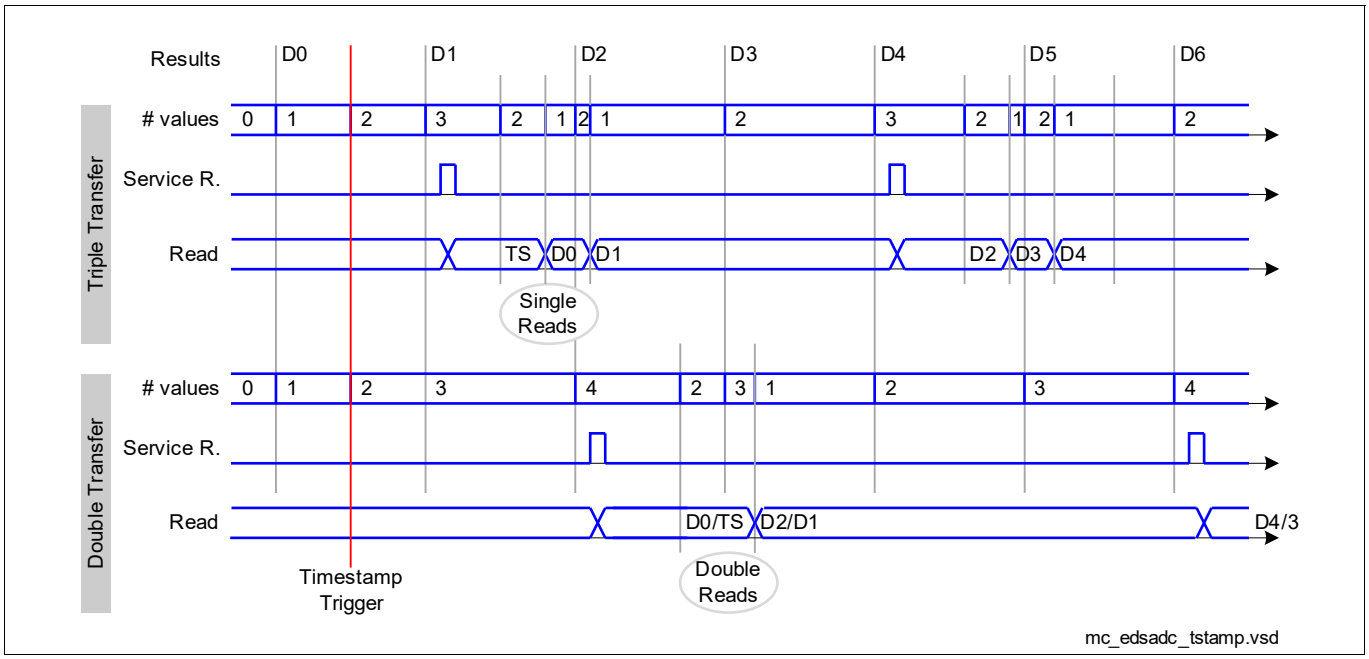


Figure 315 Read Sequences With Timestamp

Note: When setting up the service request level and e.g. DMA functionality, note that timestamp value and subsequent data value (D1) may be generated within a short timeframe, i.e. there are then 3 values in the FIFO.

33.8.4 Hardware Data Interface

The digital conversion results are directly available to other modules via the hardware data interface (HDI). Each value that is written to the result FIFO structure is also output to this interface. Each time a result event is generated, the HDI updates the data vector and generates a write strobe indicating the availability of a new result value.

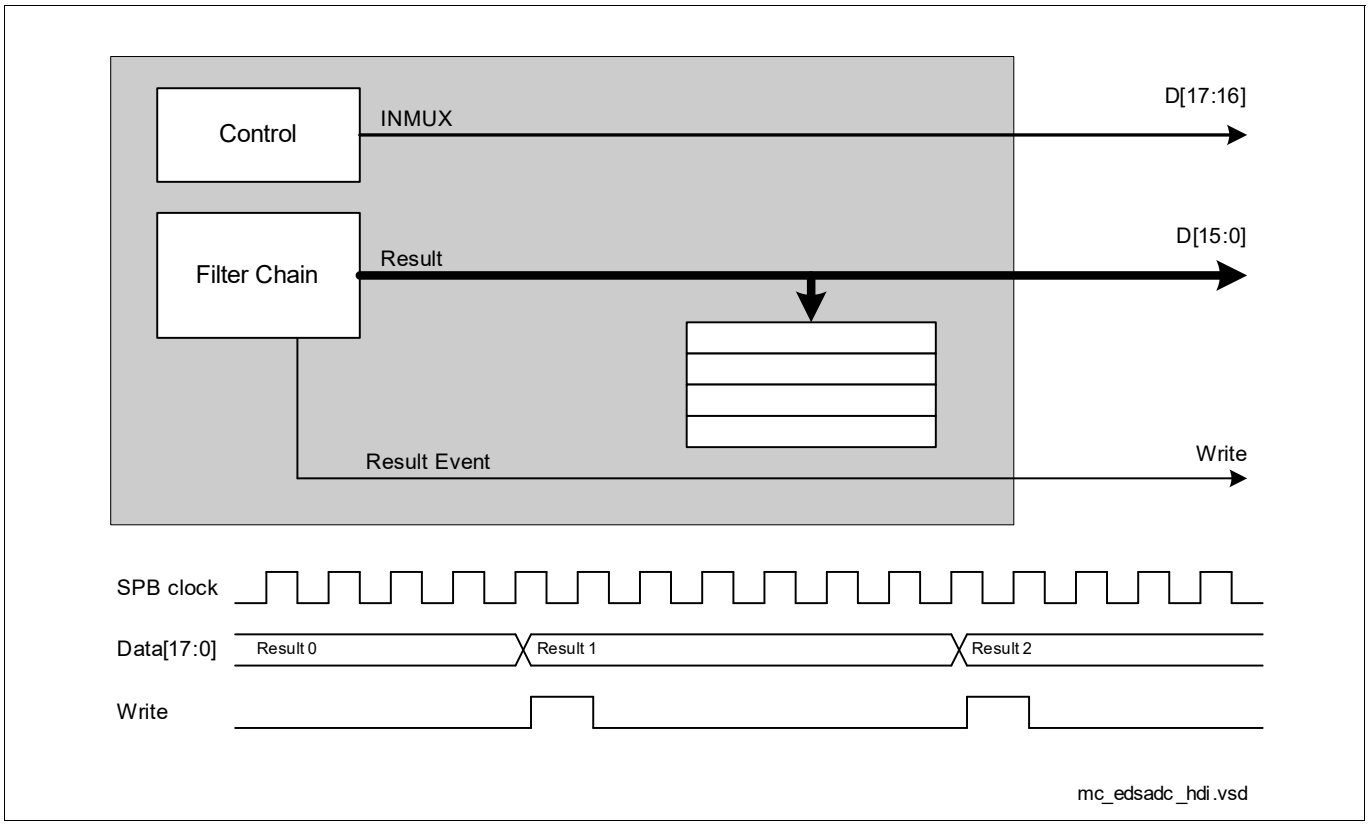


Figure 316 Hardware Data Interface

This interface writes results to the GTM to make them available for the MCSs equipped with an analog data interface (ADI).

The following data elements are available through the HDI:

Table 291 HDI Data Assignment

Bit Position	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Data Content							MUX		Conversion Result															

Note: The availability of the MUX indicator bits depends on the respective channel. Unused bits are 0.

33.9 Limit Checking

A comparator provides automatic limit checking by comparing each result to two configurable reference values. This feature can be used to supervise the input signal and detect abnormal input values. The comparator can generate a separate service request.

The two reference values are defined in the select register **BOUNDSELx (x=0-13)** and determine the valid result value band.

The result values can be taken either from the main filter chain or from the auxiliary filter. Please refer to **"Auxiliary Filter" on Page 67**.

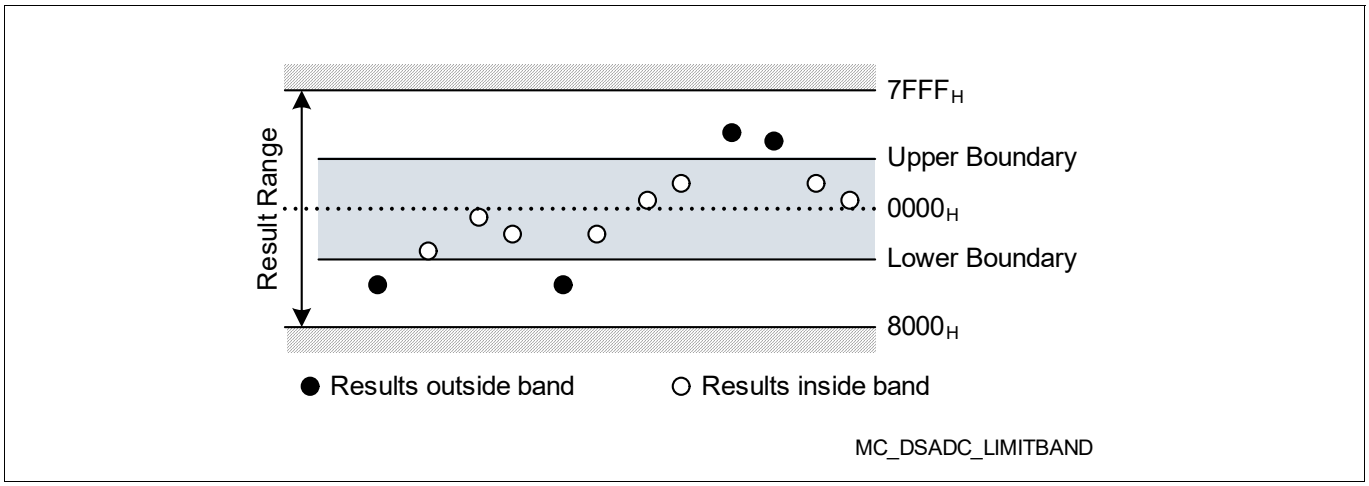


Figure 317 Result Monitoring through Limit Checking

A result value is considered inside the defined band when both of the following conditions are true:

- the value is less than or equal to the selected upper boundary
- the value is greater than or equal to the selected lower boundary

The result range can also be divided into two areas:
To select the lower part as valid band, set the lower boundary to the minimum value (8000_H) and set the upper boundary to the highest intended value.
To select the upper part as valid band, set the upper boundary to the maximum value ($7FFF_H$) and set the lower boundary to the lowest intended value.

The limit checker can generate two types of output:

- Service requests, optionally restricted by the comparators
- Range signals, indicating the result level with respect to the defined limits:
  - Signal SAULx, indicating when the results are above the upper limit
  - Signal SWIBx, indicating when the results are within the defined band
  - Signal SBLLx, indicating when the results are below the lower limit

An alarm event can be generated when a new conversion result becomes available. Alarm events can be restricted to result values that are inside or outside a user-defined band (see **Figure 317**). This feature supports automatic range monitoring and minimizes the CPU load by issuing service requests only under certain conditions. For example, an input value can be monitored and an alarm indicates a certain threshold.

Alarm events can also be suppressed completely. Bitfields SRGA and ESEL in register **FCFGMx (x=0-13)** select the service request generation mode.

The range signals (SAULx, SWIBx, SBLLx) are generated independent of service requests.

Boundary Select Register x

BOUNDSELx (x=0-13)

Boundary Select Register x **(0178_H+x*100_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BOUNDARYU															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOUNDARYL															
rw															

Field	Bits	Type	Description
BOUNDARYL	15:0	rw	**Lower Boundary Value for Limit Checking** This (two's complement) value is compared to the upper bits of the CIC filter results.
BOUNDARYU	31:16	rw	**Upper Boundary Value for Limit Checking** This (two's complement) value is compared to the upper bits of the CIC filter results.

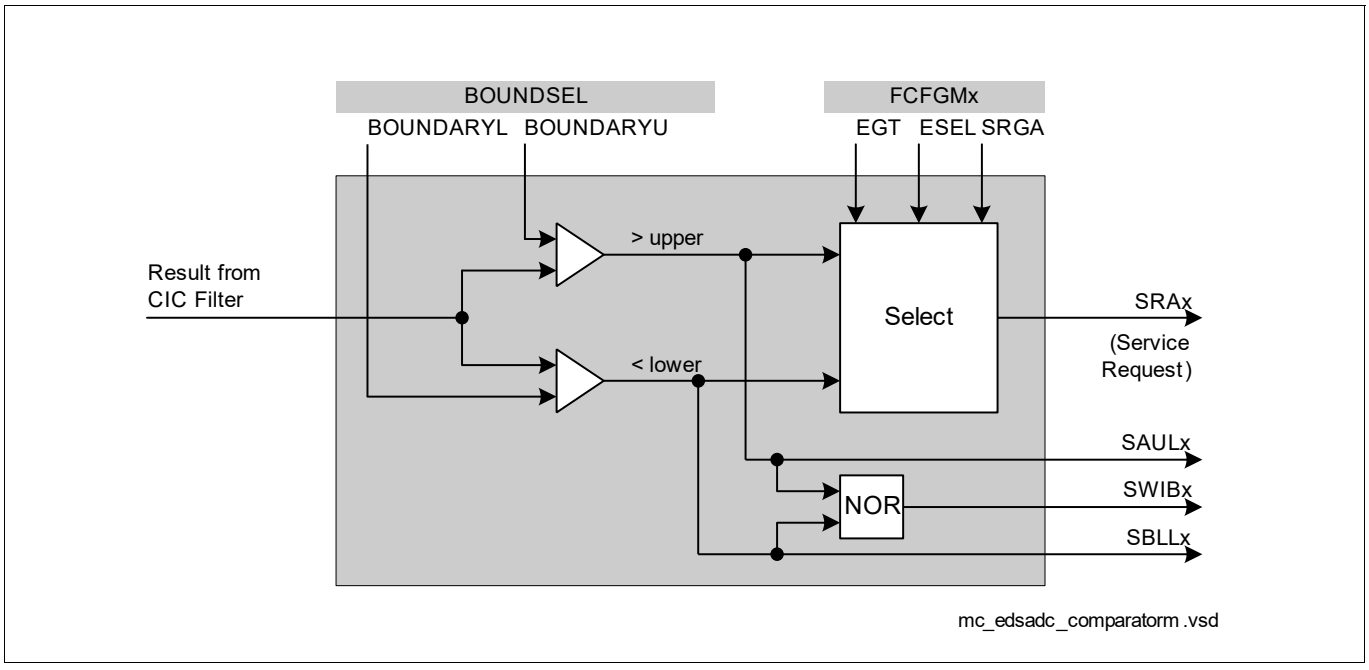


Figure 318 Comparator Structure

Selecting the Boundary Values

The comparator can be fed with result values from either the main CIC filter or from the auxiliary CIC filter. Since these filters generate result values in different formats, also the corresponding boundary values must fit to the selected format.

- The auxiliary CIC filter generates results in a fixed 1Q15 format. Due to the intrinsic gain of the on-chip modulator full-scale ranges from -22 757 to +22 757. When using an external modulator, the full-scale values depend on the gain of the used modulator.
- The main CIC filter generates results in a user-configurable format, where the full-scale value is defined by bitfield GAINCALx.CALTARGET.

For proper operation, the boundary values must relate to the selected filter's data format.

Note: There is no calibration in the auxiliary filter path. Offset and gain error must, therefore, be handled by the application software.
*The respective values can be obtained from registers **GAINCALx (x=0-13)** and **OFFCOMPx (x=0-13)** after calibration.*

33.10 Safety Features

Most analog inputs are connected to both EVADC and EDSADC channels, thus providing a basic redundancy.

On-Chip Supervision Signals

Information about the basic functionality of the EDSADC can be obtained via special on-chip signals, which supports common cause diagnosis. Every channel can output a replica of a reference voltage generated by a bandgap inside the power management system (PMS). The selected output voltage can be measured via a specific channel of the EVADC.

The selection of the supervision signal is controlled centrally via register **GLOBCFG**:

- Bitfield SVSIG enables the supervision function by selecting one of two voltages
- Bitfield SVCH selects the channel for which the supervision signal is connected to the common output.

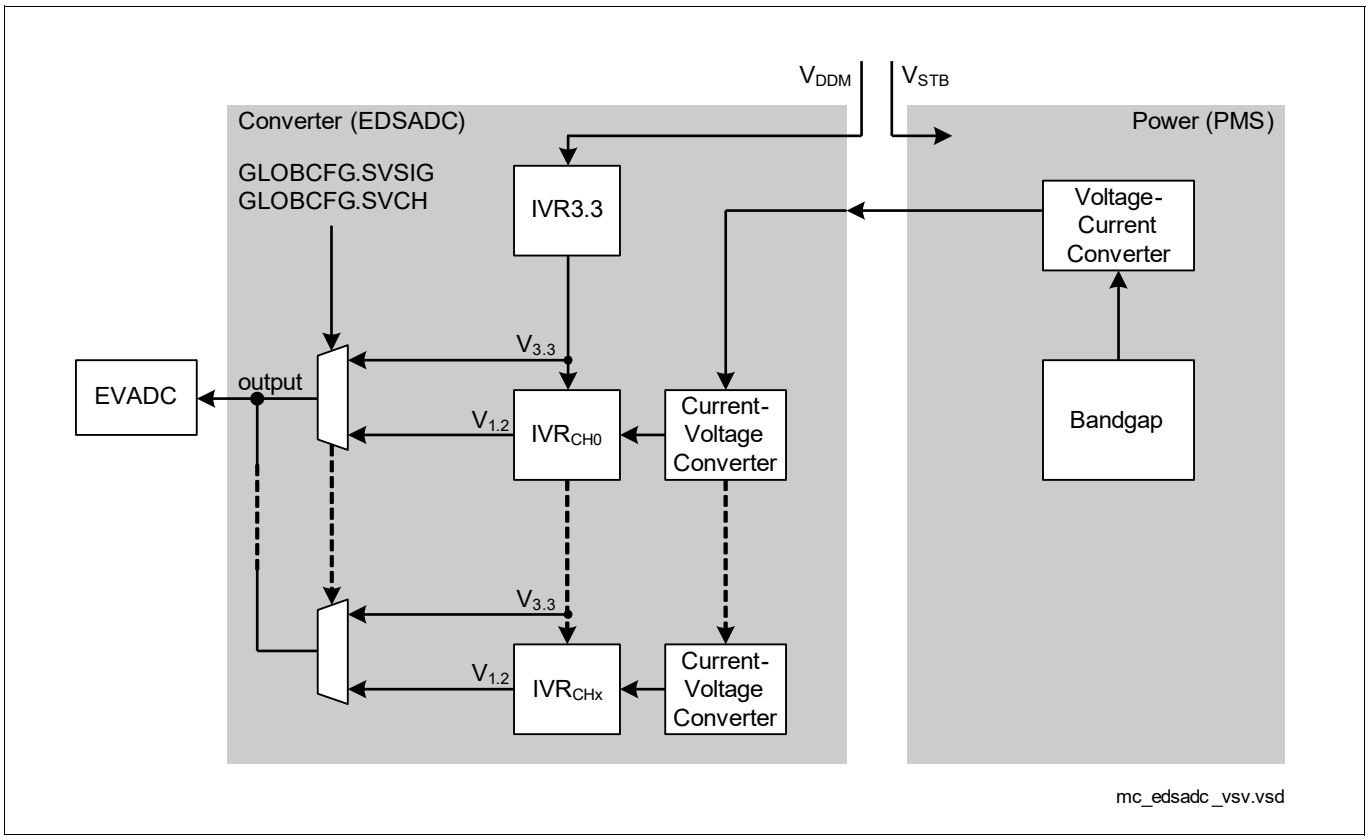


Figure 319 Test Voltages for Supervision

Measuring these voltages enables two test features:

- Compare the result with the expected value (RESULT = V_{Test} / V_{AREF} × 2^{12}).
  With V_{AREF} = 5.0 V and V_{DDK} = 1.2 V, RESULT = $3D7_H$.
- Compare the individual results of all channels to find deviations.

33.11 Service Request Generation

The EDSADC can activate service request output signals to issue an interrupt, to trigger a DMA channel, or to trigger other on-chip modules. Each channel provides 2 service request lines, SRxM and SRxA.

The product-specific appendix details the available service request connections.

Several events can be assigned to each service request output. Service requests can be generated by several types of events:

- **Result events:** indicate the availability of new valid results. Usually, this triggers a read action by CPU or DMA. Result events are generated at the output rate of the configured filter chain (indicated by bits RESEVx in register **EVFLAG**). The output rate of result service requests depends on the selected FIFO fill level (see bitfield SRLVL in register **RFCx (x=0-13)**).
  Can be issued via SRxM.
- **Alarm events:** indicate that a conversion result value is within a programmable value range. This offloads the CPU/DMA from background tasks, i.e. a service request is only activated if the specified conversion result range is met or exceeded.
  Can be issued via SRxA.
- **Special events:** indicate specific circumstances of previously configured functions.
  - Timestamp trigger event can generate a service request.
    Can be issued via SRxM (see read sequences) or SRxA (separate).
  - Capture event for sign delay measurement can generate a service request.
    Can be issued via SRxA.

Each event is indicated by a dedicated flag that can be cleared by software. If a service request is enabled for a certain event, the service request is generated for each event, independent of the status of the corresponding event indication flag. This ensures efficient DMA handling of EDSADC events (the event can generate a service request without the need to clear the indication flag).

Note: Event flags SDCVAL are cleared when the filter chain is initialized,
i.e. when the channel is started (CHxRUN = 1), when an integration window is started with FCR = 0, or when the calibration algorithm begins or ends.
Event flags ALEVx, TSVAL and RESEVx are cleared when the channel is started (CHxRUN = 1).

Note: The following registers provide a set of bits for each available channel.
The number of available channels depends on the chosen device type.

Event Flag Register

The register below shows the maximum configuration.

Other products of the family may have less channels and, consequently, less valid RESEVx/ALEVx flags.

EVFLAG
Event Flag Register **($00E0_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**ALEV13**	**ALEV12**	**ALEV11**	**ALEV10**	**ALEV9**	**ALEV8**	**ALEV7**	**ALEV6**	**ALEV5**	**ALEV4**	**ALEV3**	**ALEV2**	**ALEV1**	**ALEV0**
r		rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**RESEV13**	**RESEV12**	**RESEV11**	**RESEV10**	**RESEV9**	**RESEV8**	**RESEV7**	**RESEV6**	**RESEV5**	**RESEV4**	**RESEV3**	**RESEV2**	**RESEV1**	**RESEV0**
r		rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
RESEVx (x=0-13)	x	rwh	**Result Event** *Note: Bit RESEVx is cleared when result register RESMx is read, or when bit RESECx is written with 1.* 0_B No result event 1_B New result value is generated by the filter chain
ALEVx (x=0-13)	x+16	rwh	**Alarm Event** 0_B No alarm event 1_B An alarm event has occurred
0	15:14, 31:30	r	**Reserved, write 0, read as 0**

Event Flag Clear Register

The register below shows the maximum configuration. Other products of the family may have less channels and, consequently, less valid RESECx/ALECx control bits.

EVFLAGCLR
Event Flag Clear Register **($00E4_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**ALEC13**	**ALEC12**	**ALEC11**	**ALEC10**	**ALEC9**	**ALEC8**	**ALEC7**	**ALEC6**	**ALEC5**	**ALEC4**	**ALEC3**	**ALEC2**	**ALEC1**	**ALEC0**
r		w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**RESEC13**	**RESEC12**	**RESEC11**	**RESEC10**	**RESEC9**	**RESEC8**	**RESEC7**	**RESEC6**	**RESEC5**	**RESEC4**	**RESEC3**	**RESEC2**	**RESEC1**	**RESEC0**
r		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RESECx (x=0-13)	x	w	**Result Event Clear** 0_B No action 1_B Clear bit RESEVx
ALECx (x=0-13)	x+16	w	**Alarm Event Clear** 0_B No action 1_B Clear bit ALEVx
0	15:14, 31:30	r	**Reserved, write 0, read as 0**

Note: Software can set flags RESEVx and ALEVx and trigger the corresponding event by writing 1 to the respective bit. Writing 0 has no effect.
Software can clear these flags by writing 1 to bit RESECx and ALECx, respectively.

33.12 Resolver Support

Resolver applications determine the rotation angle by evaluating the signals from two orthogonally placed coils. These coils are excited by the magnetic field of a third coil.
Several features are available to support these applications by offering the almost complete interface hardware (except for power stages) and by preprocessing the input data (rectification, integration) to optimize evaluation by higher level software algorithms.

33.12.1 Resolver System Overview

The EDSADC can read the two return signals using two input channels and can also generate the excitation sine signal (carrier). It also provides synchronization logic to compensate the delay between the generated carrier signal and the received position signals. The integrator stage converts the carrier-based return signals to position-based values (carrier cancellation).

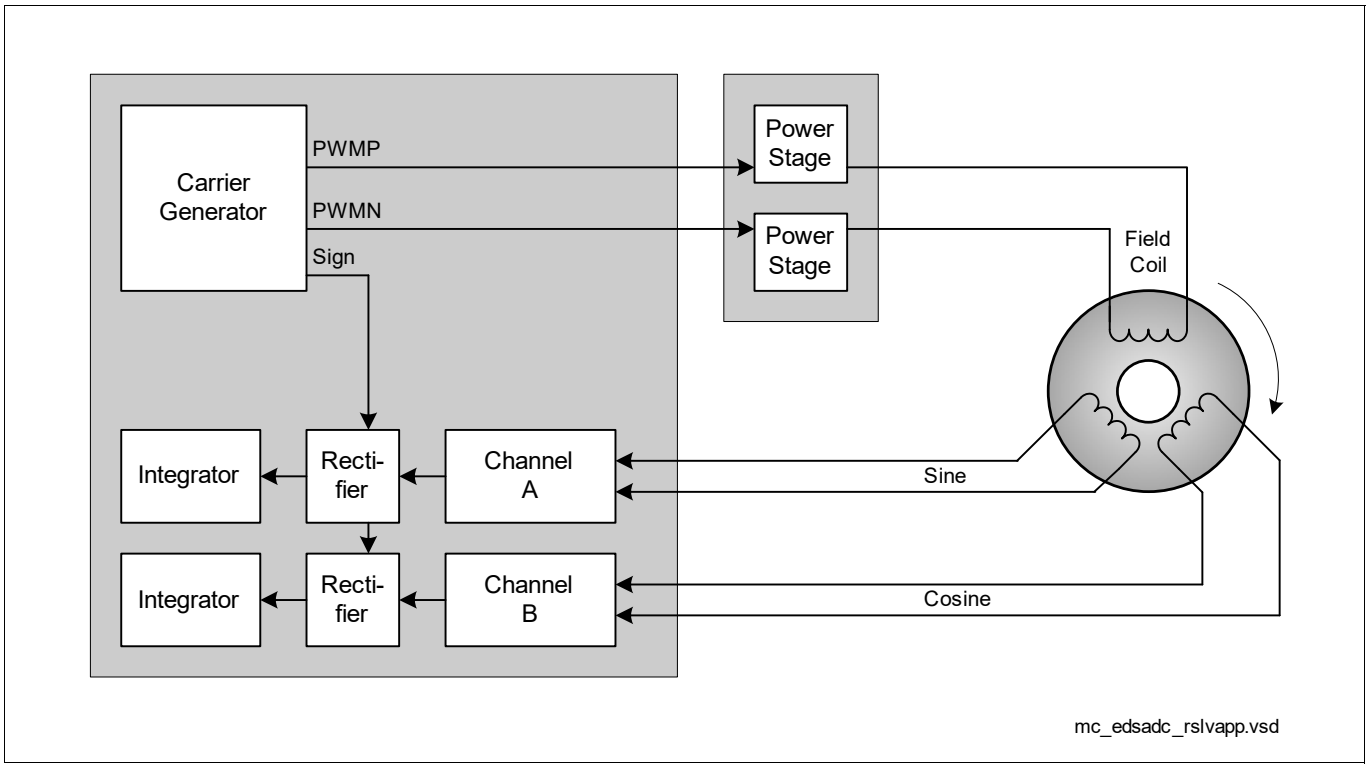


Figure 320 Resolver Application

33.12.2 Carrier Signal Generation

The carrier signal generator (CG) is supplied with the module clock signal and outputs a PWM signal that induces a sine signal in the excitation coil of the resolver. Alternatively, it can generate PWM patterns that resemble triangle or square signals (see **Figure 322**). The polarity of the carrier signal can be selected.

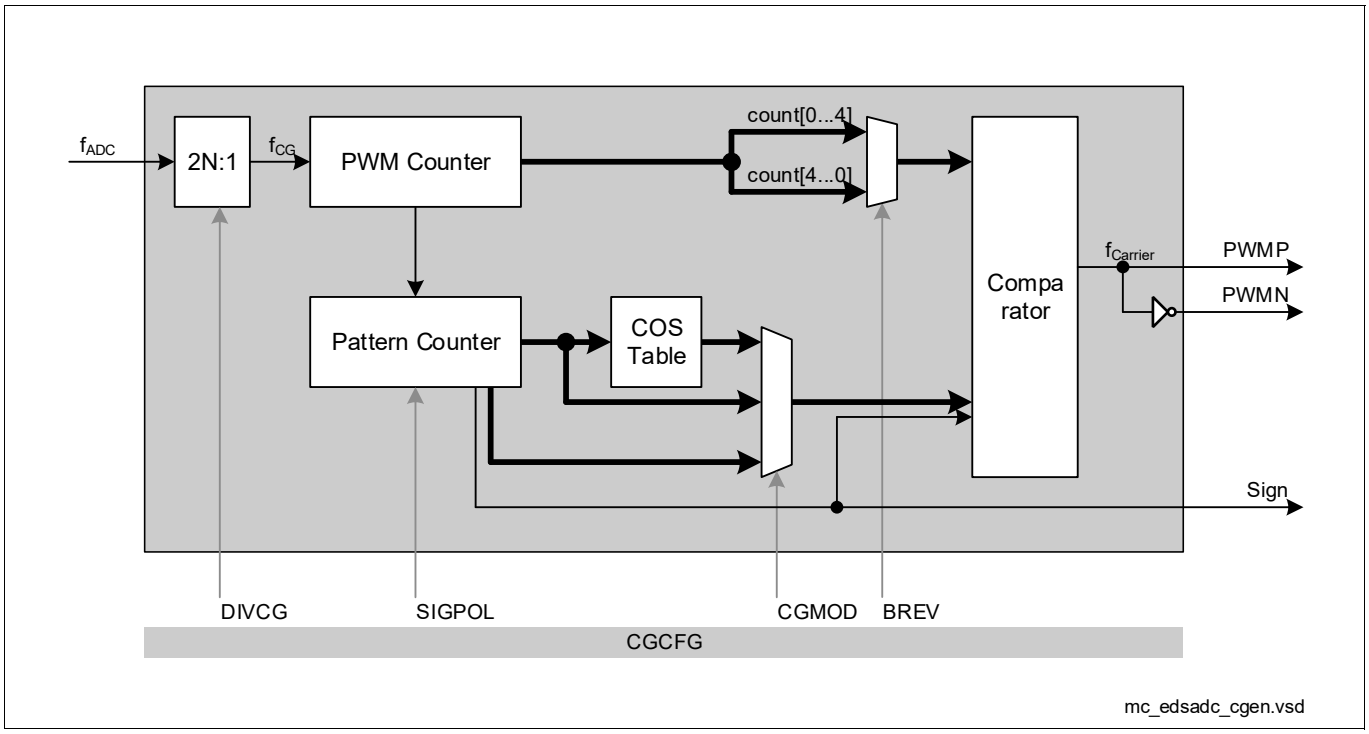


Figure 321 Carrier Generator Block Diagram

A carrier signal period consists of 32 steps. Each step equals a PWM period of 32 cycles.

Bit-reverse generation mode increases the frequency spectrum to yield a smoother induced sine signal. This is done by distributing the 0 and 1 bits over the 32 cycles of a PWM period.

The generated pattern is actually a cosine signal, i.e. it starts at the maximum output value. This is advantageous if the output pin is pulled high before the carrier signal is generated. In case of a pull-down the inverted output signal should be selected.

Note: All configurations become effective, when the carrier generator is started.

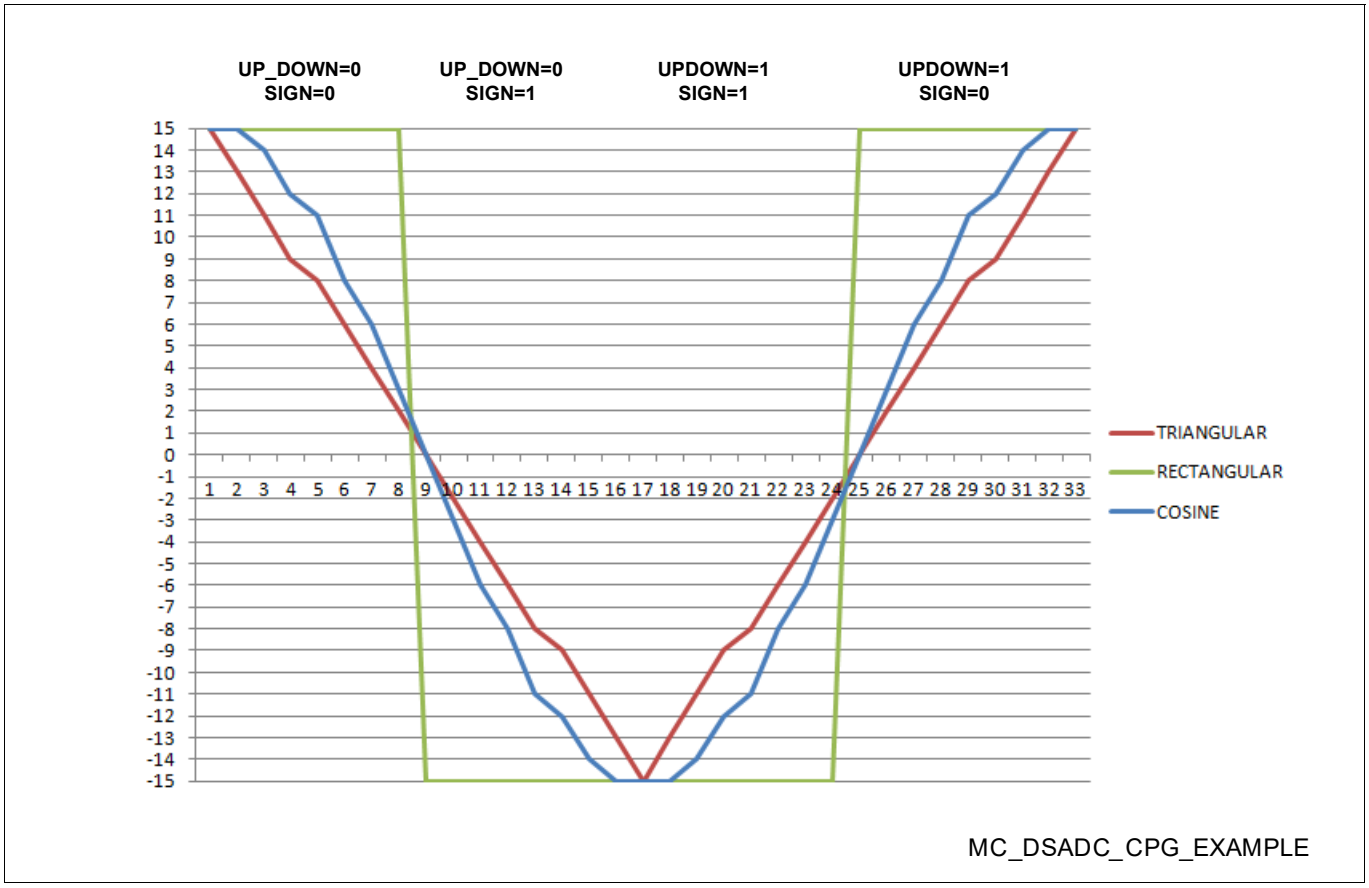


Figure 322 Example Pattern/Waveform Outputs

Carrier Generator Configuration Register

Note: The current position within a carrier signal period is indicated by bitfields STEPD, STEPS and STEPCOUNT.

CGCFG

Carrier Generator Configuration Register (00A0$_H$) Application Reset Value: 0710 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	SGNCG	STEPD	STEPS	0	STEPCOUNT			0			BITCOUNT				
r	rh	rh	rh	r	rh			r			rh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RUN	0							DIVCG				SIGPOL	BREV	CGMOD	
rh	r							rw				rw	rw	rw	

Field	Bits	Type	Description
CGMOD	1:0	rw	**Carrier Generator Operating Mode** Stopping the carrier generator (CGMOD = 00_B) terminates the PWM output after completion of the current period (indicated by bit RUN = 0). 00_B Stopped 01_B Square wave 10_B Triangle 11_B Sine wave
BREV	2	rw	**Bit-Reverse PWM Generation** 0_B Normal mode 1_B Bit-reverse mode
SIGPOL	3	rw	**Signal Polarity** 0_B Normal: carrier signal begins with +1 1_B Inverted: carrier signal begins with -1
DIVCG	7:4	rw	**Divider Factor for the PWM Pattern Signal Generator** Defines the input frequency of the carrier signal generator, derived from the selected internal clock source: $f_{CG} = f_{ADC}$ / CGP. *Note: The frequency of the carrier signal itself is f_{CG} / 1024.* 0_H CGP = 2 … F_H CGP = 32
RUN	15	rh	**Run Indicator** 0_B Stopped (cleared at the end of a period) 1_B Running
BITCOUNT	20:16	rh	**Bit Counter** Counts the 32 cycles generated for each step
STEPCOUNT	26:24	rh	**Step Counter** Counts the 8 steps generated for each quadrant of the carrier signal period
STEPS	28	rh	**Step Counter Sign** Indicates the sign of the step counter value 0_B Step counter value is positive 1_B Step counter value is negative
STEPD	29	rh	**Step Counter Direction** 0_B Step counter is counting down 1_B Step counter is counting up
SGNCG	30	rh	**Sign Signal from Carrier Generator** 0_B Positive values 1_B Negative values
0	14:8, 23:21, 27, 31	r	**Reserved, write 0, read as 0**

33.12.3 Return Signal Synchronization

In a resolver, the received return signals are induced by the carrier signal and their amplitudes are modulated with the sine and cosine magnitudes corresponding to the current resolver position. These amplitudes are determined by integrating the return signals over a carrier signal period.

To properly integrate their magnitude, the return signals must be rectified. For this purpose the carrier generator provides the sign information of the generated carrier signal (SGNCG in register **CGCFG**).

Alternatively, an external carrier signal generator can be used. If this generator delivers a sign signal, this can be input to a pin and is then used as external carrier sign signal. If no sign signal is available, the carrier signal itself can be converted by another input channel (SSCH in register **RECTCFGx (x=0-13)**) and its sign signal is then used as alternate carrier sign signal.

The rectification of the received signals must be delayed to compensate the round trip delay through the system (driver, resolver coils, cables, etc.). For the rectification, the received values are multiplied with the delayed carrier sign signal (SGND in register **RECTCFGx (x=0-13)**). This synchronization is done for each channel separately, to achieve the maximum possible amplitudes for each signal.

Note: The rectification unit is part of the integrator stage. Therefore, it is only active while the integrator is active.

The delay is realized with the sign delay counter SDCOUNT. SDCOUNT is cleared and started upon a falling edge of the carrier's sign signal (SGNCS), i.e. at the begin of the positive halfwave of the carrier signal. After counting SDPOS results from the filter chain, also the rectification signal (SGND) is cleared, indicating positive values from now on. After counting SDNEG values, the rectification signal is set, indicating negative values (see also **Figure 323**).

The compare values SDPOS and SDNEG are stored by the application software. SDPOS is the delay value that accounts for the resolver signal's round trip delay. This delay is constantly measured by capturing the current counter value into bitfield SDCAP when the first positive result (after negative results) is received in the respective channel. Software can read these value and compute a delay value e.g. by averaging a series of measured values to compensate noise. The delay for the negative halfwave (SGND = 0) is determined by adding the duration of a carrier signal halfwave. This value is written to bitfield SDNEG.

A new captured value is indicated by setting the flag SDCVAL. This flag is cleared when reading register CGSYNCx.

Capturing a new value can trigger a service request. The alternate service request line is used for this purpose. This alternate request source is selected by bitfield SRGA in register **FCFGMx (x=0-13)**.

Note: When the filter chain is initialized, bitfields SGND, SDCVAL, SDCAP and SDCOUNT are cleared.

Carrier Generator Synchronization Reg. x

CGSYNCx (x=0-13)

Carrier Generator Synchronization Reg. x (01A0$_H$+x*100$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SDNEG								SDPOS							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDCAP								SDCOUNT							
rh								rh							

Field	Bits	Type	Description
SDCOUNT	7:0	rh	**Sign Delay Counter** Counts the result values from the filter chain to delay the carrier sign signal
SDCAP	15:8	rh	**Sign Delay Capture Value** Indicates the result values counted between the begin of the positive halfwave of the carrier signal and the first received positive value.
SDPOS	23:16	rw	**Sign Delay Value for Positive Halfwave** Defines the content of SDCOUNT to generate a negative delayed sign signal (SGND).
SDNEG	31:24	rw	**Sign Delay Value for Negative Halfwave** Defines the content of SDCOUNT to generate a positive delayed sign signal (SGND).

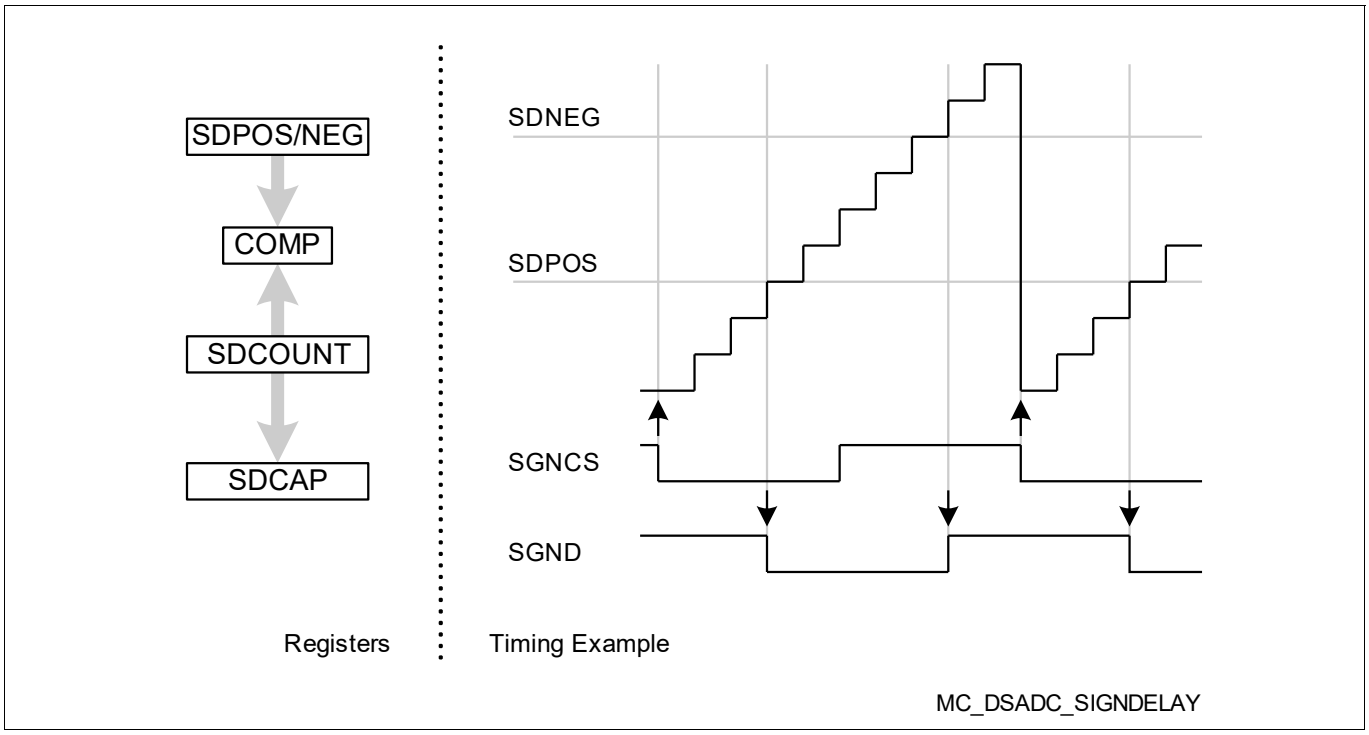


Figure 323 Sign Delay Example

Note: Whenever a new result value becomes available from the filter chain, the rectification counter is updated and the rectified value is forwarded to the integrator.

Rectification Configuration Register x

RECTCFGx (x=0-13)

Rectification Configuration Register x **($01A8_H$+x*100_H)** **Application Reset Value: 8000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SGND	SGNCS	0													
rh	rh	r													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDCVAL	0			SSCH				0		SSRC		0			RFEN
rh	r			rw				r		rw		r			rw

Field	Bits	Type	Description
RFEN	0	rw	**Rectification Enable** General control of the rectifier circuit. *Note: Rectification is only active while the integrator is active.* 0_B No rectification, data not altered 1_B Data are rectified according to SGND
SSRC	5:4	rw	**Sign Source** Selects the sign signal that is to be delayed. 00_B On-chip carrier generator 01_B Sign of result of channel selected by bitfield SSCH 10_B External sign signal A 11_B External sign signal B
SSCH	11:8	rw	**Sign Source Channel** Selects the channel providing the sign signal if SSRC = 01_B. Other products of the family may have less channels and, consequently, less valid SSCH codes. Not listed combinations are reserved. 0_H Sign result from channel 0 … D_H Sign result from channel 13
SDCVAL	15	rh	**Valid Flag** Indicates a new value in bitfield SDCAP. 0_B No new result available 1_B Bitfield SDCAP has been updated with a new captured value and has not yet been read
SGNCS	30	rh	**Selected Carrier Sign Signal** 0_B Positive values 1_B Negative values
SGND	31	rh	**Sign Signal Delayed** 0_B Positive values 1_B Negative values

Field	Bits	Type	Description
0	3:1, 7:6, 14:12, 29:16	r	**Reserved, write 0, read as 0**

33.13 Application Considerations

The operation of the EDSADC and, hence, its behavior is programmable in a wide range. This makes it suitable for different applications while requiring a certain amount of initialization and/or handling during operation.

As far as possible the configuration options can be handled by initialization. Several functions, for example power control or calibration, have been automated so they require an absolute minimum of handling during operation.

Additional hints in section **"Changing the Configuration" on Page 7**.

33.13.1 Clock Synchronization

To eliminate the interference of concurrently operating ADC channels, the converters can operate in a synchronized way so each of them can reach its optimum performance. The Phase Synchronizer distributes a clock control signal which is used by the converters to generate a local clock signal. See **Section 33.4.1**.

After reset, the clock synchronization is active. Setting the Phase Synchronizer is mandatory for the EDSADC to deliver its documented performance.

33.13.2 Calibration Recommendation

The actual calibration algorithm is executed automatically by hardware, so only a few configurations need to be selected before starting the algorithm.

The automatic calibration algorithm is executed for gain factors of 1:1 and 1:2. If a factor of 1:4 is selected calibration is done for factor 1:2.

The optimum mode for offset calibration depends on the properties of the input signal. The high-pass filter can automatically remove the offset level of a differential input signal. For irregular input signals or for single-ended operation the filter must be disabled to receive an undisturbed signal.

Note: The calibration algorithm compensates manufacturing tolerances and adjusts the channel to the selected decimation rate. It is, therefore, recommended to execute the algorithm at least once after a reset.

Define the intended full-scale value for the calibration in bitfield CALTARGET in register **GAINCALx (x=0-13)** (default value 25 000). Trigger the calibration algorithm by setting bit CALIB in register **FCFGMx (x=0-13)**. The completion and status of the calibration algorithm is indicated by bitfield CAL in register **FCNTCx (x=0-13)**.

*Note: Set bit AUTOCAL in register **FCFGMx (x=0-13)** to enable automatically triggered calibration sequences.*

33.13.3 Examples for Operation

The digital filters provide a number of configuration options to control their operation. The automatic calibration algorithm normalizes the overall gain factor of the filter chain to 1.000, independent of their configuration. The full-scale value of the result is adapted to <CALTARGET> (25 000 after reset), representing the reference voltage.

Different full-scale values can be achieved by choosing a different value for CALTARGET (see **"Handling of Overload, Overdrive and Overflow Conditions" on Page 26**).

The usable passband is a fraction of the configured output data rate.
For passbands in the range of 10 … 100 kHz this fraction is 1/3rd,
for passbands in the range of 4 … <10 kHz this fraction is 1/6th.

The output data rate is determined by the modulator frequency divided by the total oversampling rate.

Example:
CIC filter active with oversampling rate of 64, FIR0/FIR1 active (both 2:1), integrator inactive, f_{MOD} = 26.67 MHz:
Total oversampling rate is 64 × 2 × 2 = 256, output data rate is 26.67 MHz/ 256 = 104.2 kHz.

33.13.4 Supported Operating Ranges

The EDSADC can be configured for various operating modes, depending on the application's requirements. The focus can be on the input impedance, on the power consumption, or on the signal-noise-ratio SNR.

The configured oversampling rate (or decimation rate) OSR defines the SNR within a wide range. The reachable SNR is limited by thermal noise within the on-chip modulator. The relation between OSR and SNR is, therefore, no more linear in the upper area.

The table below lists some properties of these operating modes. Shown SNR values are valid for the analog supply voltage range (VDDM) between 4.5 V and 5.5 V. For analog supply voltages higher than 2.97 V and lower than 4.5 V, the SNR values degrades by 6 dB.

Note: To achieve the passbands given in this table, both FIR filters must be enabled.

Table 292 Properties of Operating Ranges

Modulator Frequency f_{MOD}	Input Current I_{RMS} [1]	$f_{PB} \le 10$ kHz [2]		$f_{PB} \le 30$ kHz [3]		$f_{PB} \le 50$ kHz [3]		$f_{PB} \le 100$ kHz [3]	
		SNR [4]	OSR	SNR [4]	OSR	SNR [4]	OSR	SNR [4]	OSR
16 MHz (f_{ADC} / 10)	6 µA	≥78 dB	≥267	≥78 dB	≥178	≥74 dB	≥107	≥65 dB [5]	≥54
20 MHz (f_{ADC} / 8)	7.5 µA	≥80 dB	≥334	≥78 dB	≥223	≥74 dB	≥134	≥68 dB [5]	≥67
26.67 MHz (f_{ADC} / 6) [6]	10 µA	**≥80 dB**	≥445	**≥80 dB**	≥297	**≥78 dB**	≥178	**≥74 dB**	≥89
40 MHz (f_{ADC} / 4)	15 µA	≥80 dB	≥667	≥80 dB	≥445	≥78 dB	≥267	≥74 dB	≥134

1) These typical values refer to an input voltage of 5 V and the typical value of the switched capacitor: $I_{RMS} = 5\,V \times f_{MOD} \times 2 \times C_{SW}$. I_{RMS} defines the equivalent input impedance.
2) Passband = f_d / 6
3) Passband = f_d / 3
4) The reachable signal-noise-ratio is limited by thermal noise within the modulator.
5) With this configuration, the signal-noise-ratio is limited by quantization noise.
6) These OSRs reference the Datasheet values.

Note: Related to hardware characteristic for passband frequency ≤ 10 kHz the FIR1 has to be used with a decimation rate of 1:1.

The equivalent input impedance, which is seen by the external sensor, depends on the input current I_{RMS}. The input current is proportional to the effective switched input capacitance, which itself depends on the selected gain factor.

Table 293 Equivalent Input Impedance

Modulator Frequency	Input Current for Gain 1 / 2 / 4	Impedance for Gain factor = 1	Impedance for Gain factor = 2	Impedance for Gain factor = 4
16 MHz	6 µA / 12 µA / 24 µA	R_{IN} = 833 kΩ	R_{IN} = 416 kΩ	R_{IN} = 208 kΩ
20 MHz	7.5 µA / 15 µA / 30 µA	R_{IN} = 666 kΩ	R_{IN} = 333 kΩ	R_{IN} = 166 kΩ
26.67 MHz	10 µA / 20 µA / 40 µA	R_{IN} = 500 kΩ	R_{IN} = 250 kΩ	R_{IN} = 125 kΩ
40 MHz	15 µA / 30 µA / 60 µA	R_{IN} = 333 kΩ	R_{IN} = 166 kΩ	R_{IN} = 83 kΩ

Quasi-Differential Input Mode

When operating the EDSADC in single-ended mode, the smaller input voltage range reduces the achievable SNR by 6 dB. Quasi-differential input mode is realized by connecting the unused input line to the common mode voltage instead of to reference ground. This centers the result values around zero.

Using gain factor 2 doubles the result value range (-full-scale for input=ground, +full-scale for input=reference). In this case the achievable SNR is reduced by only 3 dB.

33.13.5 Basic Initialization Sequence

After reset, the EDSADC is disabled to minimize the initial power consumption. By executing the following steps the EDSADC can be prepared for operation and be started:

Enable and configure the phase synchronizer according to the overall system requirements.

This is described in section “Application Considerations” in chapter CONVCTRL.

Enable the EDSADC module and prepare it for operation

```
EDSADC_CLC      = 0x00000000 ;Enable module EDSADC
EDSADC_GLOBCFG  = 0x00000000 ;Use default: auto level, sync. operation, min. dithering
EDSADC_MODCFG0  = 0x80028000 ;Mod. frequency = 160 MHz / 8 = 20 MHz,
                             ;enable differential input on channel 0-A (example)
EDSADC_DICFG0   = 0x84000000 ;Single-word read mode, no timestamp, no FIFO
EDSADC_GAINCTR0= 0x061B1193  ;Calibrate with OSR 512, calibration factor 1.0984
EDSADC_GAINCORR0=0x0011126E  ;Set shifter for operation, operation factor 1.1519
EDSADC_FCFGC0   = 0x00310031 ;Select a data rate of 100 kHz (= 20 MHz / 200);200 =
50[CIC] * 4[FIR]
EDSADC_GLOBRC   = 0x00010001 ;Enable modulator and filter chain of channel 0 (example)
WAIT                         ;Pause for wakeup time (approx. 20 µs)
                             ;(other operations can be executed in the meantime)
EDSADC_FCFGM0   = 0x90038003 ;Enable service request and FIR filters,
                             ;start calibration
```

For applications where two or more DSADC channels have to provide synchronous results, all related channels shall be enabled synchronously using a single write access to register GLOBRC. To handle the EDSADC channel specific modulator settling time, the following sequence is proposed:

- Enable all modulators of the application specific synchronization group by a single write access to the corresponding MxRUN bits in the upper half-word of the Global Run Control Register:
  - GLOBRC = XXXX °°°°H, where XXXXH depends on the number of implemented modulators;
- Wait for modulator settling time tMSET (see Data Sheet);
- Enable all modulators and corresponding digital filter chains of the application specific synchronization group by a single write access to the corresponding MxRUN and CHxRUN bits in the Global Run Control Register:
  - GLOBRC = XXXX XXXXH, where XXXXH depends on the number of implemented modulators/demodulator channel

Retrieve Conversion Results

After the calibration has finished, the channel will begin to convert the input signal.Conversions results are indicated by service requests and can be read from RESM0.

33.13.6 Module Handling in Sleep Mode

The EDSADC does not change its operating mode in sleep mode. While sleep mode is evaluated (CLC.EDIS = 0, default after reset), the module clocks are stopped upon a sleep mode request. To achieve the power reduction that is usually intended during sleep mode, the application needs to disable the EDSADC, or parts of it as required, before entering sleep mode.

Note: If any activity is intended during sleep mode make sure that sleep mode requests are disregarded (CLC.EDIS = 1) and make sure the phase synchronizer is not disabled in this case.

33.13.7 Overlap of CH9 and CH12

In some products of the TC3XX family the inputs of channels CH9B and CH12A are connected to the same pair of pins. Both channels can be used to convert input signals from these pins. To avoid coupling effect from one channel to the other, it is recommended to enable only one input path at a given time.

Note: While CH12A is operating, CH9A may operate on another input.

33.14 Summary of Registers and Locations

The EDSADC is built from a series of channels that are controlled in an identical way. This makes programming versatile and scalable. The corresponding registers, therefore, have an individual offset assigned (see **Table 294**). The exact register location is obtained by adding the respective register offset to the base address (see product-specific appendix) of the corresponding channel.

Due to the regular structure, several registers appear within each channel. This is indicated in the register overview table by formulas.

Registers with write access mode "...,M" can additionally be protected from unintended write access by setting the corresponding protection bit in register ACCPROT. Refer to **"Register Access Control" on Page 14** for more details and an association table.

Table 294 Register Overview - EDSADC (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	0000_H	U,SV	SV,E,P	Application Reset	9
ID	Module Identification Register	0008_H	U,SV	BE	PowerOn Reset	7
OCS	OCDS Control and Status Register	0028_H	U,SV	SV,P,OEN	See page 9	9
KRSTCLR	Kernel Reset Status Clear Register	$002C_H$	U,SV	SV,E,P	Application Reset	13
KRST1	Kernel Reset Register 1	0030_H	U,SV	SV,E,P	Application Reset	13
KRST0	Kernel Reset Register 0	0034_H	U,SV	SV,E,P	Application Reset	12
ACCEN0	Access Enable Register 0	$003C_H$	U,SV	SV,SE	Application Reset	11
GLOBCFG	Global Configuration Register	0080_H	U,SV	U,SV,P,M	Application Reset	16
GLOBRC	Global Run Control Register	0088_H	U,SV	U,SV,P,M	Application Reset	17
ACCPROT	Access Protection Register	0090_H	U,SV	SV,SE,P	Application Reset	14
CGCFG	Carrier Generator Configuration Register	$00A0_H$	U,SV	U,SV,P,M	Application Reset	88
EVFLAG	Event Flag Register	$00E0_H$	U,SV	U,SV,P,M	Application Reset	83
EVFLAGCLR	Event Flag Clear Register	$00E4_H$	U,SV	U,SV,P,M	Application Reset	84
MODCFGx	Modulator Configuration Register x	0100_H+x *100_H	U,SV	U,SV,P,M	Application Reset	27
DICFGx	Demodulator Input Config. Register x	0108_H+x *100_H	U,SV	U,SV,P,M	Application Reset	30

Table 294 Register Overview - EDSADC (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
FCFGMx	Filter Configuration Register x, Main	0110_H+x *100_H	U,SV	U,SV,P,M	Application Reset	63
FCFGCx	Filter Configuration Register x, CIC Filter	0114_H+x *100_H	U,SV	U,SV,P,M	Application Reset	65
FCNTCx	Filter Counter Register x, CIC Filter	0118_H+x *100_H	U,SV	BE	Application Reset	66
OVSCFGx	Overshoot Compensation Cfg. Register x	$011C_H$+x *100_H	U,SV	U,SV,P,M	Application Reset	49
IWCTRx	Integration Window Control Register x	0120_H+x *100_H	U,SV	U,SV,P,M	Application Reset	57
IIVALx	Intermediate Integration Value	0124_H+x *100_H	U,SV	BE	Application Reset	59
ISTATx	Integrator Status Register x	0128_H+x *100_H	U,SV	BE	Application Reset	59
RFCx	Result FIFO Control Register x	$012C_H$+x *100_H	U,SV	U,SV,P,M	Application Reset	74
RESMx	Result Register x Main	0130_H+x *100_H	U,SV	BE	Application Reset	75
OFFCOMPx	Offset Compensation Register x	0138_H+x *100_H	U,SV	U,SV,P,M	Application Reset	54
GAINCALx	Gain Calibration Register x	$013C_H$+x *100_H	U,SV	U,SV,P,M	Application Reset	38
GAINCTRx	Gain Control Register x	0140_H+x *100_H	U,SV	U,SV,P,M	Application Reset	38
GAINCORRx	Gain Correction Register x	0144_H+x *100_H	U,SV	U,SV,P,M	Application Reset	47
TSTMPx	Time-Stamp Register x	0150_H+x *100_H	U,SV	BE	Application Reset	69
TSCNTx	Time-Stamp Counter x	0154_H+x *100_H	U,SV	U,SV,P,M	Application Reset	70
FCFGAx	Auxiliary Filter Configuration Register x	0170_H+x *100_H	U,SV	U,SV,P,M	Application Reset	68
BOUNDSELx	Boundary Select Register x	0178_H+x *100_H	U,SV	U,SV,P,M	Application Reset	80
RESAx	Result Register x Auxiliary	0180_H+x *100_H	U,SV	BE	Application Reset	67
CGSYNCx	Carrier Generator Synchronization Reg. x	$01A0_H$+x *100_H	U,SV	U,SV,P,M	Application Reset	90

Table 294 Register Overview - EDSADC (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
RECTCFGx	Rectification Configuration Register x	$01A8_H$+x *100_H	U,SV	U,SV,P,M	Application Reset	92
VCMx	Common Mode Voltage Register x	$01B0_H$+x *100_H	U,SV	U,SV,P,M	Application Reset	34

33.15 Revision History

This is a summary of the modifications that have been applied to this chapter.

Table 295 Revision History

Reference	Change to Previous Version	Comment
V3.0.0		
Page 22	Describe 2:1 clock prescaler.	
Several	Replace f_{PER} with f_{ADC}.	
Page 11	Describe details of the module kernel reset.	
Page 21	Correct modulator selection in figure.	
Page 10	Add note about suspend mode to register OCS.	
Page 95	Add note to table “Properties of Operating Ranges”.	
Page 1, **67**	Correct data paths for auxiliary filter in figures.	
Page 61	Correct typo in formula for CIC3 group delay.	
V3.0.1		
Page 25	Changes in description and note for Slow Standby mode and Fast Standby mode.	
Page 97	Description for “WAIT” in Initialization Sequence changed from “Pause for extended wakeup time (approx. 28 us)” to “Pause for wakeup time (approx. 20 µs)”.	
Page 26	Wakeup Time changed.	
V3.0.2		
Page 91	Typo fixed (“ist” -> “is”).	
Page 95	Footnote added for 30 kHz, 50 kHz and 100 kHz.	
Page 95	Explanation for how to choose FIR1 decimation rate added.	
Page 95	Passband ≤ 10 kHz changed to < 10 kHz.	
Page 61	Headline “Group Delay” enhanced to “Group Delay and Settling Time”, additional information regarding settling time new.	
Page 61	Table “Settling Time Summary” new.	
Page 61	Enhanced example by settling time.	
Page 62	Added note regarding filter chain.	
V3.0.3		
Page 56	Updated section “Starting the Integration Window”.	
Page 13	Corrected Application Reset Value for register KRSTCLR.	
V3.0.4		
Page 55	Updated and replaced „Integrator Operation” figures.	
Page 57	Added 4. sub-item.	
Page 61	Changed values for CIC3 and FIR1.	
Page 95	Added two new sentences.	
V3.0.5		
Page 61	Updated formula.	

Table 295 Revision History (cont'd)

Reference	Change to Previous Version	Comment
Page 30	Updated result value ranges.	
V3.0.6		
Page 16, **Page 97**	Added description regarding GLOBRC register.	
Page 45	Formula updated to fit the example.	

34 Inter-Integrated Circuit (I2C)

This chapter describes the Inter-Integrated Circuit (short I2C) Module. The I2C module is not available in some variants. In these variants registers are still accessible but functionality cannot be guaranteed. The I2C module contains the following sections:

Note: The I2C module register names described in this chapter are referenced in the User's Manual by the module name prefix "I2Cm_" with m being the number of the module.

34.1 Feature List

- Compatible with I2C-bus specification version 2.1 **[1]**. See module functional restrictions in **Section 34.3.1.3**
- Master mode supported
- Multi-master mode supported (See restriction in **Section 34.3.1.3**)
- Slave mode supported
- Different speed ranges available for data transfer:
  - Standard mode up to 100 kbit/s (20kbit/s - 100kbit/s)
  - Fast mode up to 400 kbit/s (100kbit/s - 400kbit/s)
  - High-speed mode up to 3.4 Mbit/s (500kbit/s - 3.4Mbit/s)
- 7-bit and 10-bit I2C-bus addressing supported
- Automatic execution of low-level tasks like:
  - (De)Serialization of the bus data
  - Generation/detection of start and stop signal
  - Generation/detection of acknowledge signal
  - Bus state detection
  - Bus access arbitration in multi-master mode (See restriction in **Section 34.3.1.3**)
  - Recognition of device address in slave mode
  - Configurable detection of general call address
  - Configurable repeated start in master mode
- Flexible clock and timing control:
  - Prescaler for I2C kernel clock (from 0 to 255)
  - Bit rate generation via fractional divider
  - I2C-bus signal timing adjustment

- FIFO for buffering data from/to CPU with following features:
  - 8 FIFO stages based on 32 bit width
  - Configurable data alignment (byte, half word, word)
  - Configurable sizes for burst, transmit and receive package
  - FIFO usable as flow controller (seamless DMA flow)
- Advanced interrupt handling:
  - 4 data transfer interrupts (burst, last burst, single, last single)
  - Protocol interrupt with 7 sources (address match, general call, master code, arbitration lost, not-acknowledge received, transmission end, receive mode)
  - Error interrupt with 4 sources (FIFO transmit/receive overflow/underflow)
- Pretended Networking:
  - High-speed mode, fast mode and standard mode work with a minimum frequency of 5MHz in spb domain.

34.2 Overview

The I2C module communicates with the external world via a pair of I/O lines. A serial data line (SDA) and a serial clock line (SCL) carry the information between the devices. These lines are connected to a positive supply voltage via pull-up resistors. In quiescent state, when the bus is free, both lines are high. During communication the lines are alternatively pulled to low. The output stages of devices connected to the bus must have an open-drain (CMOS) or open-collector (bipolar) to perform a wired-AND function.

Figure 324 shows a block diagram of the I2C module.

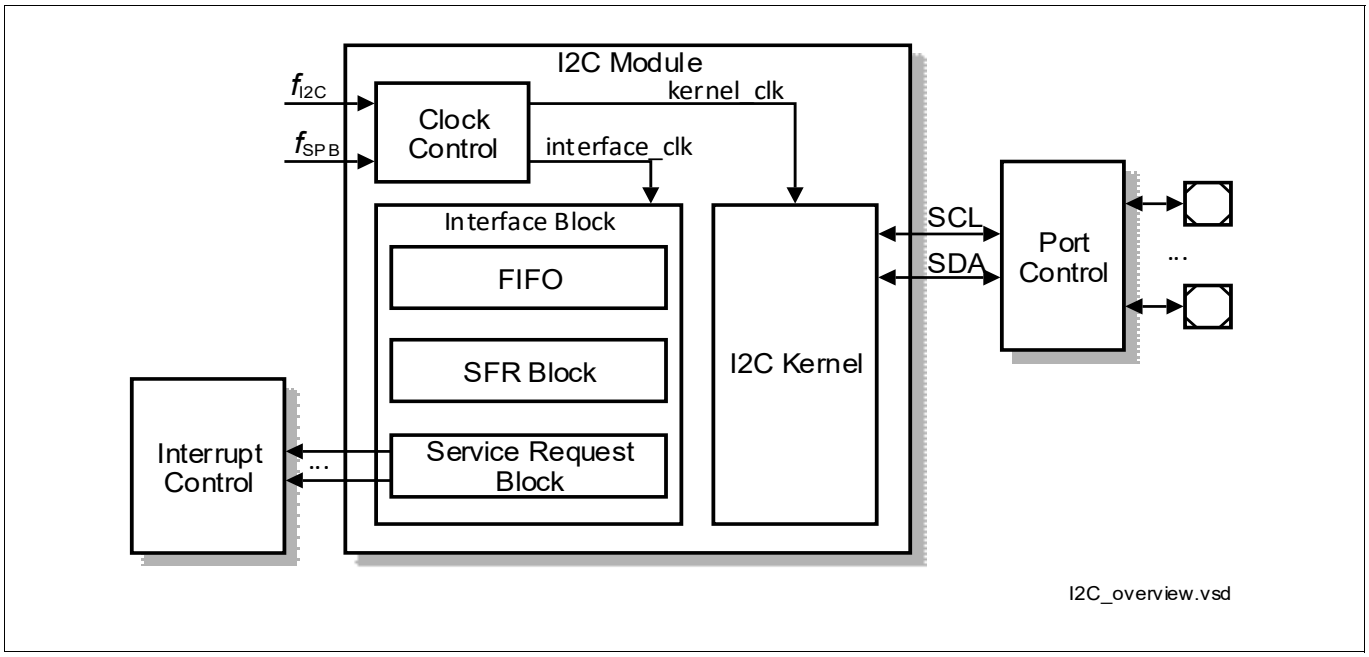


Figure 324 Block Diagram of the I2C Module

The I2C protocol was developed to provide a simple and efficient data transfer between multiple devices over a short distance. It uses a bidirectional serial bus with two wires.

The device can work as master or as slave. The master initiates the transfer, generates the clock pulses and terminates the transfer; it addresses a slave via a 7-bit or 10-bit address. Data can flow in either direction. In many applications there is only one master, typically a single-chip microcontroller, which communicates with several

slaves, e.g. general purpose peripherals or application specific circuits. But also multi-master systems with arbitration and collision detection are possible.

Data on the I2C-bus can be transferred at rates of up to 100 kbit/s in standard mode, up to 400 kbit/s in fast mode and up to 3.4 Mbit/s in high-speed mode. A slow slave may stretch the clock period. The number of devices connected to the same I2C-bus is limited only by a maximum bus capacitance of 400 pF.

The module relieves the CPU from many time-critical tasks.

The clock for the kernel of the I2C module is derived from the system clock via a prescaler. An additional fractional divider generates the desired bit rate. The exact timing of the I2C-bus signals can be adjusted.

A FIFO is used for data transfer between CPU and I2C module during transmission and reception. This allows writing and reading of multiple bytes. Data alignment and data sizes can be configured.

Six separate interrupt requests are available: for filling or emptying the FIFO, for reacting on certain protocol events and for handling of errors.

34.3 Functional Description

Description of the kernel, module implementation and integration.

34.3.1 I2C Kernel Description

Functional description of the I2C kernel.

34.3.1.1 I2C Protocol

Data is transmitted bit-by-bit on line SDA in conjunction with the clock on line SCL. To start communication, a master device generates a so called start condition. Subsequently data transfer starts. Therefore the data on the SDA line must be stable during the high period of the clock and may only change during SCL low phase. After all data have been transferred, the master closes transmission with a stop condition (see **Figure 325**).

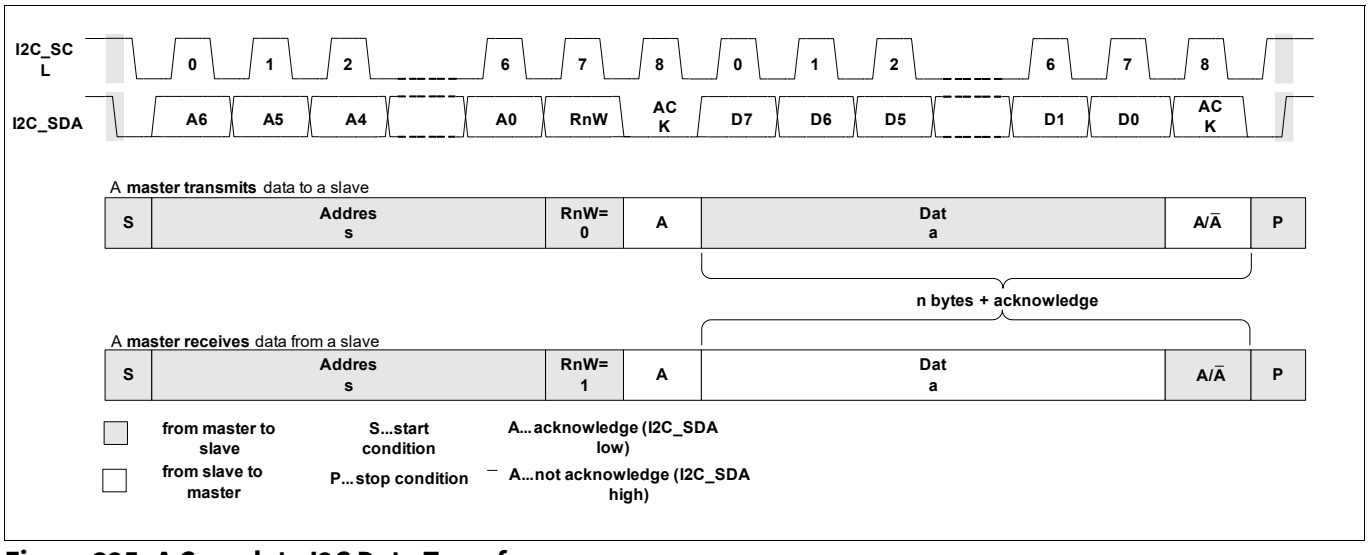


Figure 325 A Complete I2C Data Transfer

Start, Restart and Stop Conditions

A start condition is indicated by a high to low transition on line SDA while SCL remains high, a stop condition is defined by low to high transition under the same condition of high level on line SCL. The bus is considered to be busy after a start condition and to be free again after a stop condition. To allow continuous data transmission or reception without first generating a stop condition, a third condition has been introduced - the restart condition.

The bus stays busy if this condition is generated and the same or another slave can be addressed. Start and restart condition are functionally identical.

Address Transmission

After the start condition a slave address is sent. Normally the address is 7 bits long followed by an 8th bit that is a data direction bit (Read/not Write see **Figure 326**). A zero indicates a transmission (write operation); a one indicates a request to data (read operation). Following the address transmission the addressed device responds with an acknowledge (low on line SDA). If no acknowledge is returned, the master can then generate either a stop condition to abort the transfer or a repeated start condition to start another new transfer.

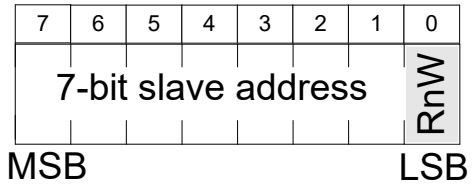


Figure 326 Address Byte Composition

Two groups of eight addresses are reserved for purposes, as shown in **Table 296**. For example the bit combination “11110XX” is reserved for 10-bit addressing mode.

Table 296 Reserved I2C-bus Addresses

Address	RnW	Description
0000000	0	General call address
0000000	1	Start byte - no device is allowed to acknowledge
0000001	x	CBUS address - I2C-bus devices may not respond
0000010	x	Reserved for different bus format
0000011	x	Reserved for future use
00001XX	x	High-speed mode master code
11111XX	x	Reserved for future use
11110XX	x	10-bit slave addressing mode

General Call

A general call is characterized by transmission of address “0000000” and RnW bit = 0. It is used for addressing every device connected to the I2C-bus. However, if a device doesn’t need any of the data supplied within the general call structure, it can ignore this address by not issuing an acknowledgement. Otherwise it acknowledges this general call and the following data bytes (if required) and behaves like a slave-receiver. The received data (e.g. programmable part of slave address, address of I2C-bus master) has to be handled by software. For further information the reader is referred to the I2C-bus specification version 2.1 **[1]**.

Data Transmission

Each byte transferred over the I2C-bus has to be 8 bits long whereby the number of bytes transmitted per transfer is unrestricted. Bytes are transferred MSB (Most Significant Bit) first. If the slave cannot receive or transmit a data byte until it has performed another function, it can hold the clock line SCL low (wired AND connection) to force the master into a wait state. Data transfer continues when the slave releases the clock line.

Acknowledge

Each transferred respectively received byte has to be followed by an acknowledge bit that is set by the recipient (master or slave depending on the transmission direction) onto the data line SDA. The acknowledge-related clock pulse is generated by the master. The device that has to set the acknowledge bit must pull down the SDA line during the acknowledge clock pulse so that it remains stable low during the high period of this clock pulse. Therefore in case of a slave the clock line may be held low until an acknowledge is released.

When the recipient of data doesn't acknowledge the data, for example because it was unable to receive the data or to transmit any data, the data line must be left high. A master-receiver that does not acknowledge the data from a transmitting slave device thereby tells the slave to stop transmission. The master then generates a stop condition to abort transfer or a repeated start condition to continue.

Clock Synchronization

To transfer messages a master generates its own clock on the SCL line. Data is only valid during the high period of the clock. Therefore data on line SDA has to be stable during this period and may only be changed during the low period of clock.

Two special cases must be considered. First, slave devices (or master devices operating as slaves) never generate clocks but are permitted to delay them. According to **Figure 327**, if device 1 as master device for example sets line SCL to low, device 2 as slave device may extend the low phase by setting its SCL output to low. A device such as a microcontroller with limited hardware for the I2C-bus can thereby slow down the bus clock and adapt the master to the internal operating rate of this device.

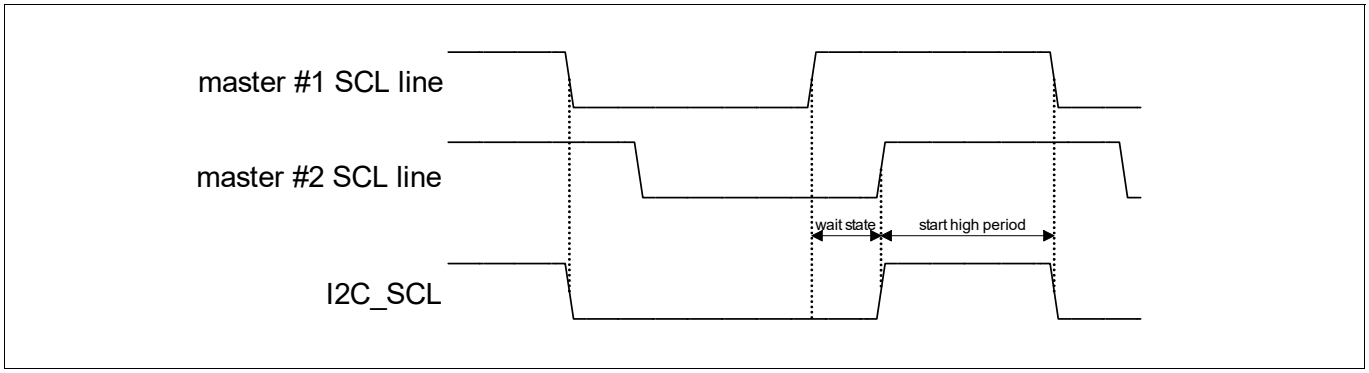


Figure 327 Clock Synchronization

Second, if several master devices access the I2C-bus at the same time, the SCL line is influenced by more than one device. As shown in the previous example, the SCL line will be held low by the device with the longest SCL low period. Devices with shorter periods enter a wait state during this time. In this way a synchronized SCL clock is generated.

Arbitration

To allow multi-master mode operation, bus arbitration is required. A master may start a transfer only if the bus is free. Two or more masters may generate a defined start condition simultaneously within a minimum hold time.

Arbitration takes place on the SDA line while SCL line is high, when one master transmits a high level while another master transmits a low level. Because of the wired AND functionality, the low level dominates the high level. As result the device that was trying to transmit a high level recognizes a different level on the SDA line and switches off its SDA and SCL output (see **Figure 328**).

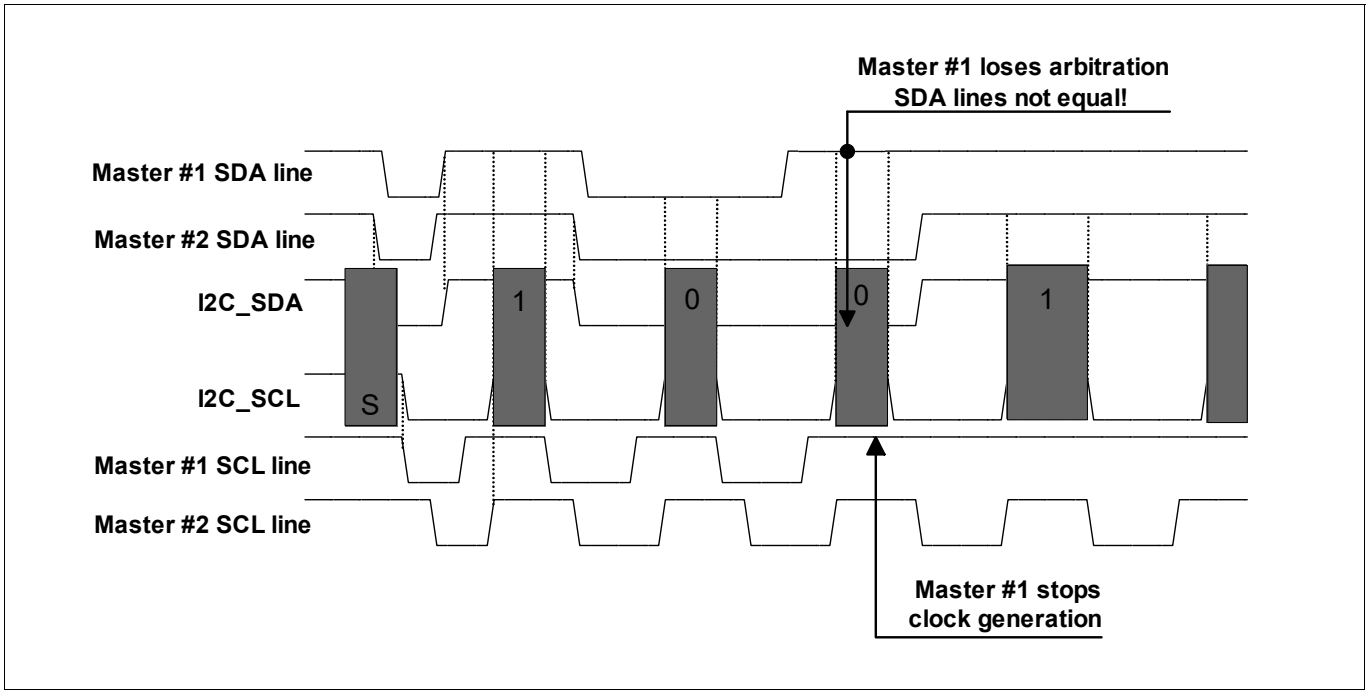


Figure 328 Example of an Arbitration Procedure

Arbitration may occur in different stages of transmission. A different level may potentially first appear during comparison of the address bits. If the masters are each trying to address the same device, arbitration continues with comparison of the RnW bit and subsequent data bits (if the masters are transmitting) or acknowledge bits (if the masters are receiving). Because information on the I2C-bus is determined by the winning master, no information is lost during the arbitration process. Since control of the I2C-bus is decided only by comparison of the data transmission stream over line SDA, there is no central master or any order of priority on the bus.

I2C-bus Formats

With the described rule types the formats shown in **Figure 329** are possible.

- Master-transmitter transmits to slave-receiver and stops after not-acknowledge.
- Master reads slave immediately after transmission of the RnW bit (high level) and the acknowledge from the slave, whereas the master becomes master-receiver and the slave becomes slave-transmitter.
- In combined format the just described regular sequences are handled, except for the repeated start condition instead of the stop condition between two sequences.

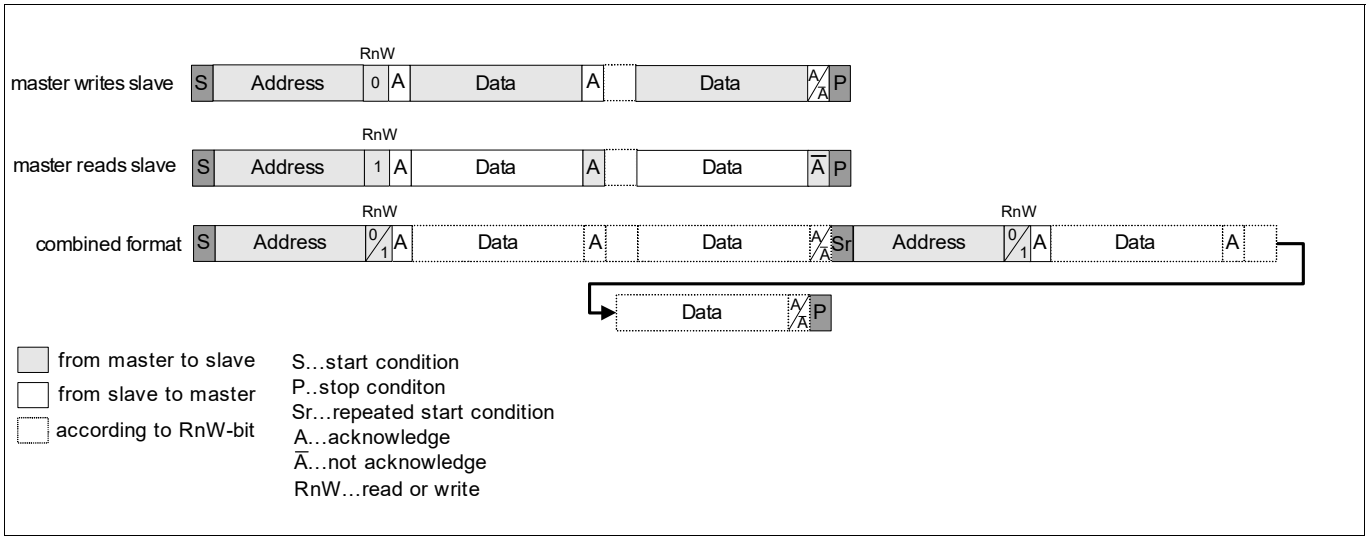


Figure 329 Types of I2C-bus Formats

10-bit Addressing Mode

The I2C address area is restricted to 112 applicable addresses with the 7-bit address mechanism described above. It turned out, that more combinations were required to prevent problems with the allocation of slave addresses for new devices. This problem was resolved with the 10-bit addressing scheme that is compatible with 7-bit addressing. Both modes may be used simultaneously. The following formats are possible:

To use 10 bits for addressing, the preamble address “11110XX” with RnW bit = 0 has to be used, including the two most significant bits “XX” of the 10-bit address, followed by an acknowledge from at least one 10-bit address matching slave and the second address byte containing the eight least significant bits. Only the (combined) 10-bit address matching slave now acknowledges. Data packages may now be transmitted to the addressed slave until stop condition or restart condition. If the master needs to read the slave, a combined format with a restart condition is necessary to

- First transmit the 10-bit address (first data package after preamble address contains 2nd address byte, RnW bit is 0) and (after the restart condition)
- Receive the requested data packages. Note that after the restart condition the preamble address “11110XX” has to be sent one more time, now with RnW bit = 1; the matching slave remembers that it was addressed before (see **Figure 330**). In case of a non matching preamble address, a not-acknowledge is returned.

The slave remains active as slave-transceiver until a stop condition or a restart condition occurs. According to the value of the following RnW bit a new slave may be addressed or the same slave is requested to transmit more data. Additionally the following combined formats (and mixtures of these formats) are allowed (**Figure 330**):

- A master transmits data to a slave and then reads back data from the same slave. The transfer direction is changed after the restart condition.
- A master transmits data to one slave and then transmits data to another slave.
- 10-bit and 7-bit addressing combined in one serial transfer. After restart condition the new addressing mode is selected by the address itself.

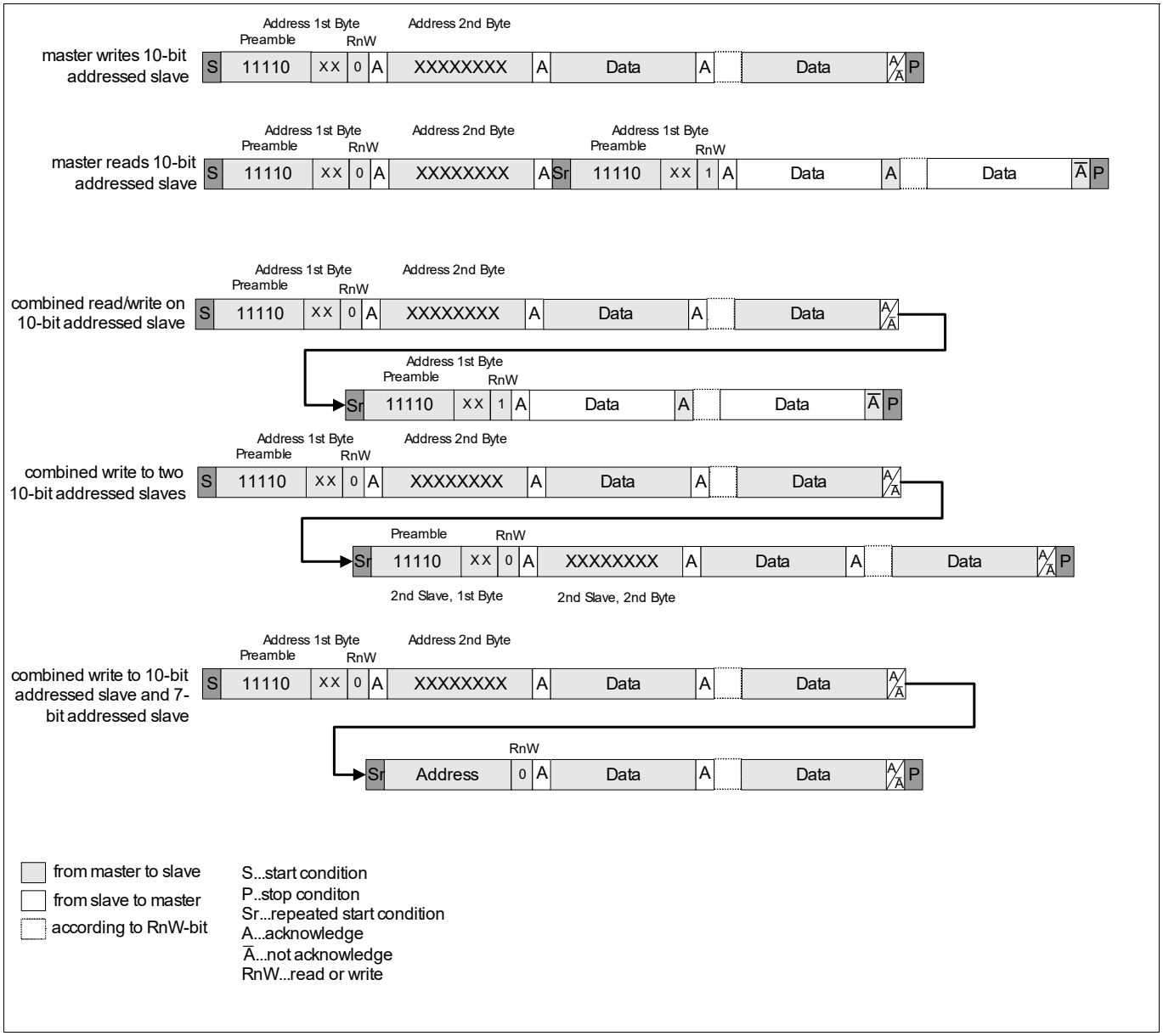


Figure 330 Types of I2C-bus 10-bit Address Formats

High-speed Mode

To support higher baud rates on the I2C-bus (up to 3.4 MBaud) the specification introduces the high-speed mode. To communicate at this higher baud rate the protocol uses a so called master code in place of the address byte to indicate that a master wants to change to the high-speed mode.

Special attention must be paid to meet the requirements of rise and fall times of the signal edges and other timing characteristics when working in high-speed mode. Also it is necessary to disconnect I2C-bus devices which are not able to follow the fast communication.

After the master sends this master code, no device is allowed to acknowledge it. The involved devices change to high-speed mode and the master starts the communication. The high-speed mode transmission behavior is the same as at lower baud rates with the exception that only restart conditions and stop conditions appear (no start condition). Since the master code determines a unique master device to control the bus, the arbitration process is finished after the master code sequence is put on the bus and must not be continued when working at the new communication speed.

Improvements to the regular I2C-bus specification for high-speed mode are given in section 13.1 of the I2C-bus specification version 2.1 **[1]**.

For high-speed mode there exists a separate configuration register **FDIVHIGHCFG** to program the baud rate; see also **Baudrate Generation for Master Mode**.

34.3.1.2 References

For more information, see the following documentation:

[1] I2C-bus specification version 2.1 standard (released January-2000)

34.3.1.3 Functional restrictions

There are some functional restrictions in this I2C module.

Multi-master mode - Master collision

The Multi-master mode is supported with a limitation/restriction in the following situation:

- The I2C module operates in master mode

- Another master wants to access/address the I2C module as slave

The resulting "master collision (problem)" cannot be handled correctly because the incoming address sent by the other master cannot be evaluated. So, if collision occurs second master has to restart access and retransmit the data.

Multi-master mode - Hold Time for the (repeated) Start condition

In a special situation this design step of the I2C module violates this timing specification by ~30% for Standard mode and by ~18,6% for Fast mode (min . 2.8 µs instead 4.0 µs for Standard mode and min. 0,488 µs instead 0,6 µs for Fast mode).

This occurs when data is written into TX FIFO, the I2C module is ready to start transmission an another master starts driving its startbit shortly before the I2C module starts driving. In this case the I2C Finite State Machine tries to win arbitration by reloading approximately half period in baudrate generator, so next SCL clock edge of the I2C module comes earlier than defined by t_7.

High-speed mode

The standard for the clock duty cycle ratio of the I2C high speed mode is 1:2. See **Chapter 34.3.1.4** in order to achieve the best duty cycle ratio for high speed mode. Note that not for every f_{I2C} frequencies 0% duty cycle ratio deviation can be achieved

34.3.1.4 Clock and Timing Control

The I2C module offers the following features to control clock and timing:

- Module clock control for integration logic clock and I2C kernel clock (see **Section 34.3.2.2**)
- Baudrate generation
- I2C signal timing adjustment

34.3.1.4.1 Baudrate Generation for Master Mode

A baudrate generation unit in the I2C kernel generates the SCL signal from the kernel_clk, controlled by settings of parameters **INC**, **DEC**, and **FS_SCL_LOW** in registers **FDIVCFG** (normal and fast mode), **FDIVHIGHCFG** (high speed mode), and **TIMCFG**.

The I2C standard has some special requirements for the clock, like

- asymmetric duty cycle
- slave or other masters are allowed to delay the clock by extending the low period. For this reason, a master must check, if any slave holds the clock line low after the master set the clock line to high. This implies that the state machine of a master needs at least 1 internal clock cycle delay between setting the clock to high and reading it back to check if it actually reached high level already. In addition more delay cycles can be introduced by the time constant of the pull up resistor and the line capacity. In any case, this protocol has the consequence that the actual baud rate will always be lower than the nominal baud rate.

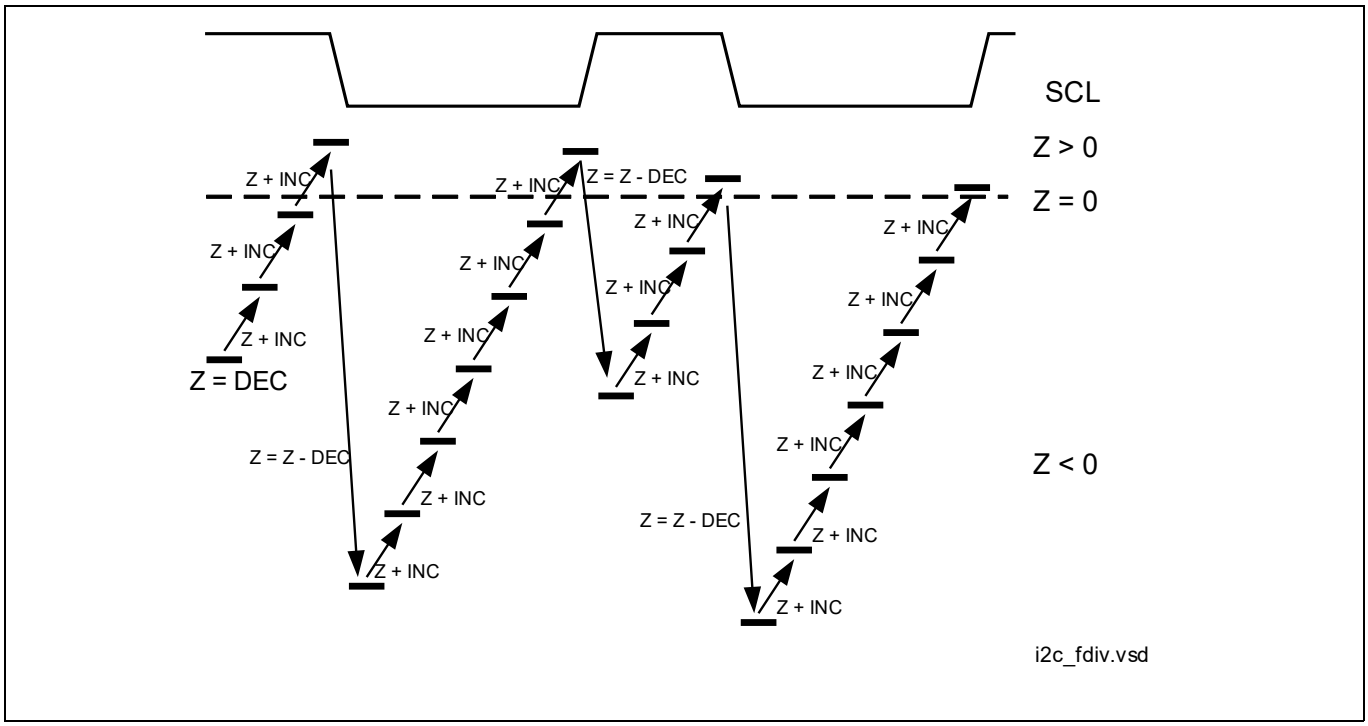


Figure 331 FDIV Principle - example for high speed mode

The operation principle of I2C fractional divider is shown in **Figure 331**. A counter Z is set to value **DEC** and is incremented using value **INC**. When counter reaches Z=0 it will be decremented using value **DEC**. With each decrementing operation SCL is toggled to achieve the required SCL waveform. Correction values are also applied to this mechanism but not shown above.

The formulas in the **Table 297** show how to calculate **DEC** and **INC** for a configured f_{SCL} frequency and a given f_{I2C}.

EN_SCL_LOW_LEN, **FS_SCL_LOW** and **SCL_LOW_LEN** are bit-fields in register **TIMCFG**.

Table 297 I2C Baudrate Generation Configuration for Master Mode

MODE	EN_SCL_LOW_LEN	FS_SCL_LOW	f_{SCL}	Low Time Correction	Duty Cycle
Standard	0	0	$fSCL = \frac{INC}{2DEC + 3INC} \times fI2C$ (34.1)	-	-
Fast	0	1		-	-
	1	X		1) $SCL\_LOW\_LEN = \frac{3INC}{2}$ (34.2)	-
High speed	X	X	$fSCL = \frac{INC}{5DEC + 2INC} \times fI2C$ (34.3)	-	1:2

1) This formula can be use to achieve a 50% duty cycle. **SCL_LOW_LEN** can be program with other values, but not greater than **DEC**.

In **Table 298** some example settings to program **INC**, **DEC** and LTC (bit-field **SCL_LOW_LEN**) are given and information about the duty cycle is shown.

Table 298 I2C INC/DEC Settings for Master Mode

I2C MODE	f_{I2C} [MHz]	INC	DEC	SCL_LOW_LEN	Duty Cycle [%]	f_{SCL} [MHz]
Standard	66.6	1	332	-	56.3	0.099
Fast	66.6	2	170	3	50	0.384
High speed	66.6	46	162	-	30.6	3.399

If a slave device is sensitive for violation of max frequency according I2C standard (eg. 3.4 MHz in High-speed mode) it is recommended to select a target SCL frequency with some margin to this limit.

34.3.1.4.2 Baudrate Generation for Slave Mode

Baudrate generation is mainly used for master mode, but there is one corner case where a support of the baudrate generation is required in slave mode: when I2C is in slave mode with a pending transmit data request and FIFO is currently empty then SCL is kept low until FIFO is filled after some time. Then transmit data is sent out immediately. SCL is kept low for a min. time of t_4.

In order to keep the set-up time,t_4 , according to I2C standard, the baudrate generation is used to guarantee the delay on SCL. The formulas in the 3rd and the 4th column of **Table** show how to configure **DEC** and **INC** for a given f_{I2C} and the intended set-up time. The formulas in column **Resulting t4 [µs]** define the real value of t_4 with the selected **DEC** and **INC** values.

The same value for **DEC** and **INC** as in master mode is not recommended and would lead to additional delay, as requested by the standard, in most cases.

Table 299 I2C Baudrate Generation Configuration for Slave Mode

MODE	min t_4[µs]	DEC / INC	Resulting t_4 [µs] 1)2)
Standard	0.25	$DEC = \frac{8}{9}((fI2C) \times (t_4) \times INC + INC)$ (34.4)	$C = INT\left(\frac{DEC + DEC(div)8}{INC} - 1\right)$ (34.5) $\frac{C}{fI2C} = t_4$ (34.6)
Fast	0.1	$DEC = \frac{4}{5}((fI2C) \times (t_4) \times INC + INC)$ (34.7)	3) $C = INT\left(\frac{DEC + DEC(div)4}{INC} - 1\right)$ (34.8) $\frac{C}{fI2C} = t_4$ (34.9)
High speed	0.01	$DEC = INC + 1$ (34.10)	$C = INT\left(\frac{DEC}{INC} + 1\right)$ (34.11) $\frac{C}{fI2C} = t_4$ (34.12)

1) The used abbreviation INT is the integer function.
2) The operator div denotes the integer division: a div b = greatest integer not greater than a/b.
3) SCL_LOW_LEN can not be greater than DEC.

In the following table some example settings are given to program **INC** / **DEC**.The used abbreviation INT is the integer function.

Table 300 I2C INC/DEC Settings for Slave Mode

I2C MODE	f_{I2C} [MHz]	INC	DEC	t_4 [ns]
Standard	66.6	20	314	250
Fast	66.6	25	160	105.1
High speed	66.6	250	251	30

The minimum kernel frequency is 8MHz for Standard and Fast Mode and 55MHz for High Speed Mode.

34.3.1.4.3 I2C Timing Adjustment

The I2C standard includes a number of requirements for the SCL and the SDA timing. These cannot always be fulfilled by the timing of the pads. Therefore, additional timing adjustment options are provided in the controller logic.

Various timings of signals can be shifted by multiples of kernel_clk cycles, as defined in register **TIMCFG**.

These settings offer a wide range of timing compensation. Not all of the values can be used and will result in a valid I2C timing, even clock or data hang-up is possible. Detailed analyses should be done - eg. during chip evaluation - taking actual kernel_clk and external timing into account.

SDA_DEL_HD_DAT

- Delays SDA output data starting from SCL falling edge for a defined number of kernel_clks.
- Must not exceed SCL low period to keep setup time for collision detection at next SCL rising edge.

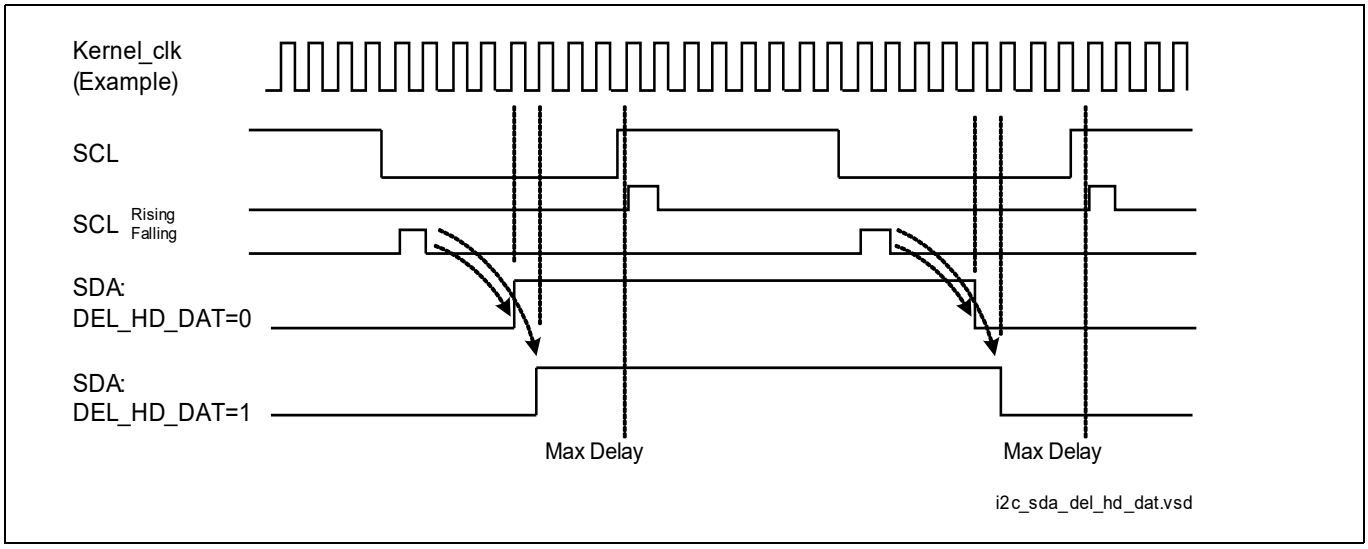


Figure 332 SDA_DEL_HD_DAT

High Speed Mode corrections

In **Figure 333**, the base period and delay corrections are shown for the entering and stop phases of HS mode.

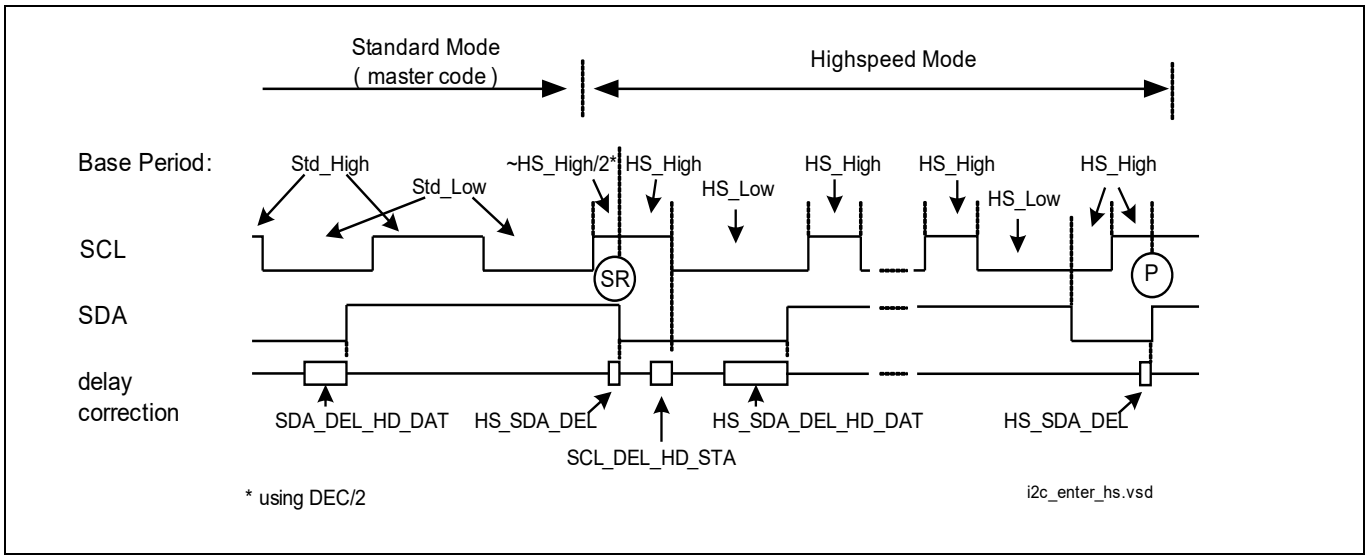


Figure 333 High Speed Mode corrections

34.3.1.5 I2C Kernel Control Logic

The I2C kernel can work in four different modes:

- Master-transmitter
- Master-receiver
- Slave-transmitter
- Slave-receiver

A state machine description can give a good and fast overview of the functionality. Also nearly all possible conditions can be rebuilt by stepping through the state machine and most of the error handling is covered by it. So this method is used to describe the I2C kernel.

Beginning with the top-level state machine, the starting up procedure is covered and together with the slave process and the master process sub-state machines, the active operations are comprehensible. These three state machines and their state transitions are described in the following.

As the I2C-bus working principle is packet orientated, the FIFO is typically configured as flow controller and so it issues the burst- and single-requests. This configuration is established by programming the bits **RXFC** and **TXFC** of the **FIFOCFG** register. Writing to the **TPS** bit-field of the **TPSCTRL** register starts the transfer.

Notes

1. *Peripheral is flow controller. As the I2C working principle is packet oriented, the FIFO is configured as flow controller and with this issues the burst- and single-requests. This configuration is established by programming the bit FC of the FIFOCFG register. Writing to the TPS-bitfield of the TPS_CTRL register starts the transfer.*
2. *Peripheral is not flow controller. In principle it is possible to use the FIFO in non flow controller mode. When the FIFO is not configured to operate as flow controller, the behavior of the FIFO operation has to be considered. The update of the FIFO fill level is only done when a stage is filled completely. In case of a (not protocol compliant) stopped transfer of the transmitter, the device is not able to evaluate the valid bytes since the FIFO issues interrupts/updates fill level only when a stage is filled completely. To prevent this behavior, the FIFO can be, for example, programmed to word alignment, so every received byte issues an interrupt/update. Anyway the software has to take care of such upcoming not protocol compliant data transmission stops, and has to handle this for example by programming an appropriate time-out.*

The Top-level State Machine

Figure 334 shows the top-level state machine. There are four states in the top-level state machine:

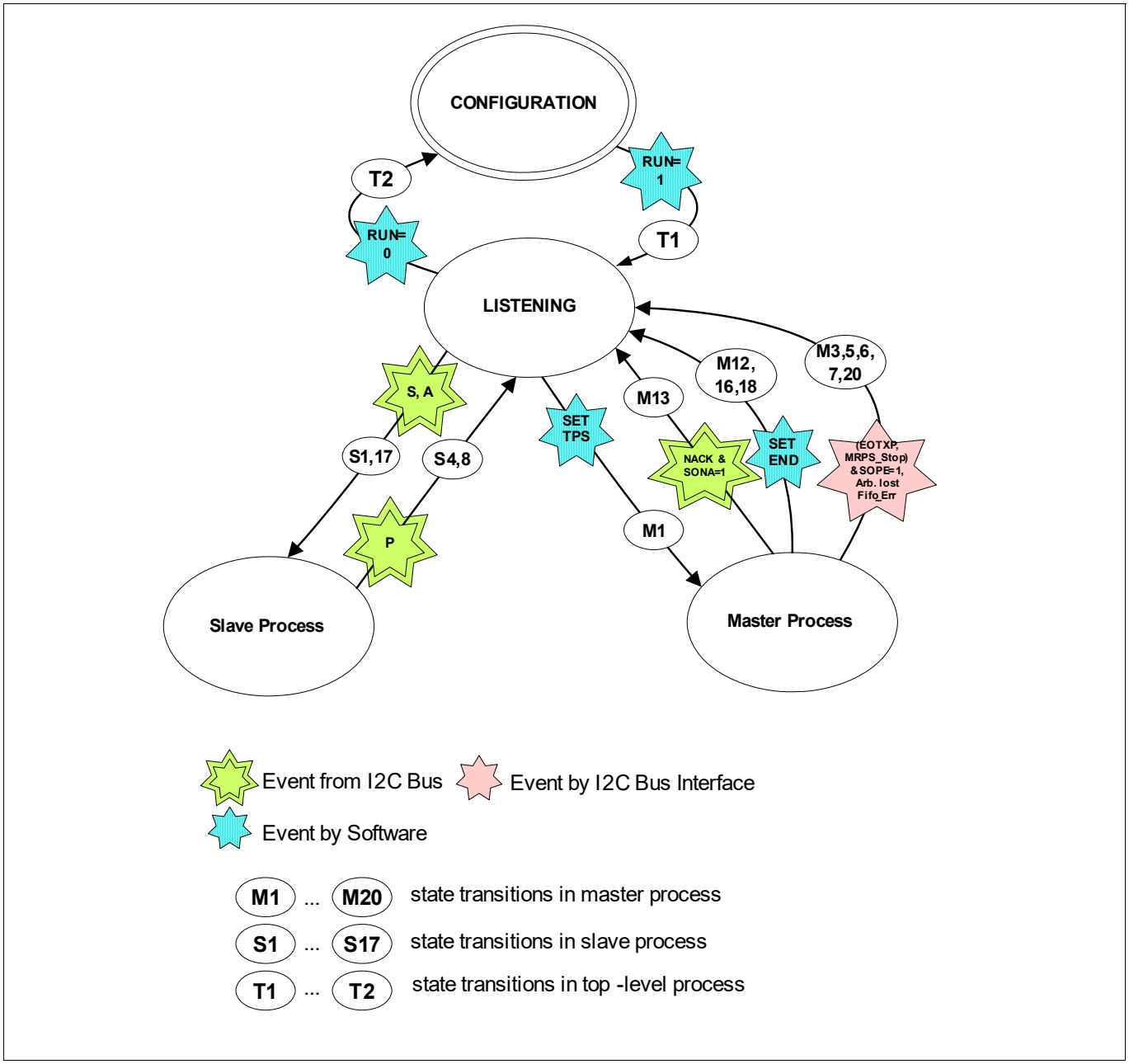


Figure 334 Top-level State Machine

CONFIGURATION: The peripheral is inactive; this is indicated by the **RUN** bit in **RUNCTRL** register, which is set to 0. In this state the peripheral should be configured by software by writing appropriate data words to the **ADDRCFG**, **FDIVCFG**, and **FIFOCFG** registers. This determines the following features:

- Peripheral is master or slave
- Address when accessed by another device
- Answering behavior to master calls and general calls
- Operating speed on the I2C-bus
- FIFO behavior

LISTENING: The peripheral has been activated by software. This state is entered in master and slave mode. The peripheral has to observe the bus for a certain time[1)] to ensure that there is no running transmission on the bus after this state has been entered. If no transitions on the I2C-bus are detected during this time period, the bus is supposed to be free. Otherwise the bit-field **BS** (bus status) in register **BUSSTAT** is set to 01_B (bus busy). Further functionality depends on the configuration: If the peripheral is a slave, only the slave process can be entered. If the device works as master, slave and master processes can be entered.

SLAVE PROCESS: A Start condition has been detected. The device is waiting for commands from the I2C-bus. This state represents the operation when acting as slave-transmitter or slave-receiver and is itemized in an own sub state machine.

MASTER PROCESS: The device is able to control the bus and work either as master-receiver or master-transmitter. Before the master can control the bus it has to set its clock frequency by programming its fractional divider with the appropriate value by writing it to the **FDIVCFG** register (when the I2C-bus is also operating in high-speed mode the appropriate values for INC and DEC have to be written to). The master process state is itemized in an own sub state machine.

The transition between the three states on the top-level are:

T1. The **RUN** bit has been set to 1 by software. The peripheral is activated and ready to observe the I2C-bus or react on transmission start commands.

T2. The **RUN** bit has been set to 0 again. I.e., because of new configuration effort, the peripheral is switched back to CONFIGURATION mode.

IMPORTANT: When “turning off” the peripheral it must be assured that no I2C-bus control responsibility is owned by the interface! Therefore the shut off process has to change the peripheral into a “secure” state and free the I2C-bus if necessary (generate a stop condition if in master mode). There are signals provided by the interface block which can be used to guarantee a secure shut off when turning off the I2C kernel.

1) One period of 1/100 kHz should be sufficient.

Slave Process Sub-state Machine

Figure 335 shows the slave process sub-state machine. There are 5 plus 1 (the listening state belongs to master and slave) different states in this sub-state machine:

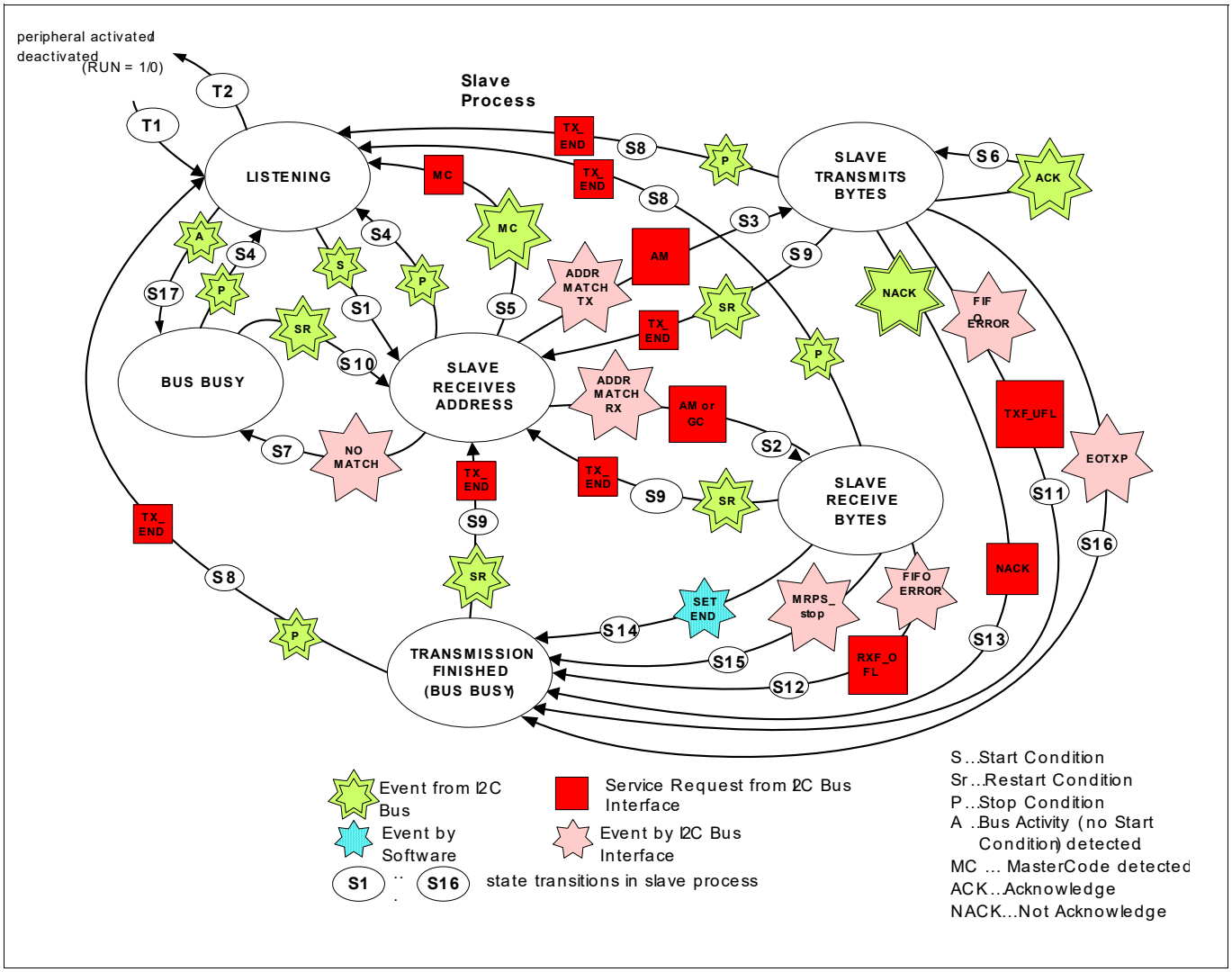


Figure 335 Slave Process Sub-state Machine

LISTENING: This state has been previously described (**Figure 334**). If the device is configured as slave, the module is listening only to the bus activity.

SLAVE RECEIVES ADDRESS: The module is shifting in the serial bits from the I2C-bus and acknowledges if the received byte(s) (first one in 7-bit address mode, first and second one in 10 bit address mode) match the adjusted address. During address matching the slave-receiver has the possibility to slow down the master clock by holding the SCL line low after each bit, since this can be used as a bit-by-bit handshake procedure.

Note: The address byte(s) is (are) not shifted into the FIFO.

SLAVE RECEIVES BYTES: The module is shifting in the serial bits from the I2C-bus and acknowledges when the whole data byte has been transported to the FIFO. The slave-receiver has the possibility to slow down the master clock by holding the SCL line low after each bit since this can be used as a bit-by-bit handshake procedure.

SLAVE TRANSMITS BYTES: The slave is allowed to hold down the SCL line when preparing the data for transmission. When valid data is available the slave releases the SCL line and waits for the clock cycles from the

master. The interrupt routine which proceeds when entering this state may write to the **TPSCTRL** register to initiate a transmission. This is only allowed if the bit-field **BS** (bus status) in register **BUSSTAT** in combination with the **RnW** clearly identifies the slave-transmitter mode of the peripheral. Otherwise writing to this register has no effect. First the I2C kernel gets a byte from the FIFO and puts it into a shift register. The module is then putting single data bits onto the I2C-bus and releases the SDA line after the 8th bit is sent.

*Note: Since the data flow control responsibility is task of the master device the slave should not stop transmitting bytes until the master tells it to do so. Therefore the **TPS** value in register **TPSCTRL** should be large enough to not run out of data bytes. In case of running out of data, the TRANSMISSION FINISHED state is reentered. The I2C data line is released to high level. The master continues receiving wrong values (0xFF).*

BUS BUSY: The bus is busy and the module is not involved in any data transfer. This is also the entry state when the bus is not free when starting up procedure is taking place (see state S17).

TRANSMISSION FINISHED (BUS BUSY): The module was involved in any data transfer. In distinction to the BUS BUSY state, the slave is still addressed by a master. Every time this state is left, caused due to a stop/restart condition, a TX_END request is generated as indication that the previous transmission has been closed.

The transitions which are connecting the several states are in detail:

S1: The module is detecting a start condition on the bus. It has to set the bit-field **BS** (bus status) in register **BUSSTAT** to 01_B. When this situation is occurring the I2C kernel is initialized since this indicates a new data transmission.

S2: The address which has been received matches the one stored in the address field in the **ADDRCFG** register. The **RnW** bit (8th bit in the first received byte) was set to 0. This means that the master wants to write to the slave. The corresponding bit **RnW** in register **BUSSTAT** is set to 1, i.e. the device continues working as slave-receiver. Also an acknowledge has to be set on the I2C-bus in the appropriate place (9th clock cycle). The I2C kernel sets the bit-field **BS** (bus status) in register **BUSSTAT** to 11_B and generates an AM (address match) request.

If the general call (GC) address matching is enabled by setting bit **GCE** in the **ADDRCFG** register, the device has to react also when it receives a general call (0x00). The GC request is issued and also an acknowledge has to be set on the I2C-bus when the GC is detected.

S3: The address which has been received matches the one stored in the address field in the **ADDRCFG** register and the SDA line was at a high level when the **RnW** bit was sent. This means that the remote master wants to read from the slave. The corresponding bit **RnW** in the **BUSSTAT** register has to be set to 0 (write). Also an acknowledge has to be set on the I2C-bus in the appropriate place (9th clock cycle). The I2C kernel sets the bit-field **BS** (bus status) in register **BUSSTAT** to 11_B and generates an AM (address match) request. The device is now working as a slave-transmitter.

S4: The device detects a stop condition on the I2C-bus and switches back to SLAVE LISTENING state. No interrupt is generated after the device was not addressed. The bit-field **BS** (bus status) in register **BUSSTAT** is set to 00_B (bus free).

S5: If the device is configured via bit MCE (master code enable) in the **ADDRCFG** register to react on a master code, the device can change baud rate to high-speed (Hs) mode when a remote master sends a master code. No device is allowed to acknowledge this master call. The interrupt routine which is called (MC request issued), has to handle the clearing of the interrupt bit in register **PIRQSC**. The LISTENING state is reentered. From now on the device works as slave in high-speed mode until a stop condition occurs.

S6: The slave-transmitter has received an acknowledge from the master-receiver and continues transmission.

S7: After the reception of the address bytes has been finished and no address match has taken place, the device changes to BUS BUSY state.

S8: The device has detected a stop condition on the I2C-bus after/while it was addressed. A TX_END (transmission end) request is generated, the bus is free again respectively the bit-field BS (bus status) in register **BUSSTAT** has to be set to 00_B again. The slave is now in LISTENING state.

S9: The slave has received a restart condition (repeated start condition) after/while it was addressed. A TX_END (transmission end) request is generated, the state switches back to SLAVE RECEIVES ADDRESS. The slave is reset again to a new transmission pending state (bit-field BS in register **BUSSTAT** = 01_B.). When the TBAM is activated (configuration register) the slave has to remember that it was addressed before because this event may be followed by a read access of the master to this device. So the address procedure after reset can be shortened and the transition S3 occurs one byte earlier (master sends 10-bit address mode byte with RnW bit = 1 (read)).

S10: A running transmission is ended by a remote master and this device wants to continue without losing control of the I2C-bus. The slave was not in transmission and resets to receive address bytes again since the new data transmission is concerning all connected slaves.

S11: The peripheral has problems to read data bytes from the FIFO (underflow/not ready). This is indicated by generating a TXF_UFL (TX FIFO underflow) request. It also changes to TRANSMISSION FINISHED state.

*Note: The device is still addressed by a master (bit-field BS in register **BUSSTAT** = 11_B.) until a stop or restart condition occurs.*

S12: The peripheral has problems to write the received byte into the FIFO (overflow/not ready). This is indicated by generating a RXF_OFL (RX FIFO overflow) request. The slave-transmitter puts a not-acknowledge on the I2C-bus to indicate its situation to the controlling master. This byte is not valid and will be discarded.

*Note: The device is still addressed by a master (bit-field BS in register **BUSSTAT** = 11_B.) until a stop or restart condition occurs.*

S13: The remote master wishes to close the connection to this slave and sets a not-acknowledge on the I2C-bus. The device generates a NACK (not-acknowledge) request and changes to TRANSMISSION FINISHED state (bit-field BS in register **BUSSTAT** = 01_B.). The interrupt routine can clear the appropriate bit via the corresponding bit in the **PIRQSC** register.

S14: The slave wants to stop the transmission and sets the **SETEND** bit in register **ENDDCTRL**. It puts a not-acknowledge on the I2C-bus after the next received byte. With this the slave gets the possibility to cancel the transmission via software intervention.

S15: The value of received bytes is equal to bit-field MRPS (maximum receive packet size) of the FIFO register **MRPSCTRL**. The FIFO part of the interface generates the signal MRPS_stop to the I2C kernel, which indicates the end of the received packet capacity. The slave produces a not-acknowledge to the master and changes to TRANSMISSION FINISHED. For more details on FIFO MRPS operation see **Section 34.3.1.6.5**.

S16: The number of transmitted bytes is equal to the TPS bit-field of the FIFO register **TPSCTRL**. The FIFO controller generates the EOTXP (end of transmit packet) signal to the I2C kernel, which indicates the last byte to transmit. After the last byte the device changes to TRANSMISSION FINISHED. For more details on FIFO EOTXP operation see **Section 34.3.1.6.2**.

Note: The master is not informed about this situation. Since the data flow control responsibility is task of the master device, it may continue receiving wrong values (0xFF).

S17: The module detects bus activity; no start condition has been detected before. A transmission may be currently in progress. The device changes to BUS BUSY state. This transition is performed if a start condition has been missed because the device was not activated (CONFIGURATION state).

Master Process Sub-state Machine

Figure 336 shows the master process sub-state machine. There are four different states in this sub-state machine:

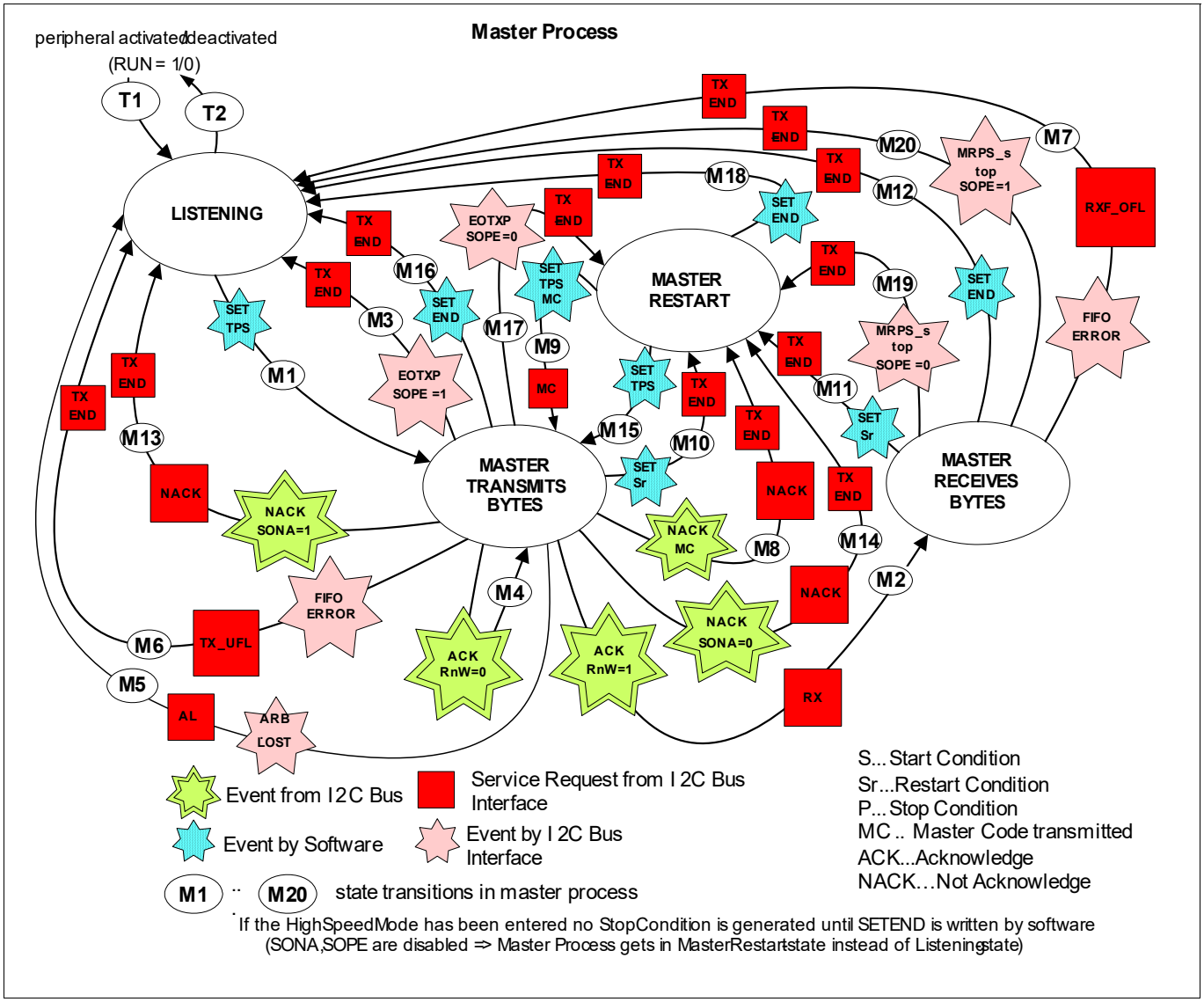


Figure 336 Master Process Sub-state machine

LISTENING: This state has been previously described (**Figure 334**). If the device is configured as master, it is listening to events either from software (writing to **TPSCTRL** register) or from the I2C-bus.

MASTER TRANSMITS BYTES: The device has started a transmission on the I2C-bus by software. The first byte after a start condition contains the address byte including the RnW bit. According to this bit (RnW bit in **BUSSTAT** register is set by hardware) the direction of the following transfer is determined. Bytes are transmitted on the I2C-bus until the FIFO is empty or a stop is initiated by software.

If bit MCE (master code enable) is set in the **ADDRCFG** register, the device has to check if the transmitted address byte is a master code.

MASTER RESTART: The device wants to restart a transmission without losing the control over the I2C-bus. The clock SCL stays low as long as no further communication is requested. The only event that should occur in this state is writing to the **TPSCTRL** register. Nevertheless the device can give up the I2C-bus control by writing to bit

SETEND in the **ENDDCTRL** register. Of course, when this situation occurs, the peripheral produces a stop condition on the I2C-bus.

If bit MCE (master code enable) is set in the **ADDRCFG** register, the device has to check if the transmitted address byte is a master code.

MASTER RECEIVES BYTES: The master has ordered bytes from the slave and handles the data transfer from the I2C-bus to the FIFO. This means that the bits are shifted into a buffer register in the I2C kernel and afterwards are passed to the FIFO. Every time a byte was transferred to the FIFO successfully the master-receiver produces an acknowledge in the according position.

The state transitions in the master process are:

M1: The software has written the number of bytes which have to be transmitted over the I2C-bus into the control register **TPSCTRL**. The FIFO generates appropriate data transfer requests which transfers valid data into the FIFO. A start condition is set on the bus and the bit-field BS (bus status) in register **BUSSTAT** is set to 10_B (busy master). The first (the first and the second) byte is (are) the 7-bit (10-bit) address of the slave which shall be accessed.

If the master wishes to initiate a transition to high-speed mode, the transition is only valid in combination with the MCE (master code enable) bit in the **ADDRCFG** register. The software has to write the master code into the FIFO, this means also the value of the **TPSCTRL** is predetermined with 1.

M2: The master receives an acknowledge from slave and the bit RnW in the **BUSSTAT** register was set to 1 during the addressing phase. A slave has been addressed successfully as transmitter, the state moves to MASTER RECEIVES BYTES. An RX (Receive) request is generated to distinguish between transmit- and receive-FIFO requests in non flow controller mode.

M3: The valid data packet from the FIFO was transferred successfully to the I2C-bus and the FIFO indicates this by the EOTXP (end of transmit packet) signal. If the RnW bit was set to 0 (writing to a slave) and bit SOPE (stop on packet end) in register **ADDRCFG** was set to 1, the transfer is closed by putting a stop condition on the bus. After this stop condition has been detected the peripheral indicates this successful transfer to the software via the TX_END (transmission end) request. The bus status bit-field is set to 0x0 (bus free).

M4: After the master transmits a byte it releases the SDA line to allow the addressed slave to pull down the SDA line (acknowledge). If this event occurs (and RnW in register **BUSSTAT** was set to 0), the master can continue to transfer bytes on the I2C-bus.

M5: The SDA line level is not equal to the one that is set by the master. This means that another master also tries to control the I2C-bus. This is indicated by the AL (arbitration lost) request and the device switches to the LISTENING state. The interrupt routine has to clear bit AL in the **PIRQSC** register. The Arbitration lost indication is not executable when the device works in high-speed mode (no arbitration procedure in high-speed mode since there is only one master remaining). The arbitration process must be operating at each bit which is sent by the master! Before interrupting the CPU, the bit-field BS (bus status) in register **BUSSTAT** is set to 01_B.

M6: If the master wants to transmit bytes but the FIFO is empty (underflow), it has to generate a TXF_UFL (TX FIFO underflow) request. The bit-field BS (bus status) in register **BUSSTAT** is reset to 00_B, and a TX_END (transmission end) request is generated after the master has put a stop condition on the bus.

M7: The I2C kernel wants to transfer bytes to the FIFO but without success. Then it has to produce an RXF_OFL (RX FIFO overflow) request. The current byte is not-acknowledged and a stop condition is put on the bus (bit-field BS in register **BUSSTAT** is set to 00_B and a TX_END (transmission end) request is generated). Requests (SREQ, BREQ, LSREQ or LBREQ) are generated for the outstanding data, the target is to empty the FIFO. (

M8: If master code has been transmitted by the device and a NACK (not-acknowledge) has been received, the MASTER RESTART state is entered. The clock SCL stays low as long as no further communication is requested. This is done by writing bit-field TPS in register **TPSCTRL** (see transition M9).

M9: A new communication in high-speed mode is started by writing bit-field TPS in register **TPSCTRL** when a master code has been previously transmitted (M8). The switch into high-speed mode (MC request) is done as soon

as clock and data are released to high (see **Figure 337**). State MASTER TRANSMITS BYTES is entered. During high-speed mode, all fast and standard mode devices must be disconnected from the bus. This can be done in the MC request.

M10: Bit SETRSC has been set during the transmission of the data (the software wants to change direction/slave and therefore writes to the **ENDDCTRL** register). The I2C kernel resets and changes to MASTER RESTART state and a TX_END (transmission end) request is generated. Therefore, the master does not lose the control of the I2C-bus after the transmission has ended. The master stops transmission after the current byte was acknowledged/not acknowledged. Valid data in the FIFO are discarded. In this situation the FIFO is flushed to produce an initial situation.

M11: If the master wants to change direction/slave immediately in this state, it has to write to bit SETRSC in the **ENDDCTRL** register. The master puts a not-acknowledge on the appropriate place during the current transmitted byte. This indicates the slave-transmitter to stop transmission. The master switches to MASTER RESTART state; a TX_END (transmission end) request is generated.

M12: By sending a not-acknowledge on the bus, the master indicates the slave an upcoming stop of the transmission. This is established by writing to bit SETEND in register **ENDDCTRL**. Stop condition and TX_END (transmission end) request are generated. Requests (SREQ, BREQ, LSREQ or LBREQ) are generated for the outstanding data, the target is to empty the FIFO. The LISTENING state is entered; the bit-field BS (bus status) in register **BUSSTAT** is set to 00_B.

M13: If the master is receiving a not-acknowledge and bit SONA (stop on not-acknowledge) in the **ADDRCFG** register is 1, it produces a stop condition, resets the bit-field BS (bus status) in register **BUSSTAT** and issues a NACK (not-acknowledge) request and a TX_END (transmission end) request. The FIFO is flushed; remaining data in the FIFO stages are discarded.

M14: If bit SONA (stop on not-acknowledge) in the **ADDRCFG** register is 0, the peripheral wants to remain the controlling master of the I2C-bus after receiving a not-acknowledge. The next state is MASTER RESTART. A NACK (not-acknowledge) request and a TX_END (transmission end) request indicate this event to the software. The FIFO transfers the valid data to the memory.

M15: The master decided to transfer data on the bus again and writes to the **TPSCTRL** register. The first byte is interpreted as address byte (the 8th bit indicates RnW). The appropriate data transfer requests are activated by the FIFO controller. After the SCL line is high again (slave can stretch the low period) the master is allowed to put a restart condition onto the bus and is again in the MASTER TRANSMITS BYTES state starting to transmit bytes again.

M16: The software has written the SETEND bit in the **ENDDCTRL** register. The peripheral stops transmission of bytes after the current byte has been written on the bus. The I2C kernel flushes the FIFO since the remaining data in the FIFO is invalid. A TX_END (transmission end) request is generated.

M17: The FIFO indicates the end of the data packet and the peripheral was configured to go into MASTER RESTART state after that event. Since the data transfer has ended as intended a TX_END (transmission end) request is produced.

M18: The master has no data to send anymore but still controls the I2C-bus. Therefore it can use the SETEND bit to give up the I2C-bus control by setting a stop condition. A TX_END (transmission end) request is generated.

M19: A value different to zero is programmed to bit-field MRPS (maximum receive packet size) of the **MRPSCTRL** register. If the number of received bytes is equal to the MRPS bit-field, the FIFO produces the MRPS_stop signal. When bit SOPE (stop on packet end) in the **ADDRCFG** register is set to 0, the master generates a not-acknowledge and changes to MASTER RESTART state, so the master does not lose the control over the I2C-bus. A TX_END (transmission end) request is generated.

M20: (See also M19.) The FIFO produces the MRPS_stop signal which indicates that the desired amount of received data bytes is reached. The peripheral generates a not-acknowledge and changes to LISTENING state because bit SOPE (stop on packet end) in the **ADDRCFG** register is set to 1. Of course, the interface produces a

stop condition; so the I2C-bus is free again (change also the bus status bit-field!). A TX_END (transmission end) request is also generated. Requests (SREQ, BREQ, LSREQ or LBREQ) are generated for the outstanding data, the target is to empty the FIFO. The LBREQ or the LSREQ indicate the completed reception of data. For more details on FIFO MRPS operation see **Section 34.3.1.6.5**.

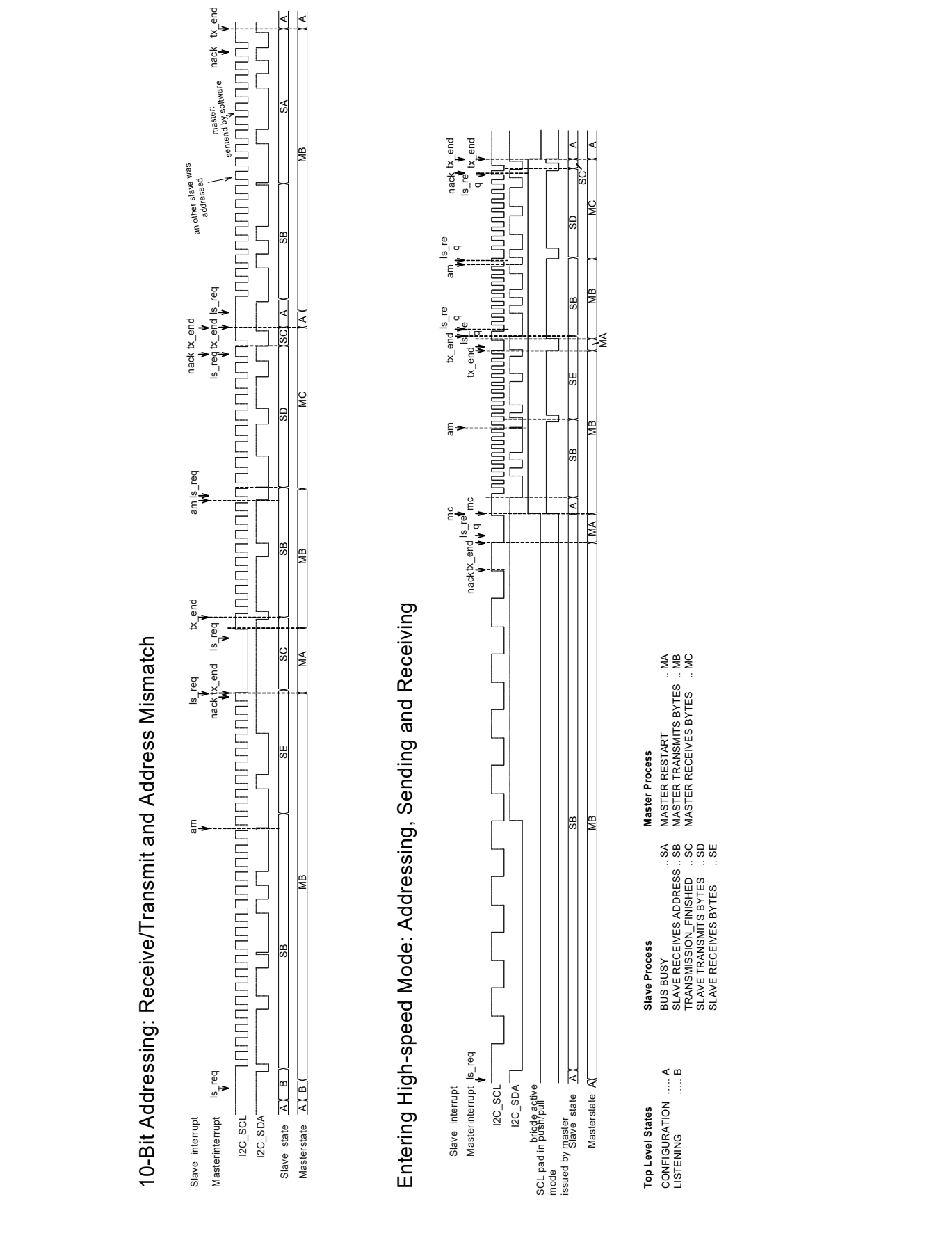


Figure 337 Connection Examples including Requests in Master and Slave Mode

34.3.1.6 FIFO Operation

The I2C module uses a FIFO between its kernel and the bus, in order to adapt the character processing speed of the peripheral to the transfer rate of the bus system. The FIFO has a transmission state and a reception state. **Figure 338** shows the bidirectional half duplex FIFO unit with the input and output lines and with the corresponding FIFO registers.

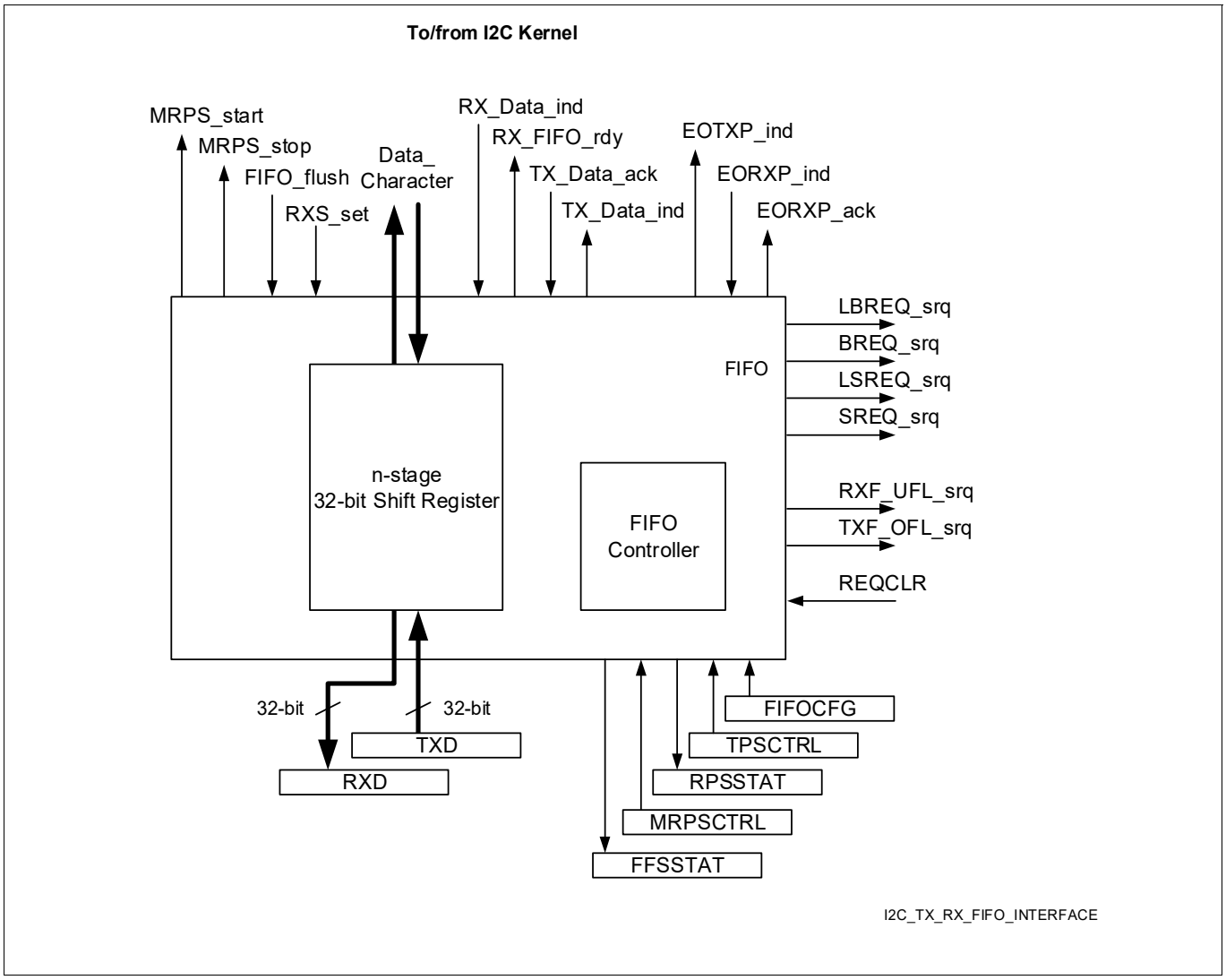


Figure 338 Bidirectional Half Duplex FIFO

Notes

1. *When the FIFO isn't serviced in time (transmission: filled with transmit data, reception: read the received data), an underflow/overflow event will lead to abortion of the transaction.To avoid this behaviour the software shall be configured to transfer a maximum of 32 bytes per transfer. If more than 32 bytes should be transmitted, the transmissions should be divided in maximum 32 data bytes per transfer*
2. *If bit-field **CRBC** in register **FIFOCFG** is disable, a linked list should be used to avoid that the CPU has to get active to clear the request. The DMA channel that serves I2C has to be configured with a linked list that first makes the FIFO-TX/RX transfer and second clears the interrupt by writing 0x0 to register ICR (Interrupt Clear Register) .*

34.3.1.6.1 Data Transmission

The software can move transmit data to the FIFO by writing to the Transmission Data Register **TXD**. Moving data to the FIFO can be done according to the data transfer request signals generated by the FIFO controller as described in **Section 34.3.1.6.2**. But with the register **FFSSTAT**, which indicates the number of full FIFO stages, the data transfer to the FIFO can be handled completely without data transfer requests. The data alignment within the FIFO is described in **Section 34.3.1.6.3**. The FIFO shifts data to the I2C kernel by means of the handshake lines tx_data_ind and tx_data_ack.

With the signal FIFO_flush the I2C kernel is able to clear the FIFO content. But if a data transfer request (see **Section 34.3.1.6.2**) is pending, then the FIFO is not cleared until the corresponding signal REQCLR has been cleared.

In bit-field **CRBC** in the register **FIFOCFG**can be configured how to clear the FIFO request.

The I2C-bus interface should know when it is receiving the last character of a data packet from the FIFO. Therefore the software should define a transmit packet size in bit-field TPS of register **TPSCTRL** and then the FIFO controller indicates the last character of the packet to the I2C kernel by setting the EOTXP (end of transmit packet) signal.

Since TPS is double buffered the software can write the size of the next packet into **TPS** as soon as the **TPS** value of the current packet has been loaded into the TPS counter, i.e. reading **TPS** returns 0. It loads the new TPS value as soon as ithas room in the FIFO.

Also the characters of the next packet can be moved to the FIFO immediately after the current packet characters as indicated in **Figure 339**. If byte or half word alignment is used as described in **Section 34.3.1.6.3**, then the characters of two different packets must not be aligned in a single stage.

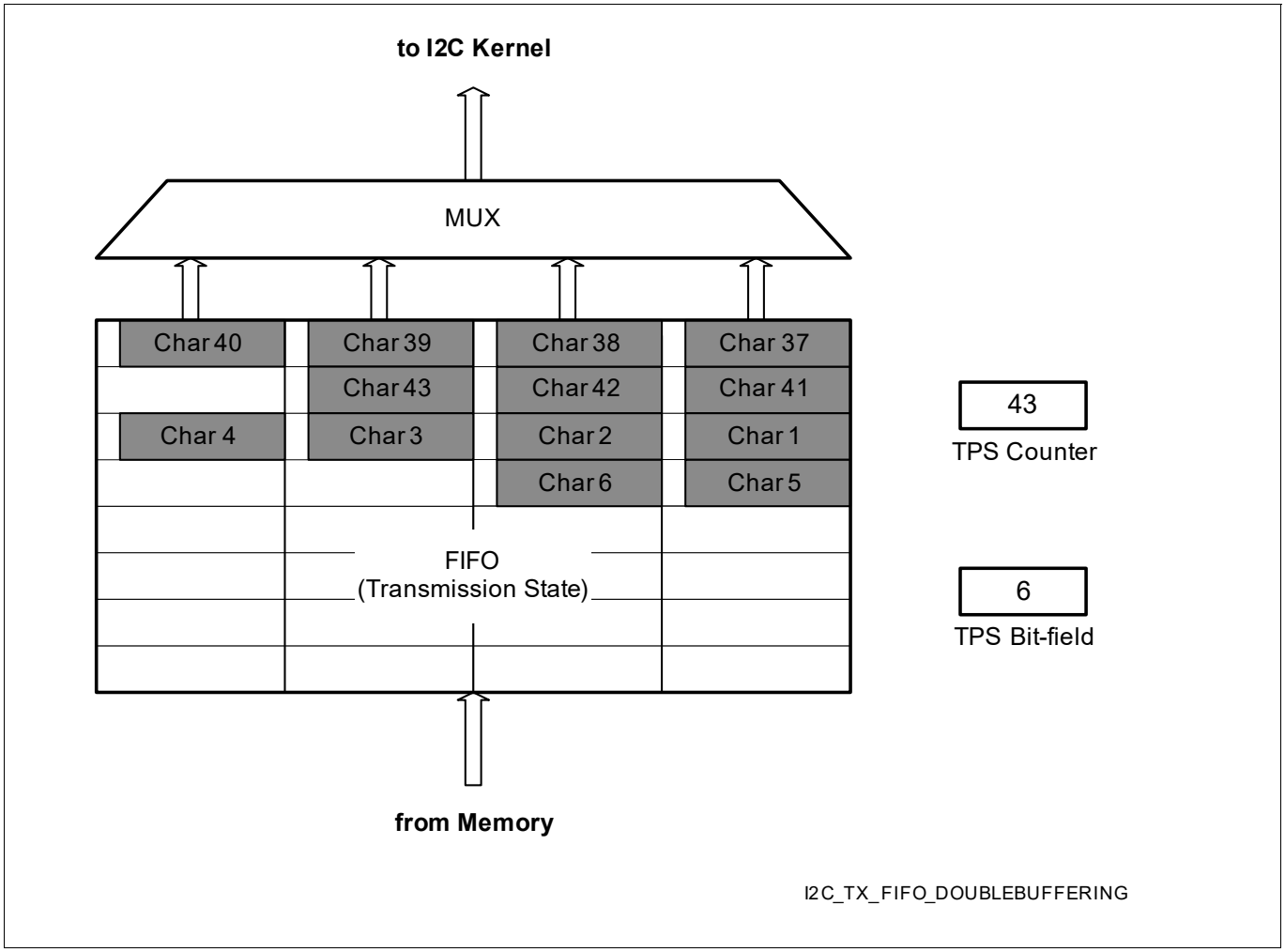


Figure 339 FIFO Containing Data of Two Different Transmit Packets

Error Handling

- If a TX FIFO overflow occurs (i.e. the software writes data too fast to the FIFO), then the new incoming character is discarded and a TX FIFO overflow interrupt request is generated.
- If a TX FIFO underflow occurs (i.e. the I2C kernel tries to read from an empty FIFO), then the I2C kernel generates a TX FIFO underflow interrupt request.

34.3.1.6.2 Transmit Request Generation

For the following it is assumed that the FIFO is used as flow controller, selected via bit **TXFC** of the **FIFOCFG** register.

In this case the bit-field **TPS** of the **TPSCTRL** register is not only used for the generation of the EOTXP (end of transmit packet) signal (see **Section 34.3.1.6.1**), but also to initiate the whole data transfer. I.e. the software indicates that it wants to transmit a data packet by writing the packet size to the bit-field **TPS**. Then the FIFO unit asserts burst requests BREQ until the amount of data still to be transferred is less than or equal to the burst size set in bit-field **TXBS** of register **FIFOCFG**. At this point, if the remaining data is equal to the burst size, then a last burst request LBREQ is issued. Otherwise, single requests SREQ and a last single request LSREQ will be issued.

If a data transfer (BREQ, LSBREQ, SREQ, LSREQ) request is pending, then no more request/interrupt will be done until the corresponding signal REQCLR has been cleared. The CPU (software) or DMA have to transfer the data to the FIFO and clear the generated interrupt.

Notes

1. *Pre-loading **TPS** with the packet size of the next frame, before the current packet has been transmitted completely, enables a continuous data throughput (e.g. of audio data).*
2. *If the FIFO is not used as flow controller, the I2C-bus interface should use bit-field **TPS** of the **TPSCTRL** register to set the EOTXP signal. A burst request BREQ is generated each time the number of empty FIFO stages is equal or greater than the burst size set in bit-field **TXBS** and the previous request was cleared. No single requests SREQ and LSREQ will be issued.*

34.3.1.6.3 Transmit Data Alignment

Depending on bit-field **TXFA** (TX FIFO Alignment) in register **FIFOCFG**, the FIFO can deal with byte aligned, half word aligned or word aligned characters (or even more) as it is described in the following.

Note: Byte alignment is synonymous with character alignment, half word alignment with character alignment of two data characters, and word alignment with character alignment of four characters.

Byte Aligned TX Characters

If the TX characters are byte aligned, then the FIFO extracts the bytes and shifts them to the I2C kernel via a multiplexer as it is shown in **Figure 340**.

Since the number of characters of the packet to be transmitted is not necessarily a multiple of 4, the last word must be filled up with dummy bytes if necessary. The FIFO transmits only the valid bytes of the last word depending on the setting of the bit-field **TPS** of the **TPSCTRL** register.

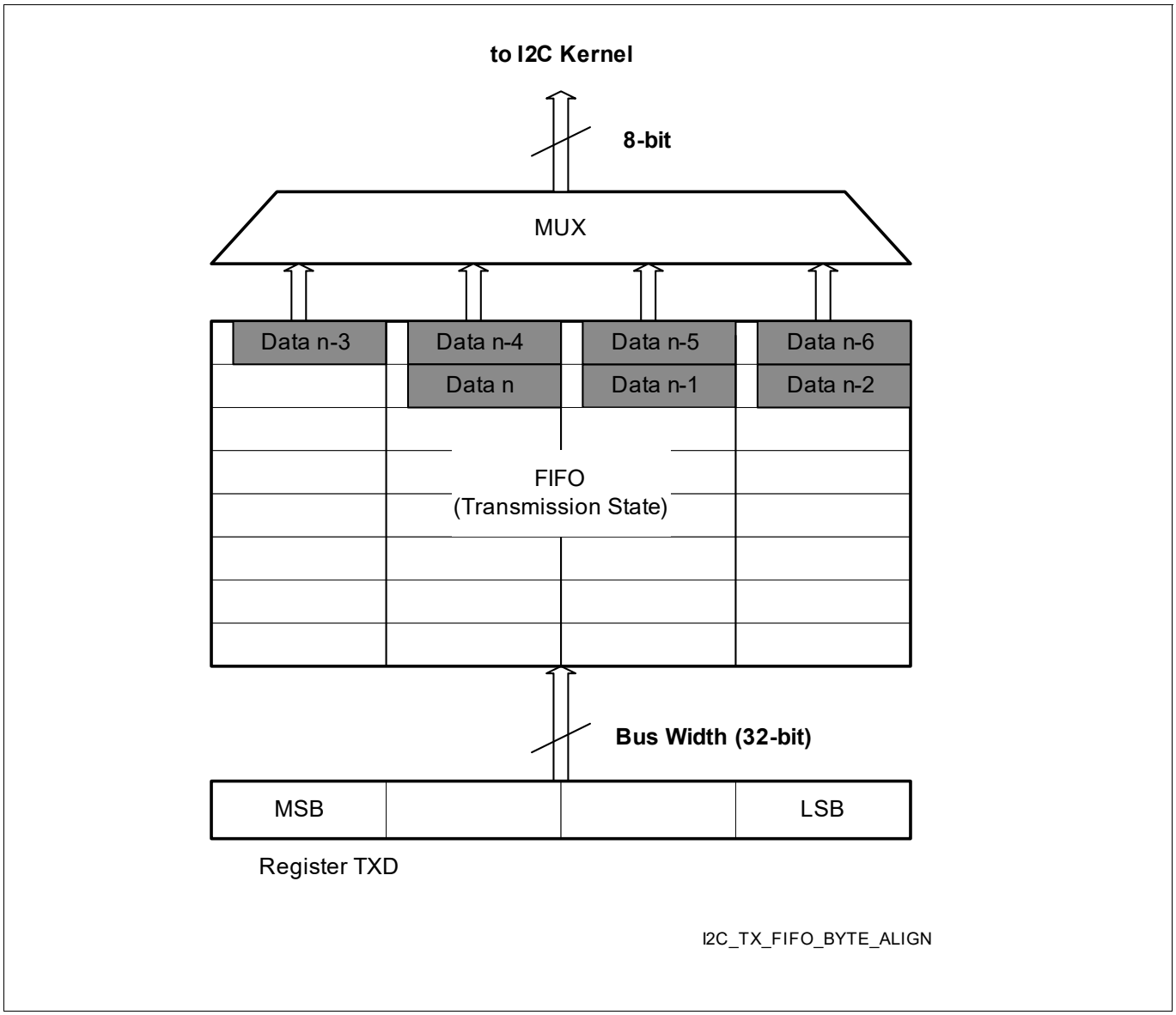


Figure 340 FIFO in Transmission State with Byte Aligned Data

Half Word Aligned

If the TX characters are half word aligned, then the FIFO extracts the half words and shifts them to the I2C kernel via a multiplexer as it is shown in **Figure 341**.

Note: The I2C kernel extracts the right data bits from the half word it receives from the FIFO.

Since the number of characters of the packet to be transmitted is not necessarily a multiple of 2, the last word must be filled up with a dummy half word if necessary. The FIFO transmits only the valid half words of the last word depending on the setting of the bit-field **TPS**.

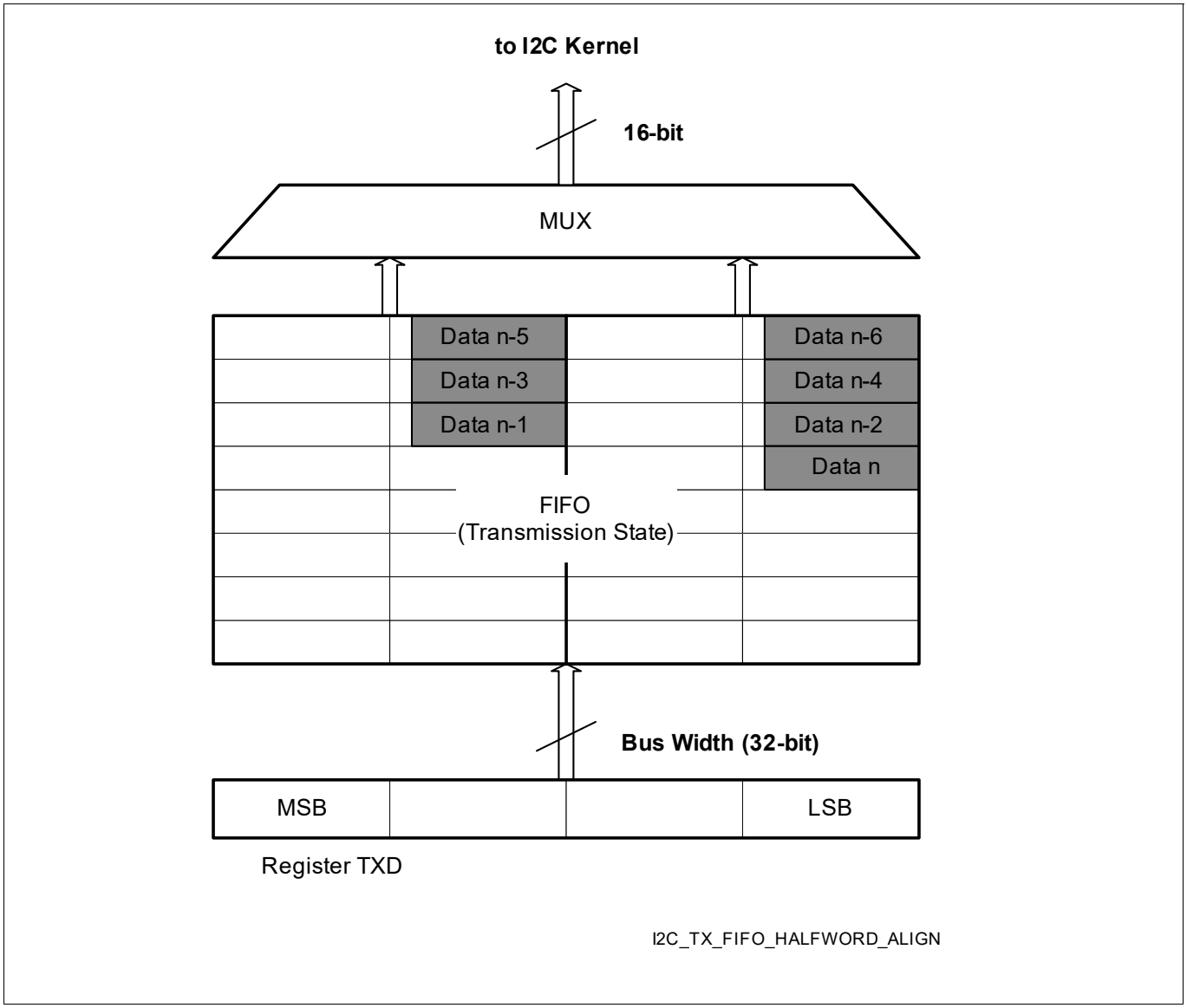


Figure 341 FIFO in Transmission State with Half Word Aligned Data

Word Aligned

If the TX characters are word aligned, then the FIFO shifts them to the I2C kernel as it is shown in **Figure 342**.

Note: The I2C kernel extracts the right data bits from the word it receives from the FIFO.

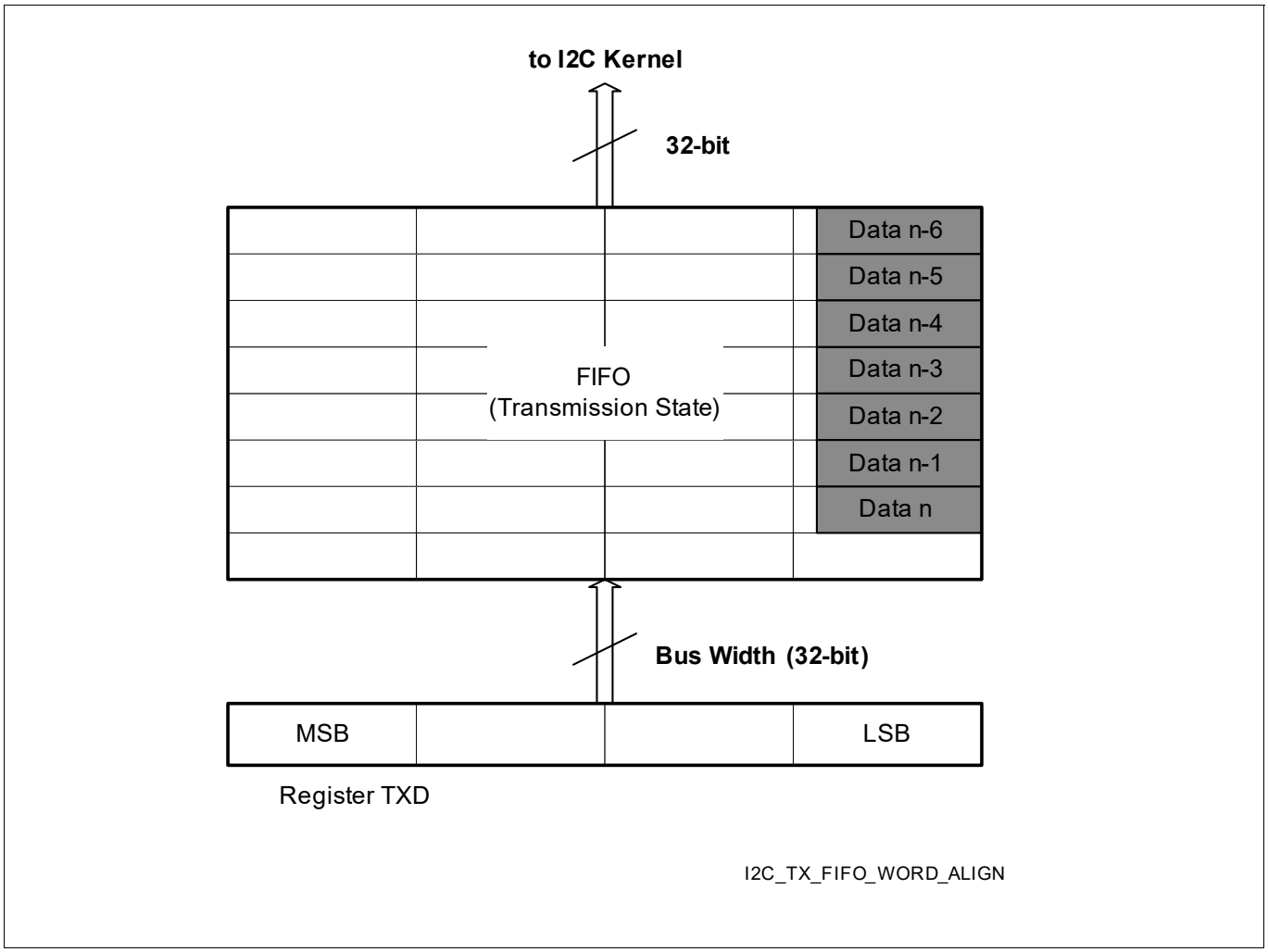


Figure 342 FIFO in Transmission State with Word Aligned Data

34.3.1.6.4 Data Reception

The software can read received data from the FIFO by reading the Reception Data Register **RXD**. Reading data from the FIFO should be done according to the data transfer request signals generated by the FIFO controller. The register **FFSSTAT** indicates the number of filled FIFO stages. The data alignment within the FIFO is described in **Section 34.3.1.6.6**. The I2C kernel shifts data into the FIFO by means of the handshake signals RX_FIFO_rdy and RX_Data_ind.

Before the first reception starts, the I2C kernel has to set the RXS_set line. This causes the assertion of RX_FIFO_rdy to indicate the I2C kernel that one data character can be written into the FIFO.

With the signal FIFO_flush the I2C kernel is able to clear the FIFO content. But if a data transfer request is pending and the FIFO is configured as flow controller, then the FIFO is not cleared until the corresponding signal REQCLR has been cleared.

In bit-field **CRBC** in the register **FIFOCFG**can be configured how to clear the FIFO request.

Error Handling

- If the RX FIFO is completely full (i.e. the software reads out the data too slow), then the I2C kernel generates an RX FIFO overflow interrupt request.
- If an RX FIFO underflow occurs (i.e. the software reads from empty FIFO), then an RX FIFO underflow interrupt request is generated.
- If an error (including the FIFO overflow condition described above) is detected in the I2C kernel and the current reception is stopped, then it also has to generate an EORXP_ind signal, so that the remaining characters in the FIFO can be moved out by means of data transfer requests.

34.3.1.6.5 Receive Request Generation

If the whole data of the current received packet has been moved into the buffer, then the I2C kernel will set the EORXP_ind (end of receive packet indication) signal. The received characters, which are shifted in the FIFO, are counted by anRPS (received packet size) counter. If an EORXP_ind occurs, then the content of the RPS counter is moved to the register **RPSSTAT** and the counter is reset. If data alignment in the FIFO is used, then the **RPSSTAT** value can be used to check for valid characters in the last read word (see **Section 34.3.1.6.6**).

For the following it is assumed that the FIFO is used as flow controller, selected via bit RXFC of the **FIFOCFG** register.

The FIFO unit asserts burst requests BREQ each time the FIFO filling level is greater than the burst size set in bit-field RXBS of register **FIFOCFG**. An EORXP_ind causes the FIFO unit to assert burst requests BREQ until the amount of data still to be transferred is less than or equal to the burst size. At this point, if the remaining data is equal to the burst size, a last burst request LBREQ is issued. Otherwise, single requests SREQ and a last single request LSREQ will be issued.

If a data transfer (BREQ, LSBREQ, SREQ, LSREQ) request is pending, then no more request/interrupt will be done until the corresponding signal REQCLR has been cleared. The CPU (software) or DMA have to get the data from the FIFO and clear the generated interrupt.

When the packet has been shifted out of the FIFO by software, the FIFO controller provides the EORXP_ack (end of receive packet acknowledge) signal to the I2C kernel, which allows the kernel to set the next EORXP_ind.

The next received packet may be shifted into FIFO immediately after the current packet, so that the FIFO contains data of two different packets. But the EORXP_ind of the next packet must not be asserted before the EORXP_ind of the current packet has been acknowledged. The **RPS** counter is counting the characters of the next packet, while in the register **RPSSTAT** the packet size of the current packet remains readable. The first character of the new packet is always shifted in a new FIFO stage as indicated in **Figure 343**. (The FIFO data alignment is described in **Section 34.3.1.6.6**.)

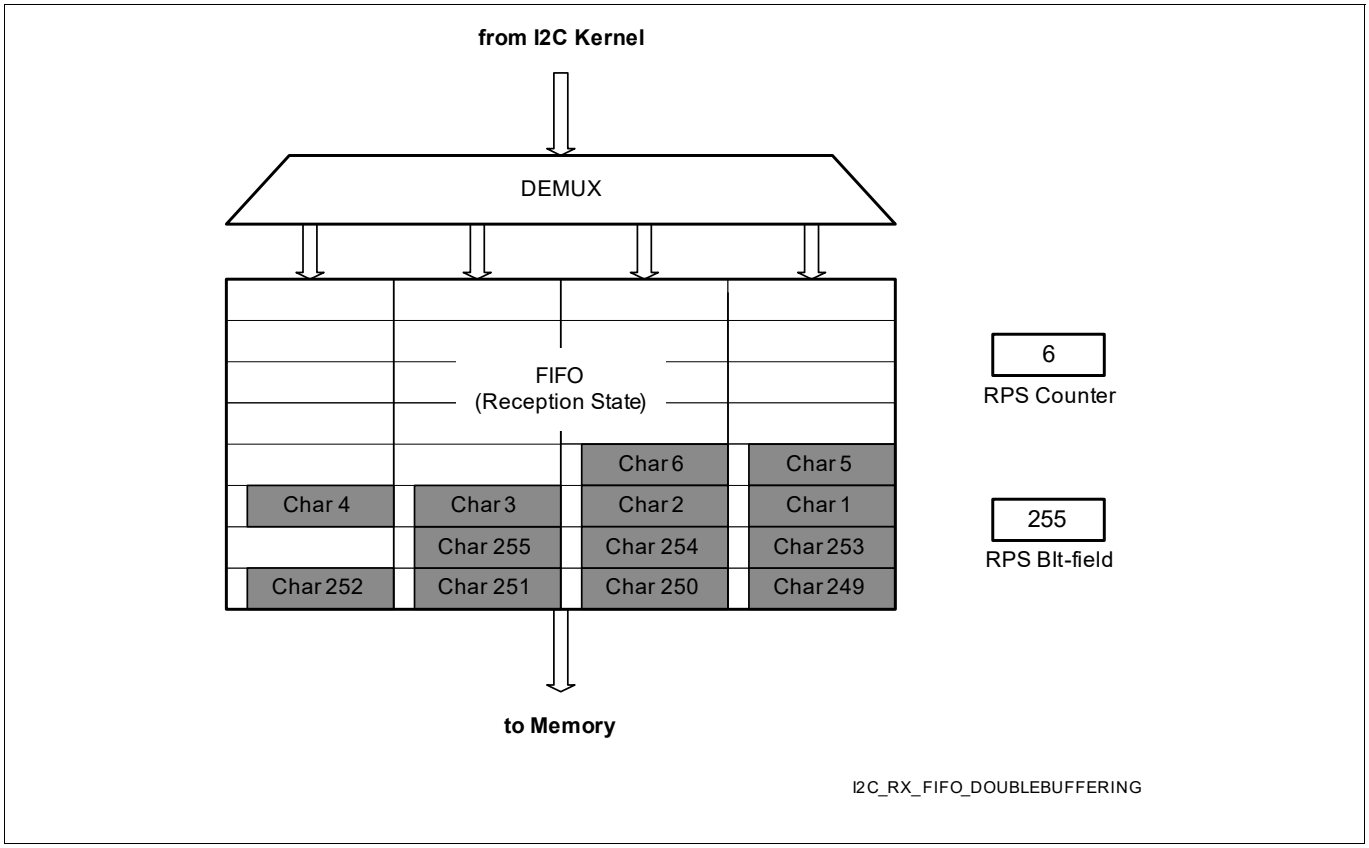


Figure 343 FIFO Containing Data of Two Different Receive Packets

If the number of the received characters of one packet is equal to the maximum received packet size, which is defined by bit-field MRPS in register **MRPSCTRL**, then the FIFO controller gets into the state, where it generates LBREQ or SREQ and LSREQ. If the I2C kernel generates an EORXP_ind in this state, then this indication is valid for the received packet and the FIFO accepts no more characters from the I2C kernel until the current packet has been moved out of the FIFO or the start of a new packet has been detected by asserting RXS_set.

The bit-field **MRPS** is double buffered, i.e. the software can write the **MRPS** value of the next packet as soon as the current **MRPS** value has been loaded, i.e. reading **MRPS** returns 0. It loads the new MRPS value as soon as it has room in the FIFO. If an **MRPS** overflow occurs (one character before the last character is received), then the MRPS_stop trigger to the I2C kernel is generated. But if the next **MRPS** value is written to **MRPS** before the last data character of current frame has been completely received, then no MRPS_stop is generated. The I2C-bus interface will use the MRPS_stop trigger to stop the reception of the corresponding data packet.

But the **MRPS** bit-field can also be used to initiate a reception, when the peripheral is master-receiver. Therefore writing to MRPS generates a trigger MRPS_start to the I2C kernel. The next MRPS value written to **MRPS** generates only an MRPS_start when an MRPS_stop was generated before.

The MRPS features can only be used if the FIFO is the flow controller.

Notes

1. *When the MRPS control feature of the FIFO is used to limit the packet size of a continuous incoming data stream (MRPS is only written once and left unchanged), characters exceeding MRPS are discarded. The FIFO can not handle more than two packets. When the FIFO is filled with data characters of two packets then a further RXS_set will be overseen by the FIFO. Additionally the FIFO does not set the RX_FIFO_rdy.*
2. *If the FIFO is not used as flow controller, a single request SREQ is always generated as soon as the readable FIFO stage is filled up with characters. Additionally, a burst request BREQ is generated as soon as the number of filled FIFO stages is equal to or greater than the burst size set in bit-field RXBS of register FIFOCFG. When the I2C kernel*

sets EORXP_ind, the FIFO controller provides the EORXP_ack (end of receive packet acknowledge) signal to the I2C kernel, which allows the kernel to set the next EORXP_ind.

34.3.1.6.6 Receive Data Alignment

Depending on bit-field **RXFA** (RX FIFO Alignment) in register **FIFOCFG**, the FIFO can deal with byte aligned, half word aligned or word aligned characters (or even more) as it is described in the following.

Byte Aligned

If the RX data should be byte aligned, then the characters from the I2C kernel are aligned and shifted to the FIFO by a demultiplexer as it is shown in **Figure 344**.

Since the number of characters of the received packet is not necessarily a multiple of 4, the upper bytes of the last word can be invalid. The software has to check for invalid bytes of the last word by means of the bit-field **RPS** (received packet size) in register **RPSSTAT**.

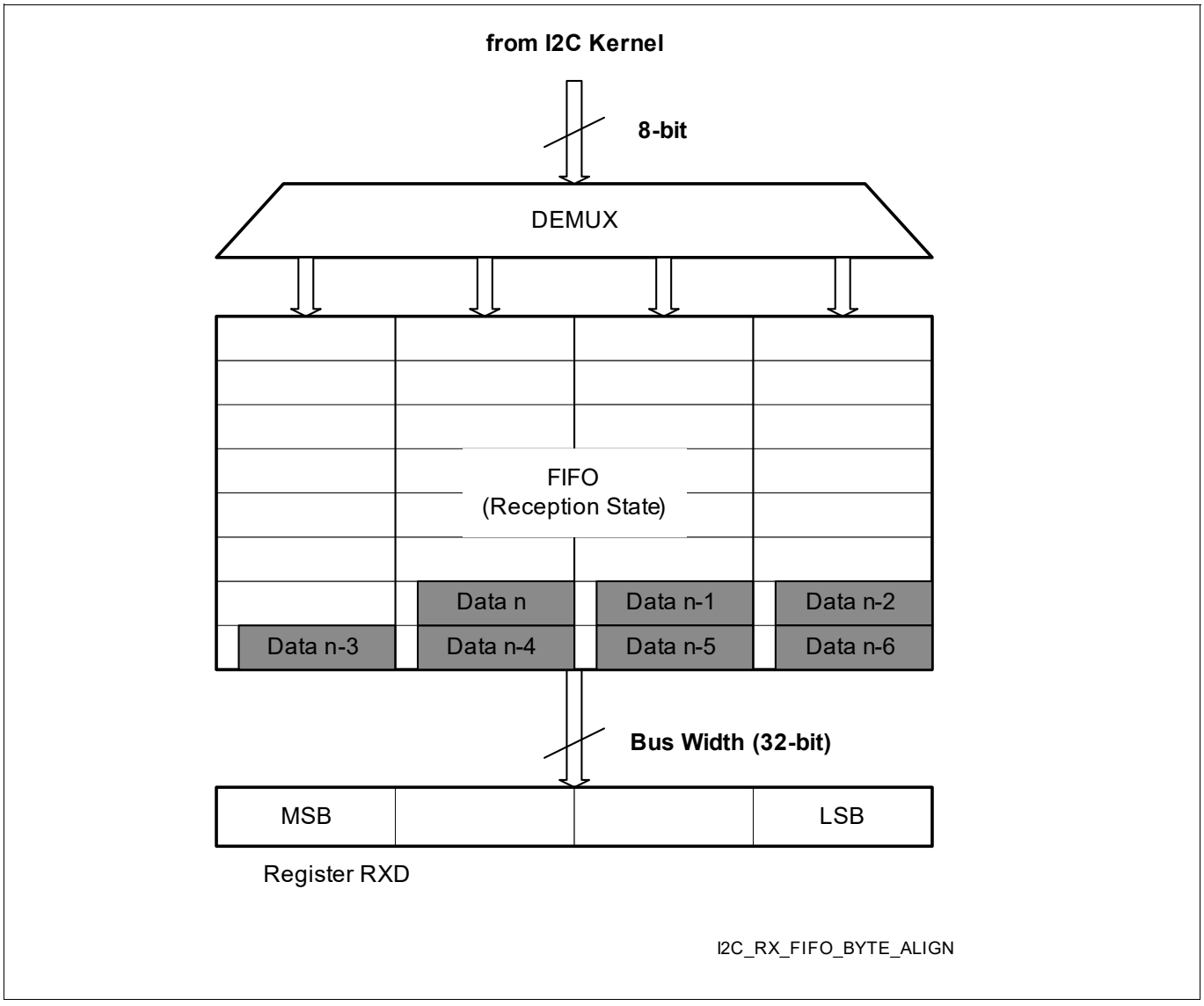


Figure 344 FIFO in Reception State with Byte Aligned Data

Half Word Aligned

If the RX data should be half word aligned, then the characters from the I2C kernel are aligned and shifted to the FIFO by a demultiplexer as it is shown in **Figure 345**.

Note: The I2C kernel fills up the half words with dummy bits.

Since the number of characters of the received packet is not necessarily a multiple of 2, the upper bytes of the last word can be invalid. The software has to check for invalid half words of the last word by means of the bit-field RPS (received packet size) in register **RPSSTAT**.

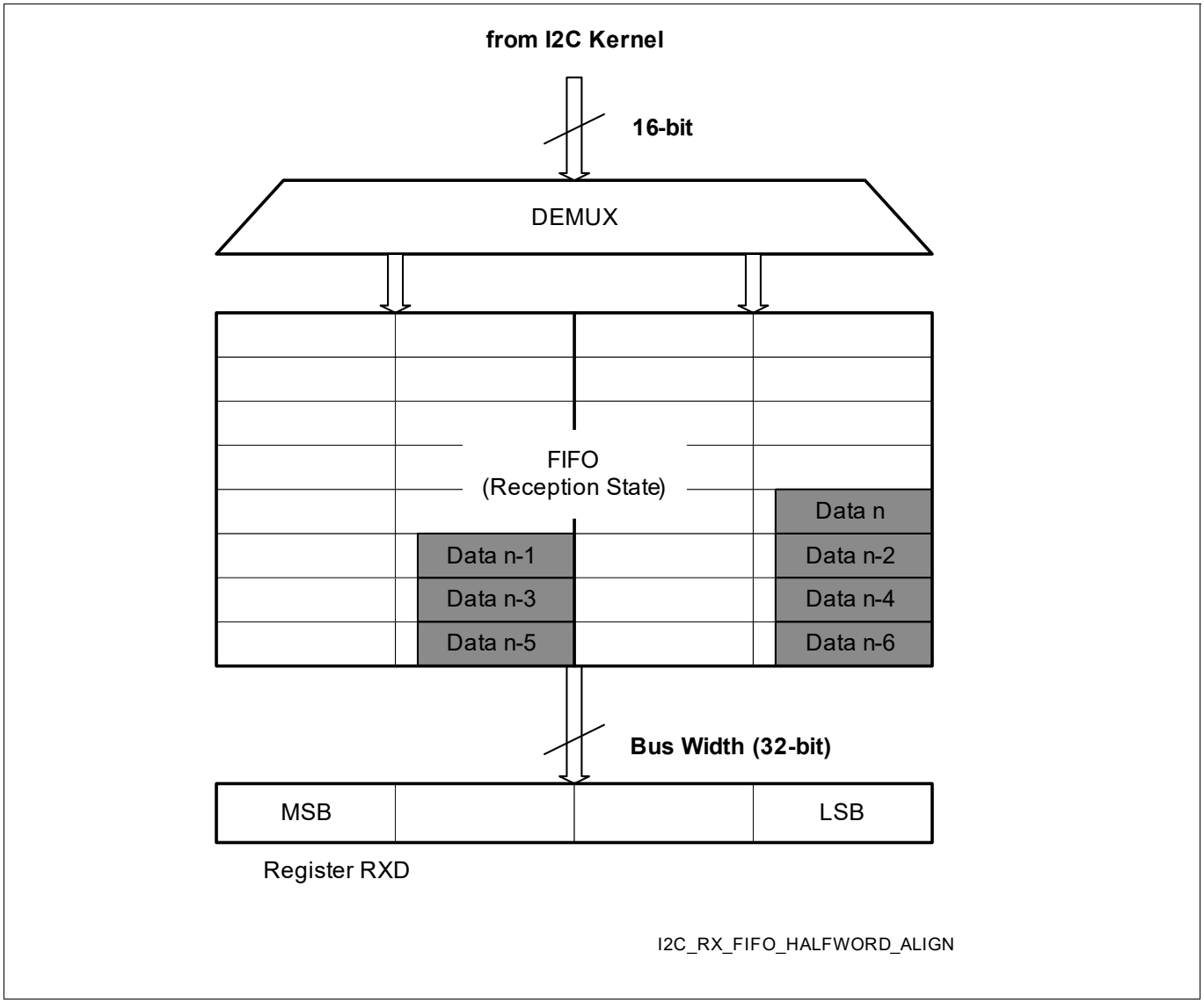


Figure 345 FIFO in Reception State with Half Word Aligned Data

Word Aligned

If the RX data should be word aligned, then the characters from the I2C kernel are shifted to the FIFO as it is shown in **Figure 346**.

Note: The I2C kernel fills up the words with dummy bits.

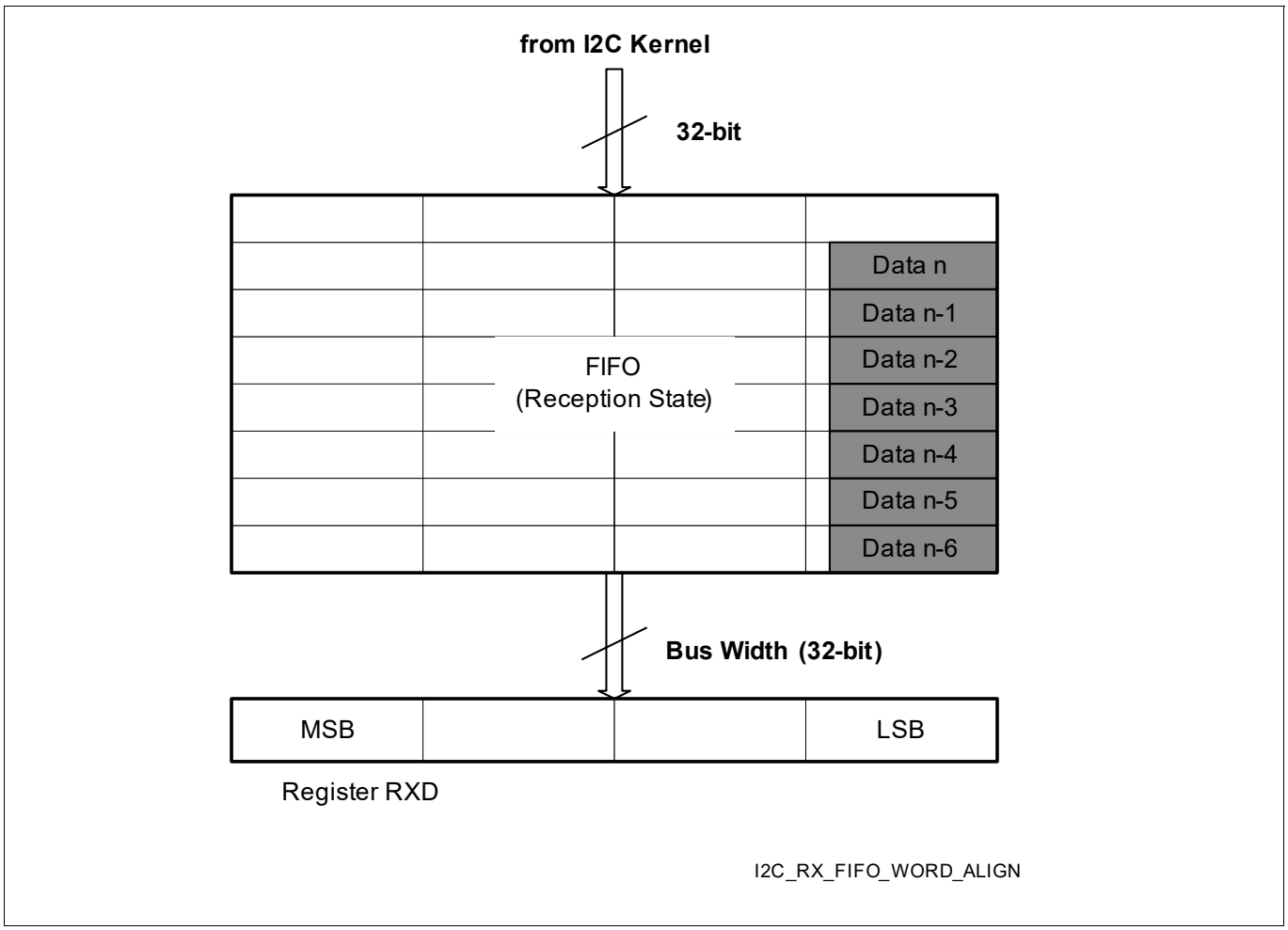


Figure 346 FIFO in Reception State with Word Aligned Receive Data

34.3.1.6.7 Switching between Transmission and Reception

Initially, the FIFO is in the TX state, so that the software can initiate a TX transfer simply by writing data to the bit-field **TXD**. If the I2C kernel wants the FIFO to switch to the RX state in the meantime, then it can set the RXS_set line and even if transmission is ongoing the TX transfer is aborted. This causes the FIFO to flush its content, switch to the RX state and reset the received packet size counter. If the software wants to write to the register **TXD** when the FIFO is in RX state, then this causes a bus error. As soon as the FIFO has received an EORXP_ind signal from the kernel and the software has moved all RX characters out of the FIFO via the register **RXD**, the FIFO automatically switches back into the TX state.

If the software writes to the register **TPSCTRL** when the FIFO is in the RX state, the TPS value is pending until the FIFO returns to the TX state. Then the transmission is initiated, if the FIFO is the flow controller.

If the FIFO is flow controller and the software writes to the register **MRPSCTRL**, then an MRPS_start is generated independent of the state of the FIFO.

With the signal FIFO_flush the I2C kernel is able to clear the FIFO content. But if a data transfer request (xREQ) is pending, then the FIFO is not cleared until the corresponding signal REQCLR has been cleared.

If the FIFO has been cleared in the RX state, then it switches back to the TX state afterwards.

34.3.1.6.8 Switching between Reception and Transmision

When the I2C kernel is changing from RX state to TX state the FIFO is flushed. To avoid loosing data when the FIFO is flushed the software should proceed in the right sequence. In a scenario where the device is addressed as slave and is asked to return data, this new data must be entered in the FIFO only after detection of the address and TX_END interrupt requests and then can transfer the data to the FIFO or can trigger the DMA that fills the FIFO for the TX transfer.

34.3.1.7 Service Request Block Operation

The SRB (Service Request Block) of the I2C module is used to prepare the interrupt and data transfer requests for the interrupt controller.

34.3.1.7.1 Overview of Service Requests

The I2C service requests are partially combined in the SRB (see **Section 34.4.4** and **Section 34.4.5**). **Figure 347** shows an overview of the service requests. **Table 301** provides a list of all service requests of the I2C module.

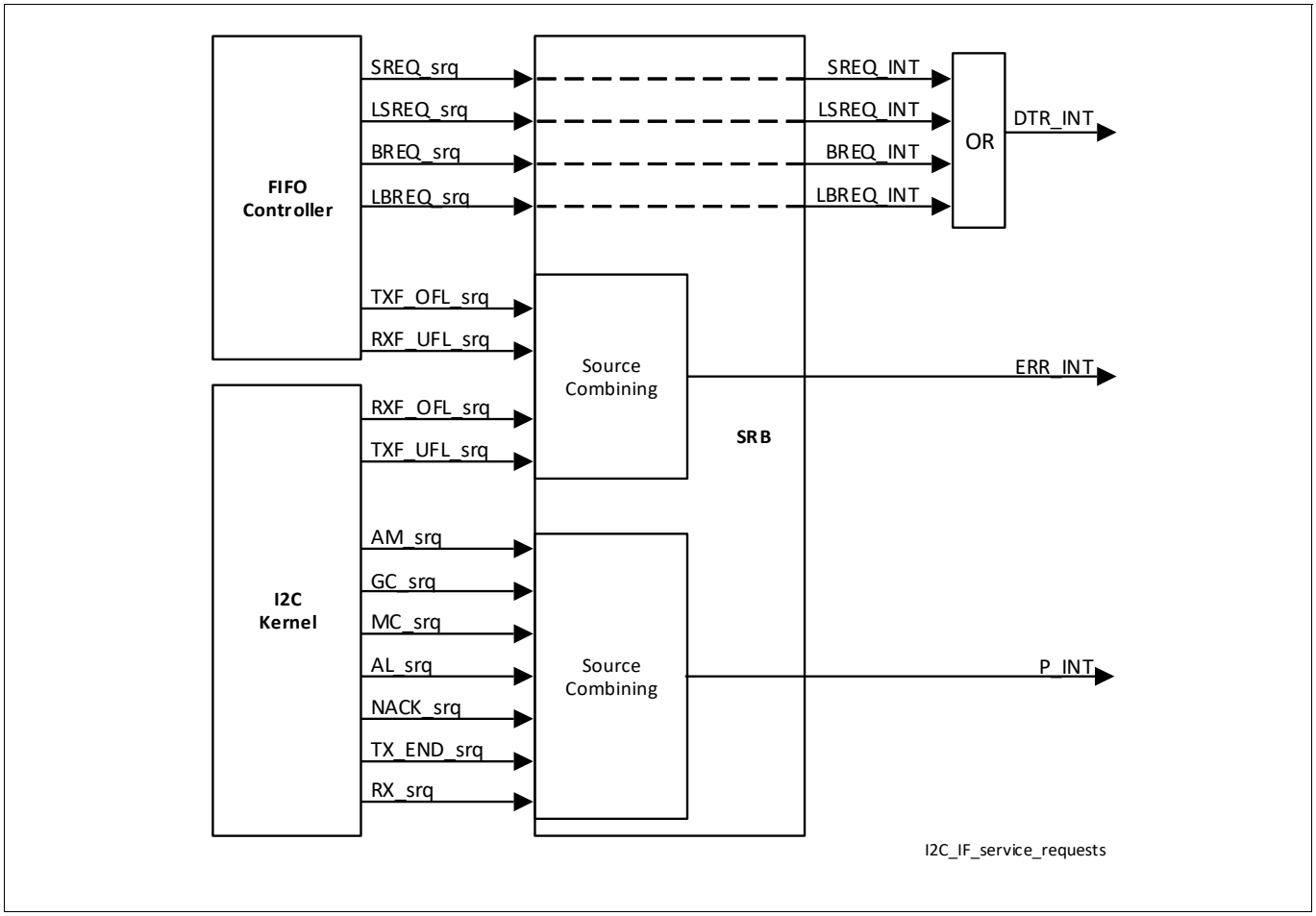


Figure 347 Overview of I2C Module Service Requests

Table 301 I2C Module Service Requests

Service Request	Interrupt Request	Description
BREQ_srq	DTR_INT	**Burst Data Transfer Request** FIFO requests a transfer of a programmed burst number of words from/to the memory.
LBREQ_srq	DTR_INT	**Last Burst Data Transfer Request** FIFO requests a last burst transfer from/to the memory.
SREQ_srq	DTR_INT	**Single Data Transfer Request** FIFO requests a single transfer of a word from/to the memory.
LSREQ_srq	DTR_INT	**Last Single Data Transfer Request** FIFO requests a last single transfer from/to the memory.
TXF_OFL_srq	ERR_INT	**TX FIFO Overflow Request** FIFO has detected a TX FIFO overflow.
TXF_UFL_srq	ERR_INT	**TX FIFO Underflow Request** I2C kernel has detected a TX FIFO underflow. The transmission is finished after the current byte. A stop condition is generated if the kernel works as master. The kernel changes to listening state.
RXF_OFL_srq	ERR_INT	**RX FIFO Overflow Request** I2C kernel has detected an RX FIFO overflow and the incoming character is discarded. The kernel puts a not-acknowledge on the bus and changes to listening state. A stop condition is generated if the kernel works as master.
RXF_UFL_srq	ERR_INT	**RX FIFO Underflow Request** FIFO has detected an RX FIFO underflow.
AM_srq	P_INT	**Address Match Request** Device (master/slave) has been addressed by a remote master (also indicated in bit-field BS in register **BUSSTAT**).
GC_srq	P_INT	**General Call Request** When the general call matching process is activated this interrupt indicates that another master has put a general call on the I2C-bus.
MC_srq	P_INT	**Master Code Request** When the master code matching process is activated this interrupt indicates the appearing of a master code on the I2C-bus issued by a remote master. The request is generated after a not-acknowledge and the clock is released to high again.
AL_srq	P_INT	**Arbitration Lost Request** Arbitration is lost after the device has tried to start a transmission on the I2C_bus.
NACK_srq	P_INT	**Not-acknowledge Received Request** Not-acknowledge received when working as transmitter (i.e. RnW bit is 0).

Table 301 I2C Module Service Requests (cont'd)

Service Request	Interrupt Request	Description
TX_END_srq	P_INT	**Transmission End Request** In master mode this event is produced by the I2C kernel to indicate that the transmission of the current packet has ended properly after the stop condition has been put on the I2C-bus or MASTER RESTART state has been entered. (At this point, a restart condition can be generated or the connection can be finished by generating a stop condition). This request is created in master mode at any stop condition to indicate that the bus is free again and it could be obtained. In slave mode this event is produced by the I2C kernel if it was addressed by a master and the current transmission was terminated by a stop or restart condition.
RX_srq	P_INT	**Receive Mode** Request I2C kernel indicates to the FIFO switching from transmit to receive mode. When FIFO is operating in non flow controller mode, this service request can be used to distinguish between transmit and receive FIFO data requests.

The generation of these events is visualized in the state machine **Figure 334**, **Figure 335** and **Figure 336**. The timing is shown in **Figure 337**.

34.3.1.7.2 Interrupt Service Request Structure

The I2C module provides level based service request lines, which can be processed by an interrupt controller. The service requests must be cleared in the interrupt service routine. For test purposes, all service requests can also be set by SW via a register bit. All service requests can be masked within the peripheral. Furthermore, requests that are not necessarily mutually exclusive are combined in order to reduce the number of request lines.

Flow of Data Transfer Requests

Figure 348 shows the flow of a data transfer request, which comes from the FIFO controller.

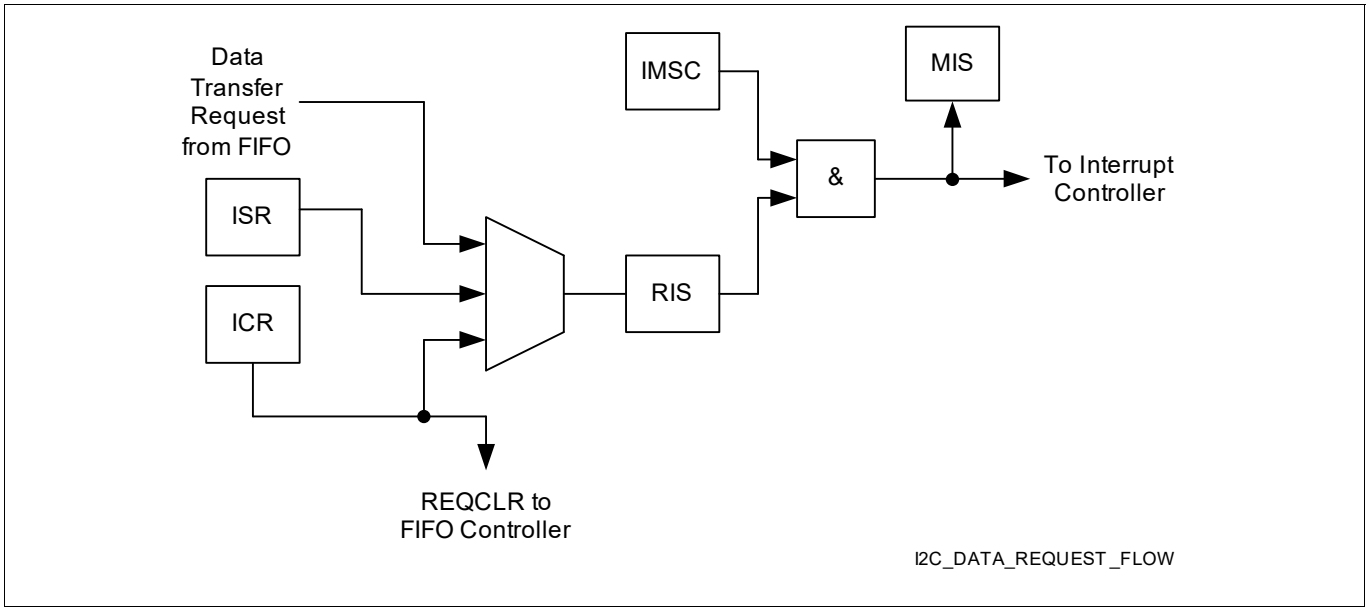


Figure 348 Data Transfer Request Flow

A data transfer request sets the corresponding status bit in the Raw Interrupt Status Register **RIS**. The status bit can also be set by writing 1 to the corresponding bit in the Interrupt Set Register **ISR**. It will be cleared by writing 1 to the corresponding bit in the Interrupt Clear Register **ICR**.

The Interrupt Mask Control Register **IMSC** enables or disables the requests to the interrupt controller. The Masked Interrupt Status Register **MIS** contains the current masked values of the requests.

If a request is disabled via register **IMSC** while the request is active, the request will be removed but only until the **IMSC** bit is set again, unless the request has been cleared in the meantime.

If a request is disabled via register **IMSC** and the source becomes active, then the request bit will only be set in the **RIS** register. If the corresponding **IMSC** bit is later enabled, the request will consequently become active in the **MIS** register.

Before enabling a request, it is good practice to always first clear the corresponding bit in the **RIS** register via **ICR**.

Multiple Source Interrupt Requests

For error and protocol interrupt requests which have multiple sources, the interrupt structure is extended. Additional register sets are implemented within the SRB.

Figure 349 gives an overview of the combining of several request sources to a single request.

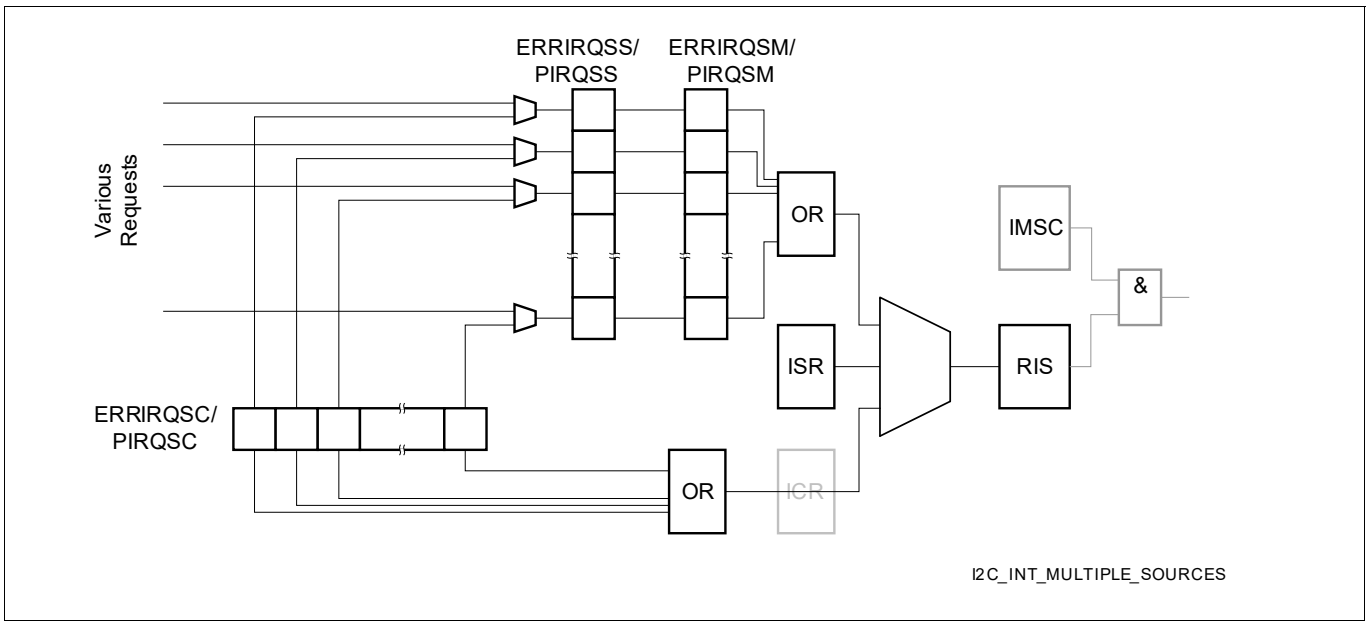


Figure 349 Interrupt Request with Multiple Sources

An interrupt request sets the corresponding raw status bit in the Interrupt Request Source Status Register **ERRIRQSS** (for error) or **PIRQSS** (for protocol).

The status bits can be cleared via the Interrupt Request Source Clear Register **ERRIRQSC** (for error) or **PIRQSC** (for protocol). If no further error/protocol request sources are active, the whole error/protocol interrupt request is cleared. This register replaces the functionality of the bit I2C_ERR_INT (for error) or bit I2C_P_INT (for protocol) in the Interrupt Clear Register **ICR**.

The Interrupt Request Source Mask Register **ERRIRQSM** (for error) or **PIRQSM** (for protocol) enables or disables the interrupt requests. The request lines from the enabled sources are combined (OR) to a single level sensitive request line, which acts as input for the corresponding bit I2C_ERR_INT (for error) or I2Cm_bit I2C_P_INT (for protocol) in the Raw Interrupt Status Register **RIS**.

34.3.2 I2C Module Implementation

This section describes I2C module interfaces with the clock control, port control and interrupt control.

34.3.2.1 Interfaces of the I2C Module(s)

Figure 350 shows the specific implementation details and interconnections of the I2C module(s):

- A clock control block generates the clock signals required for the module.
- For the I2C-bus lines SCL and SDA, different I/O lines can be selected.
- The interrupt outputs of the module are connected to the interrupt control unit.

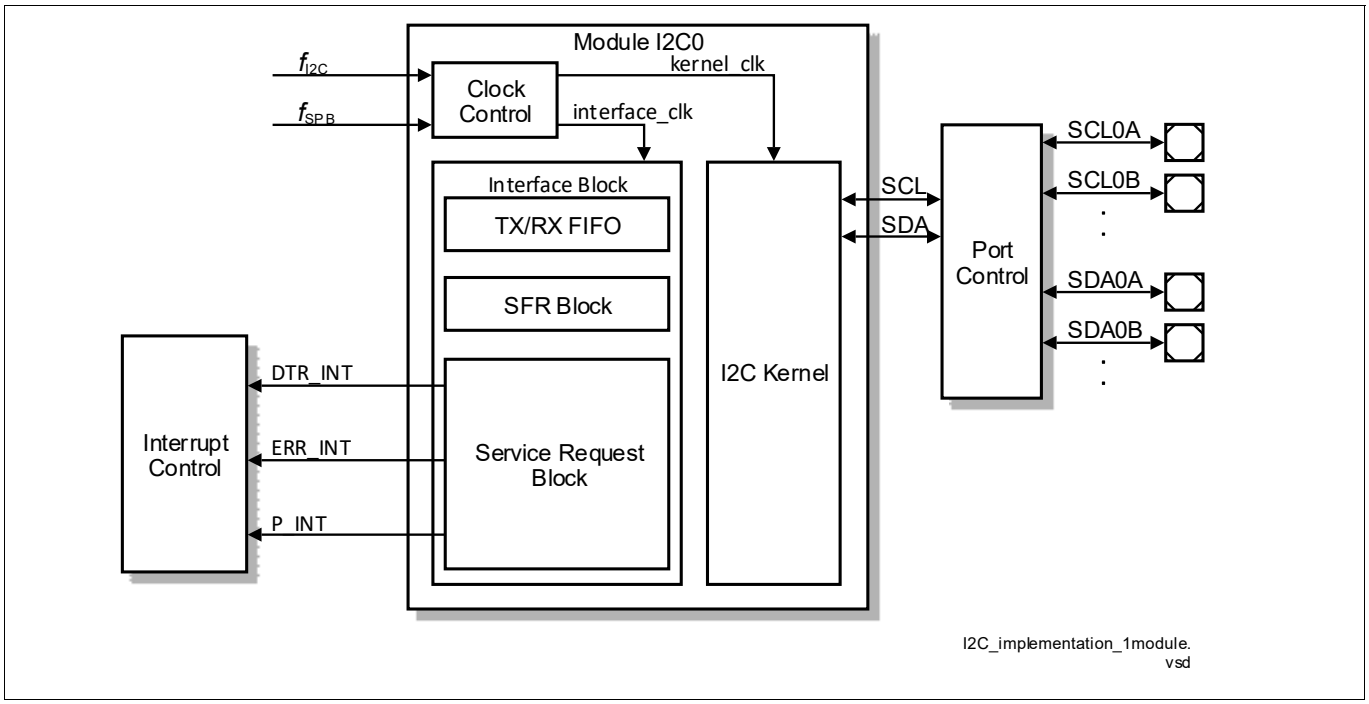


Figure 350 I2C Module Implementation and Interconnections

34.3.2.2 Module Clock Control

The clock control allows the programmer to adapt the peripherals functionality and power consumption to the application's requirement. By programming the kernel's operating frequency, an optimal ratio between power consumption, EMC behavior and functionality can be achieved. Furthermore, for power saving reasons the peripheral can be disabled as a whole by switching off the peripheral's clocks via clock gating cells.

A simplified description of the clock control is shown in **Figure 351**. See register **CLC1** for a description of the clock control parameters.

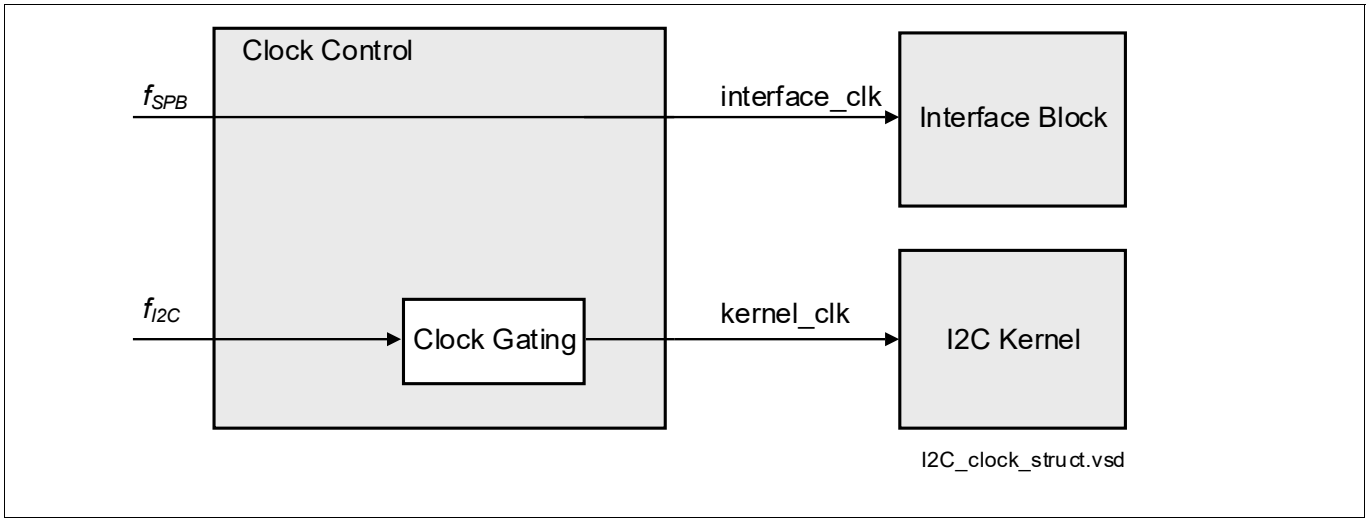


Figure 351 Module Clock Control

The following clocking modes are supported for the peripheral:

- Disabled mode
- OCDS suspend mode

Disabled Mode

In Disabled Mode, the clocks are stopped and register write accesses are not possible (except access to **CLC1** register itself).

There are several ways possible to bring the module into Disabled mode:

- Software disable
  - Set **CLC1**.DISR = 1 (software disable request)
- Hardware disable
  - Must be enabled via **CLC1**.EDIS = 0
  - Drive input signal sleep_n_i = 0(active low: 0= disable; 1= enable)

In all cases, current bus accesses are finished and a kernel handshake is issued to make sure that the state machine is in a safe state (same as **CLC1**.FSOE = 0 in OCDS Suspend Mode; see description below).

Module is in Disabled Mode after reset and must be enabled by setting **CLC1**.DISR = 0 and **CLC1**.RMC > 0-.

OCDS Suspend Mode

In this mode the clocks are disabled. If bit SPEN in register **CLC1** is set, the mode can be entered by activating the input module_ocds.

In OCDS suspend mode all registers of the suspended module can be read. Note that each read during disabled clock can not affect hardware. For example multiple reads to FIFO port registers always return the same value as the FIFO control is not working.

No write access to the registers is supported during suspend mode, except writes to the **CLC1** register. For writing other registers the suspend mode for the module has to be removed first by clearing bit SPEN. After the write access is executed, bit SPEN should be set again.

OCDS suspend mode supports two different disable features, selectable by bit FSOE in register **CLC1**:

*Note: To write to SPEN or FSOE, bit SBWE in register **CLC1** must also be set in the same write access.*

- **Secure Clock Switch Off**

Disabling the clock via secure clock switch off additionally activates the handshake mechanism with the peripheral. The peripheral is switched to a secure state before disabling the clock.

Note: This feature is only available in slave mode. In master mode, there is no difference and I2C will always perform "fast clock switch off".

- **Fast Clock Switch Off**

Selecting the fast clock switch off stops the clock immediately after all pending read and write accesses are finished (including synchronization mechanisms with the I2C kernel).

34.3.2.3 Bus Peripheral Interface Registers

Clock Control 1 Register

This register controls the clock gating and dividing circuitry of the peripheral.

CLC1
Clock Control 1 Register **(00000_H)** **Application Reset Value: $0000\ 0003_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								0							
r								r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMC								0		FSOE	SBWE	EDIS	SPEN	DISS	DISR
rwh								r		rw	w	rw	rw	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable not requested 1_B Module disable requested
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module. 0_B Module enabled 1_B Module disabled
SPEN	2	rw	**Module Suspend Enable Bit for OCDS** 0_B Module suspend disabled 1_B Module suspend enabled
EDIS	3	rw	**External Request Disable** *Note: This bit is not used in AURIX™ TC3xx Platform and should always be written with 0.* 0_B External clock disable request is enabled 1_B External clock disable request is disabled
SBWE	4	w	**Module Suspend Bit Write Enable for OCDS** This bit is always read as 0. 0_B Bits SPEN and FSOE are write protected 1_B Bits SPEN and FSOE are overwritten by respective value of SPEN or FSOE
FSOE	5	rw	**Fast Switch Off Enable** 0_B FSOE Clock switch off in OCDS suspend mode via Disable Control Feature (Secure Clock Switch Off) 1_B Fast clock switch off in OCDS suspend mode

Field	Bits	Type	Description
RMC	15:8	rwh	**Clock Divider for Standard Run Mode** Max. 8-bit divider value If RMC is set to 0 the module is disabled. *Note: As long as the new divider value RMC is not valid, reading register returns 0000 00XX$_H$*
0	7:6, 23:16, 31:24	r	**Reserved** Read as 0; should be written with 0.

34.3.2.4 Port and I/O Line Control

Not only the interconnections between the I2C module and the port I/O lines have to be configured, but also the function and the characteristics of the port pins. The following control operations must be executed for the used port pins:

- Selection of I2C module via output port multiplexer
- Configuration of output port function with open drain
- Configuration of pad driver characteristics
- Selection of I2C input lines

34.3.2.5 Interrupt Control

The interrupt functionality is described in the chapter for the interrupt router (IR).

The signal names used in the interrupt router description are explained in **Table 302**.

Table 302 Interrupt Router Signal Names for I2Cm Module (m = 0/1)

Signal Name	Name in Module	Explanation
SRC_I2CmDTR	DTR_INT	**Data Transfer**
SRC_I2CmERR	ERR_INT	**Error**
SRC_I2CmP	P_INT	**Protocol**

34.3.3 Module Integration

This section describes the topics regarding the integration of the module on chip.

34.3.3.1 Integration Overview

The I2C module consists of a delivery providing an AHB bus interface integrated into the FPI bus system by using an FPI2AHB adapter.

The adapter contains a slave bus interface providing translation between the FPI and the AHB bus protocols. The module supports additionally the following features:

- power saving (sleep mode)
- debug suspend
- local module reset
- PISEL - port input selection regarding the serial data input

The power saving and debug suspend features are supported by the AHB based I2C module itself, and the local module reset and PISEL features are supported by the FPI2AHB adapter, see **Figure 352**.

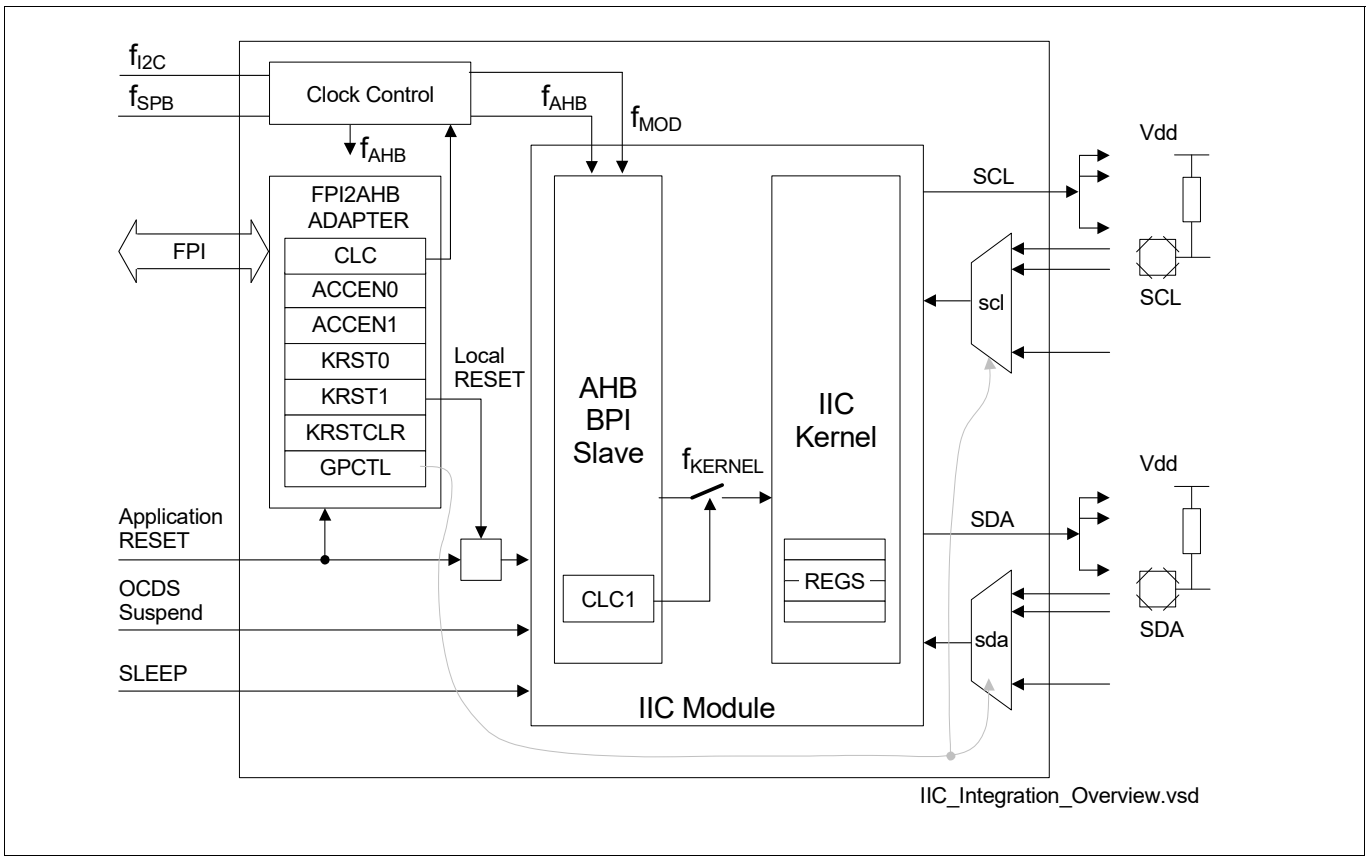


Figure 352 Integration Overview

34.3.3.2 BPI_SPB Module Registers Overview

Figure 353 shows all registers associated with the BPI_FPI module, configured for one kernel. See also **Table 303** for the address mapping.

BPI_FPI Registers Overview

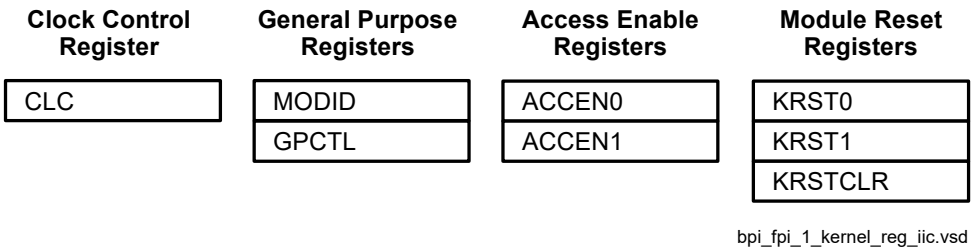


Figure 353 BPI_FPI Registers

This section describes the registers implemented in the slave (FPI2AHB) adapter component.

34.3.3.2.1 BPI_SPB Module Registers

Kernel Reset

If a kernel reset is requested, the adapter will synchronously assert the internal kernel reset output and error any ongoing accesses on the FPI bus not addressed to the adapter's internal SFRs.

The method of using the kernel reset output from the adapter to initialise the associated AHB module is module specific and outside the scope of this specification.

Any AHB accesses in progress will be synchronously terminated.

GPCTL

A single, adapter specific, SFR is implemented in the adaptor.

The SFR contains general purpose read/write control bits without ENDINIT protection. All the bits are connected to output ports on the FPI2AHB entity. The SFR will implement no safety requirements and is not be suitable for controlling hardware related to safety functions. The SFR supports, byte, half word and word transactions only. All other transactions are rejected with an FPI error termination.

Note: In the I2C module, this register implements the PISEL functionality. The SDA and SCL input multiplexors are controlled in parallel with each PISEL setting. Since 3 bits are implemented for PISEL, up to 8 sets of SDA and SCL pins can be defined.

Principle of Operation

When the controlling state machine detects an FPI access, it compares the address to the address of the SFR. In the case that the address matches the SFR address, the normal state machine transitions are interrupted and the access is not passed to the AHB bus.

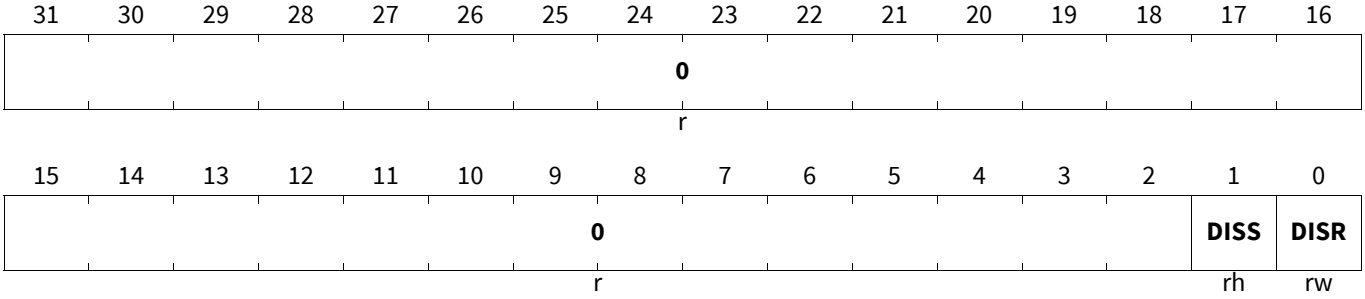
If the access opcode is not SDTB, SDTH or SDTW, then the access is terminated with an FPI error. If the opcode is supported, the SFR data will be returned on the FPI bus if the transaction is a read or the appropriate bits of the register will be updated if the access is a write.

Clock Control Register

The Clock Control Register, CLC, acts globally and allows the complete AHB module to be disabled to reduce power consumption when the module is not required. When the module is disabled (DISS=1_B), only register accesses to the adapter register address space are permitted. All other accesses to the module address space will be errored.

CLC

Clock Control Register **(10000_H)** **Application Reset Value: $0000\ 0003_H$**



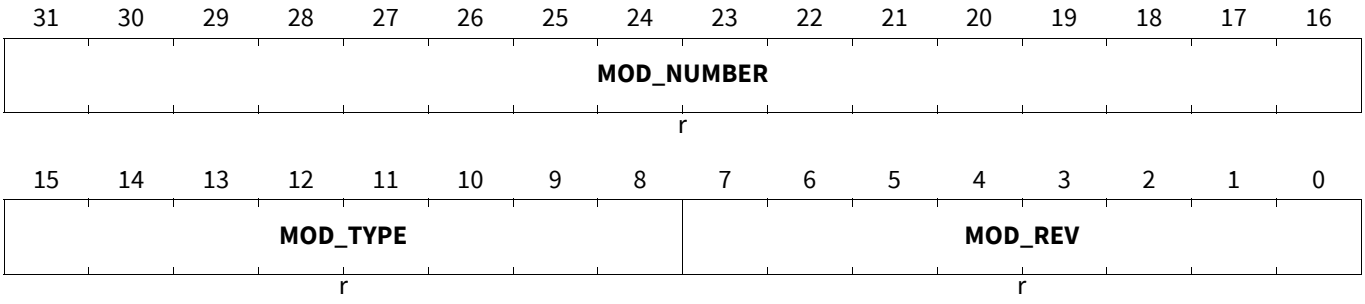
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Module Identification Register

The identification register allows the programmer version-tracking of the module.

MODID

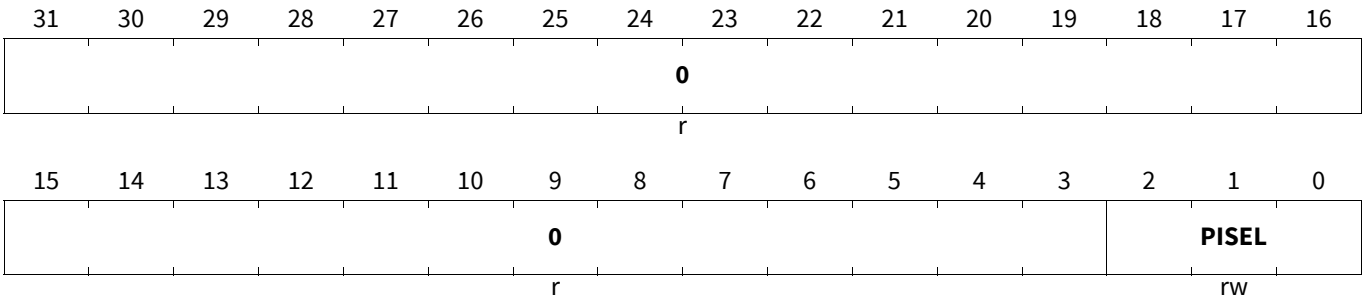
Module Identification Register **(10004_H)** **Application Reset Value: $00C2\ C0XX_H$**



Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01H (first revision).
MOD_TYPE	15:8	r	**Module Type** The bit field is set to C0H which defines the module as a 32-bit module.
MOD_NUMBER	31:16	r	**Module Number Value** This bit field defines a module identification number.

General Purpose Control Register

GPCTL
General Purpose Control Register (10008$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
PISEL	2:0	rw	**Port Input Select** Used to select the input pins providing the serial data and clock input signals, in the range of 0 to 7. *Note: In AURIX™ TC3xx Platform, not all PISEL options are available. See Data Sheet.* 000_B SDA0A and SCL0A are selected. 001_B SDA0B and SCL0B are selected. 010_B SDA0C and SCL0C are selected. 011_B SDA0D and SCL0D are selected. 100_B SDA0E and SCL0E are selected. 101_B SDA0F and SCL0F are selected. 110_B SDA0G and SCL0G are selected. 111_B SDA0H and SCL0H are selected.
0	31:3	r	**reserved** Reads 0. Should be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions to registers with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The adapter is prepared for an 6 bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6 bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B,...,EN31 -> TAG ID 011111B.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

ACCEN0

Access Enable Register 0 **(1000C$_H$)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENj (j=0-31)	j	rw	**Access Enable for Master TAG ID j** This bit enables write access to the module register addresses for transactions with the Master TAG ID j 0_B Write access will not be executed. Read accesses will be executed. 1_B Write and read accesses will be executed

Access Enable Register 1

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). These tags are not used in this system and so no programmable bits are provided.

ACCEN1

Access Enable Register 1 **(10010$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES															
r															

Field	Bits	Type	Description
RES	31:0	r	**Reserved** Read as 0; should be written with 0.

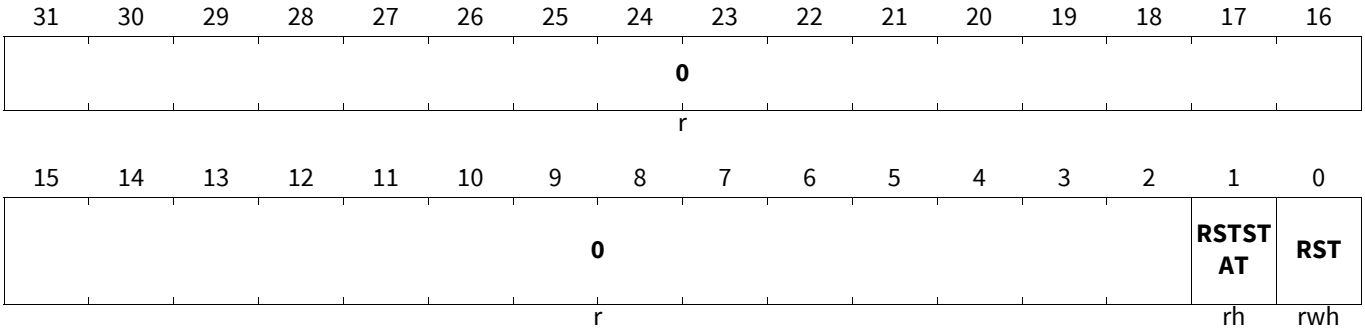
Kernel Reset Register 0

The Kernel Reset function is used to synchronously reset the AHB module. To activate the kernel reset, it is necessary to set the RST bits by writing with 1_B in both Kernel Reset Registers.The RST bit will be re-set by the adapter with the end of the adapter kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to 1_B in the same clock cycle the RST bit is reset. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLE.CLR register bit.

During the execution of the kernel reset until RSTSTAT is set, write accesses to the module registers will result in an error acknowledge. Adapter registers can still be accessed.

KRST0
Kernel Reset Register 0 **(10014_H)** **Application Reset Value: $0000\ 0000_H$**



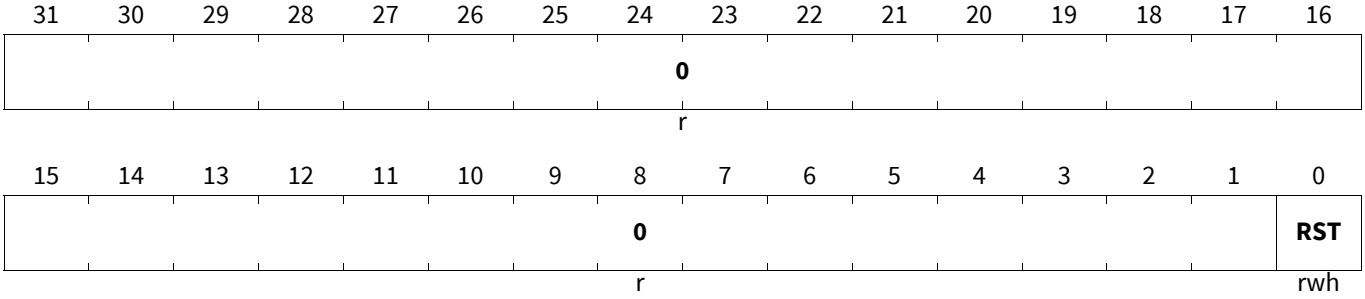
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (reset to 0_B) b after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates whether a kernel reset was executed or not. This bit is set after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1

Kernel Reset Register 1 **(10018_H)** **Application Reset Value: $0000\ 0000_H$**



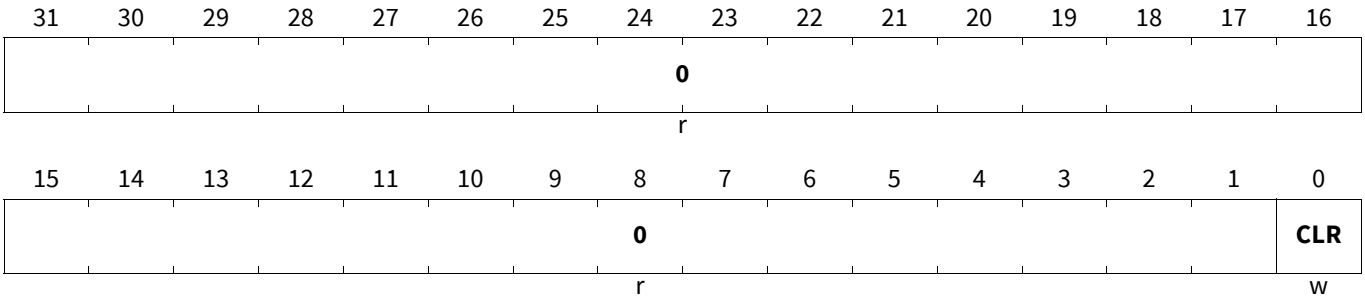
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to 0_B) after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0_B; should be written with 0_B.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **($1001C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0_B. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0_B; should be written with 0_B.

34.4 Registers

This section describes the internal registers of the I2C module. All register names described in this section are also referenced in other parts of the User's Manual by the module name prefix "I2Cm_". For an overview of all internal module registers, see **Section 34.4**.

In the following, the registers of the I2C module are listed. First of all, some explanation on the access conditions is given.

Special I2C Register Access Condition

Besides the general register protection, the I2C module has two main modes that must be considered when programming the peripheral:

- **Configuration Mode:** In this mode the peripheral can be prepared for transmission and reception via the configuration registers, which are only writable in this mode. The peripheral is in the configuration mode when bit RUN is set to 0.
- **Run Mode:** In this mode the peripheral is ready to transmit or receive data. Its configuration registers are locked for write access which will generate a bus error. The peripheral is in the run mode when bit **RUN** is set to 1.

I2C Registers Overview

There are the following blocks of registers (see **Figure 354**):

- Bus Peripheral Interface Registers
- Global Module Control Registers
- FIFO Registers
- Basic Interrupt Registers
- Error Interrupt Source Registers
- Protocol Interrupt Source Registers

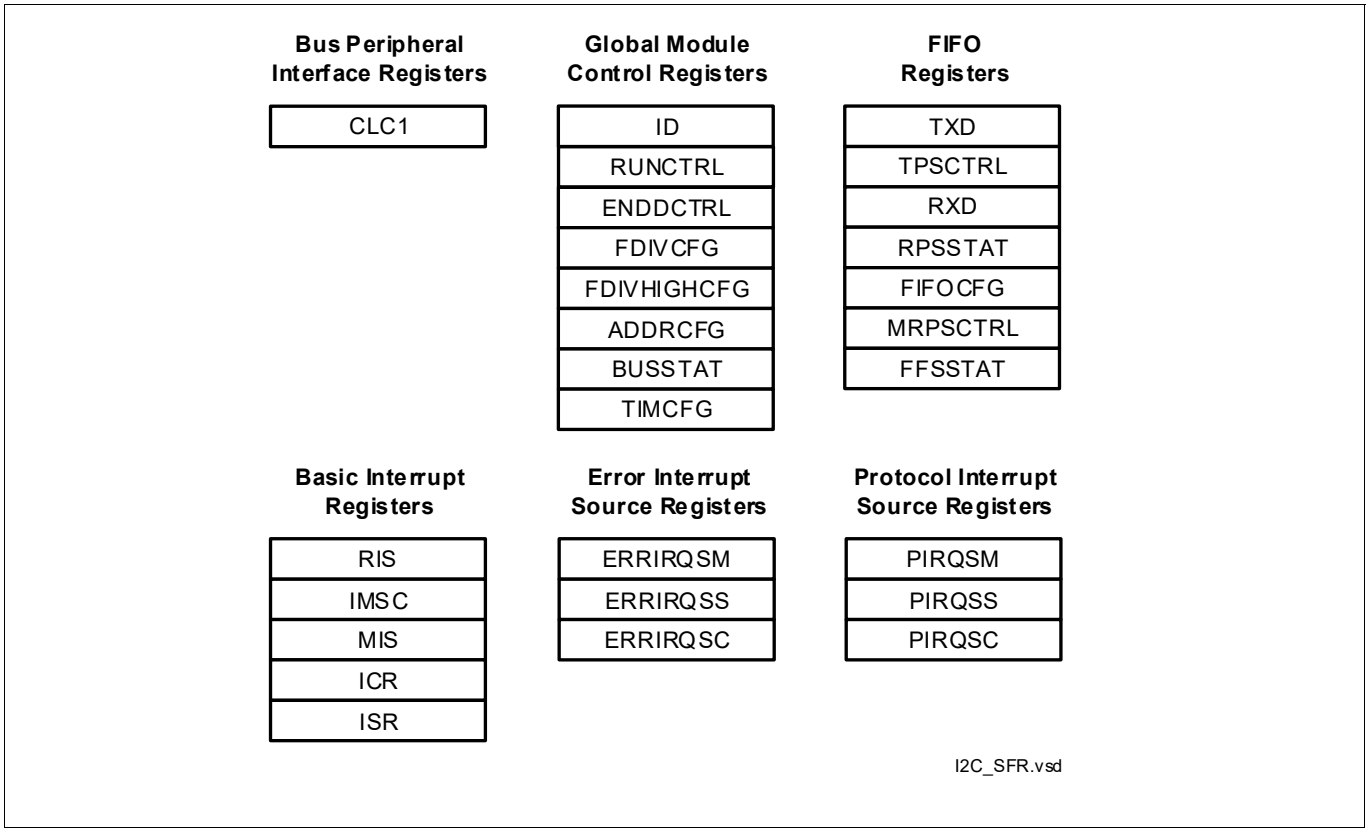


Figure 354 I2C Module Registers

The registers overview in **Table 303** shows the internal register names of the module instances, the offset addresses and the links to the names used in this specification.

Table 303 Register Overview - I2C (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC1	Clock Control 1 Register	00000_H	U,SV	U,SV,P	Application Reset	45
ID	Module Identification Register	00008_H	U,SV	BE	Application Reset	59
RUNCTRL	RUN Control Register	00010_H	U,SV	U,SV,P	Application Reset	59
ENDDCTRL	End Data Control Register	00014_H	U,SV	U,SV,P	Application Reset	60
FDIVCFG	Fractional Divider Configuration Register	00018_H	U,SV	U,SV,P	Application Reset	63
FDIVHIGHCFG	Fractional Divider High-speed Mode Configuration Register	$0001C_H$	U,SV	U,SV,P	Application Reset	64
ADDRCFG	Address Configuration Register	00020_H	U,SV	U,SV,P	Application Reset	61
BUSSTAT	Bus Status Register	00024_H	U,SV	BE	Application Reset	62
FIFOCFG	FIFO Configuration Register	00028_H	U,SV	U,SV,P	Application Reset	69
MRPSCTRL	Maximum Received Packet Size Control Register	$0002C_H$	U,SV	U,SV,P	Application Reset	70
RPSSTAT	Received Packet Size Status Register	00030_H	U,SV	BE	Application Reset	68
TPSCTRL	Transmit Packet Size Control Register	00034_H	U,SV	U,SV,P	Application Reset	67
FFSSTAT	Filled FIFO Stages Status Register	00038_H	U,SV	BE	Application Reset	71
TIMCFG	Timing Configuration Register	00040_H	U,SV	U,SV,P	Application Reset	64
ERRIRQSM	Error Interrupt Request Source Mask Register	00060_H	U,SV	U,SV,P	Application Reset	77
ERRIRQSS	Error Interrupt Request Source Status Register	00064_H	U,SV	BE	Application Reset	77
ERRIRQSC	Error Interrupt Request Source Clear Register	00068_H	U,SV	U,SV,P	Application Reset	78
PIRQSM	Protocol Interrupt Request Source Mask Register	00070_H	U,SV	U,SV,P	Application Reset	80
PIRQSS	Protocol Interrupt Request Source Status Register	00074_H	U,SV	BE	Application Reset	81

Table 303 Register Overview - I2C (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
PIRQSC	Protocol Interrupt Request Source Clear Register	00078_H	U,SV	U,SV,P	Application Reset	82
RIS	Raw Interrupt Status Register	00080_H	U,SV	BE	Application Reset	72
IMSC	Interrupt Mask Control Register	00084_H	U,SV	U,SV,P	Application Reset	73
MIS	Masked Interrupt Status Register	00088_H	U,SV	BE	Application Reset	74
ICR	Interrupt Clear Register	$0008C_H$	U,SV	U,SV,P	Application Reset	75
ISR	Interrupt Set Register	00090_H	U,SV	U,SV,P	Application Reset	75
TXD	Transmission Data Register	08000_H	U,SV	U,SV,P	Application Reset	67
	Reserved ($03FFC_H$ Byte)	08004_H	BE	BE		
RXD	Reception Data Register	$0C000_H$	U,SV	BE	Application Reset	68
	Reserved ($03FFC_H$ Byte)	$0C004_H$	BE	BE		
CLC	Clock Control Register	10000_H	U,SV	SV,E,P	Application Reset	49
MODID	Module Identification Register	10004_H	SV	BE	Application Reset	50
GPCTL	General Purpose Control Register	10008_H	SV	SV,P	Application Reset	51
ACCEN0	Access Enable Register 0	$1000C_H$	SV	SV,SE	Application Reset	51
ACCEN1	Access Enable Register 1	10010_H	SV	SV,SE	Application Reset	52
KRST0	Kernel Reset Register 0	10014_H	SV	SV,E,P	Application Reset	52
KRST1	Kernel Reset Register 1	10018_H	SV	SV,E,P	Application Reset	53
KRSTCLR	Kernel Reset Status Clear Register	$1001C_H$	SV	SV,E,P	Application Reset	54

Notes

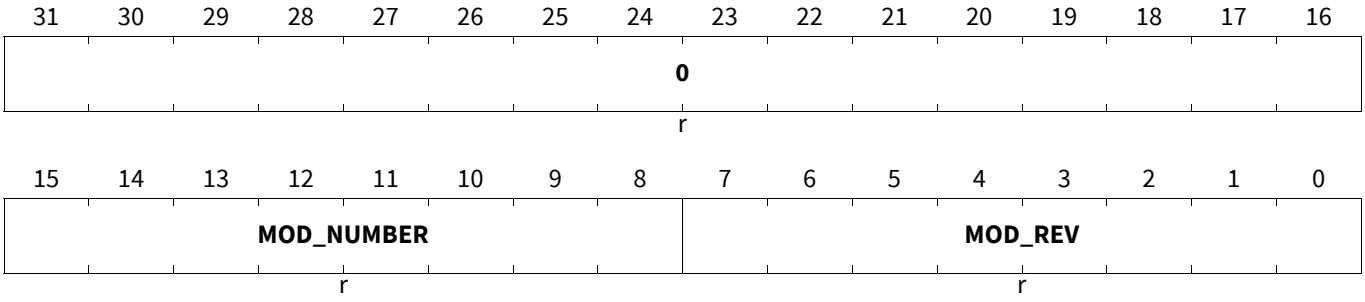
1. *All I2C registers are Application Reset registers.*

34.4.1 Global Module Control Registers

Module Identification Register

This register contains read-only information about the module and its revision.

ID
Module Identification Register **(00008_H)** **Application Reset Value: $0000\ 5705_H$**

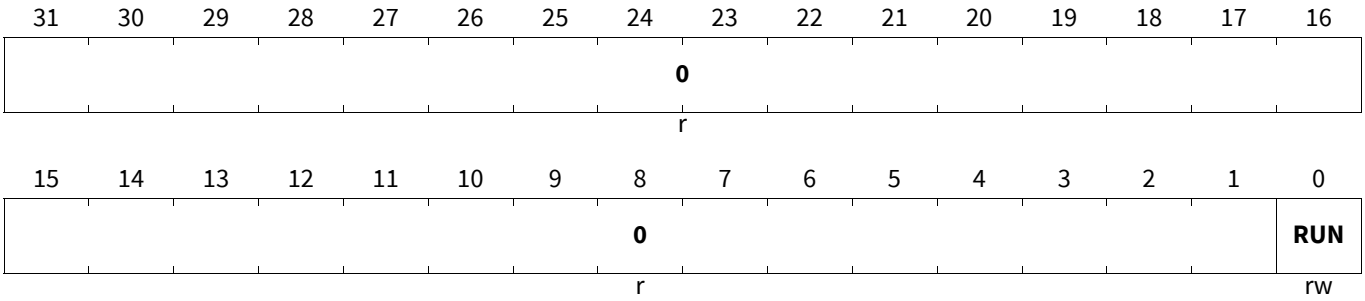


Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision Number** This bit-field defines the revision number.
MOD_NUMBER	15:8	r	**Module Number** This bit-field defines the module identification number.
0	31:16	r	**Reserved** Read as 0; should be written with 0.

RUN Control Register

This register selects configuration mode or run mode.

RUNCTRL
RUN Control Register **(00010_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RUN	0	rw	**Enable I2C-bus Interface** 0_B I2C-bus interface disabled; write access to configuration registers enabled 1_B Participation in I2C-bus communication enabled (if properly configured); write access to configuration registers disabled

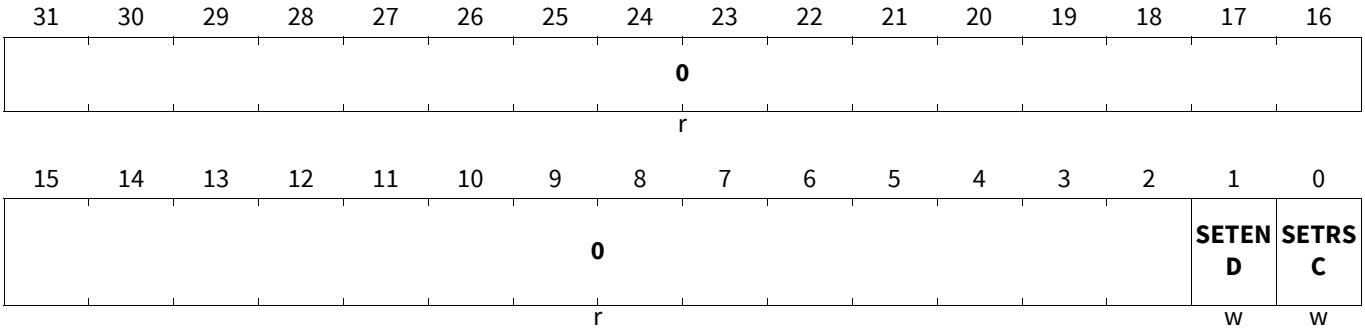
Field	Bits	Type	Description
0	31:1	r	**Reserved** Read as 0; should be written with 0.

End Data Control Register

This register is used to either turn around the data transmission direction or address another slave without sending a stop condition. Also the software can stop the slave-transmitter by sending a not-acknowledge when working as master-receiver or even stop data transmission immediately when operating as master-transmitter. The writing to the bits of this control register is only effective in certain states.

ENDDCTRL

End Data Control Register **(00014_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SETRSC	0	w	**Set Restart Condition** This bit is always read as 0. 0_B Has no effect. 1_B The master wants to restart a data transmission (changing slave/direction). The effect depends on the current state. MASTER RECEIVES BYTES: The master puts a not-acknowledge on the bus and switches to MASTER RESTART state. MASTER TRANSMITS BYTES: After the current byte has been sent, the master switches to MASTER RESTART state.

Field	Bits	Type	Description
SETEND	1	w	**Set End of Transmission** This bit is always read as 0. *Note: Do not write 1 to this bit when bus is free. This will cause an abort after the first byte when a new transfer is started.* 0_B Has no effect. 1_B The effect depends on the current state. MASTER RECEIVES BYTES: After receiving the current byte, the master puts a not-acknowledge on the bus to indicate the transmission end to the slave-transmitter. Next it produces a stop condition on the bus and changes its state to LISTENING. MASTER TRANSMITS BYTES: After sending the current byte and receiving an acknowledge or not-acknowledge from the slave-receiver, the master puts a stop condition on the bus to close the data transmission and changes its state to LISTENING. MASTER RESTART: The master puts a stop condition on the bus to close the data transmission and changes its state to LISTENING. SLAVE RECEIVES BYTES: The slave-receiver puts a not-acknowledge on the bus after the received byte and changes its state to TRANSMISSION FINISHED.
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Address Configuration Register

This configuration register contains the I2C-address (when addressed as a slave) and some bits that control the basic operation of the peripheral.

ADDRCFG
Address Configuration Register **(00020_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										SOPE	SONA	MnS	MCE	GCE	TBAM
r										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						ADR									
r						rw									

Field	Bits	Type	Description
ADR	9:0	rw	**I2C-bus Device Address** This bit-field determines the address of the device when addressed as a slave. (Watch out for reserved addresses by referring to I2C-bus spec V2.1.) Depending on setting of TBAM, this is either a 7-bit address (bits [7:1]) or a 10-bit address (bits [9:0]).

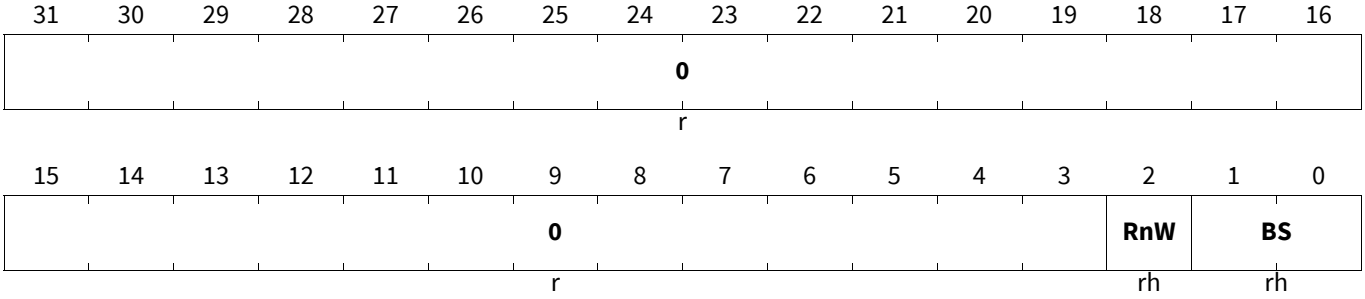
Field	Bits	Type	Description
TBAM	16	rw	**Ten Bit Address Mode** *Note: When this bit is zero, only bits 7 down to 1 of the ADR field are valid.* 0_B 7-bit address mode enabled. 1_B 10-bit address mode enabled.
GCE	17	rw	**General Call Enable** 0_B Ignore general call occurrence. 1_B Enable general call detection; when detected, an acknowledge will be put on the bus
MCE	18	rw	**Master Code Enable** 0_B Device is not able to get along with high-speed mode 1_B Device is able to handle master code
MnS	19	rw	**Master / not Slave** 0_B Peripheral is configured as slave 1_B Peripheral is configured as master
SONA	20	rw	**Stop on Not-acknowledge** *Note: After successful transmission of a master code (during high-speed mode) SONA is not considered till a stop condition is manually generated by SETEND.* 0_B Device changes to MASTER RESTART state. 1_B Device puts a stop condition on the bus and changes to LISTENING state.
SOPE	21	rw	**Stop on Packet End** **Notes** *1. This bit-field should be used only in Master Mode. In slave modeshould always be 0.* *2. If device works as receiver a not-acknowledge is always generatedon package end.* *3. After successful transmission of a master code (during high-speedmode) SOPE is not considered till a stop condition is manually generated by SETEND.* 0_B Device enters MASTER RESTART state when the data packet end is indicated by the FIFO. 1_B Device puts a stop condition on the bus when the data packet end is indicated by the FIFO and changes to MASTER LISTENING state.
0	15:10, 31:22	r	**Reserved** Read as 0; should be written with 0.

Bus Status Register

This register contains status information of the I2C-bus. This additional information can be used by software to start appropriate actions.

BUSSTAT

Bus Status Register **(00024$_H$)** **Application Reset Value: 0000 0000$_H$**



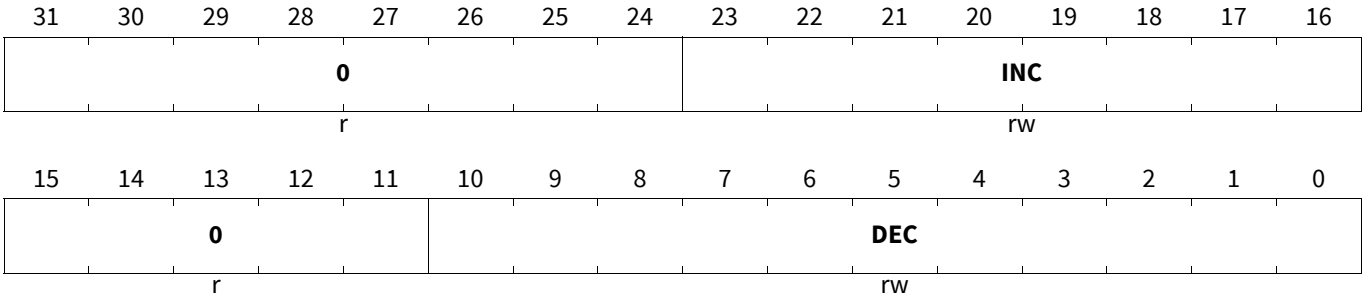
Field	Bits	Type	Description
BS	1:0	rh	**Bus Status** Shows the current status on the I2C-bus. 00_B I2C-bus is free (no start condition detected). 01_B A start condition has been detected on the bus (bus busy). 10_B The device is working as master and has claimed the control on the I2C-bus (busy master). 11_B A remote master has accessed this device as slave.
RnW	2	rh	**Read/not Write** Set by hardware automatically after address byte has been sent/received. 0_B Working as transmitter (Write to I2C-bus). 1_B Working as receiver (Read from I2C-bus).
0	31:3	r	**Reserved** Read as 0; should be written with 0.

Fractional Divider Configuration Register

This configuration register is used to program the fractional divider of the I2C-bus for standard and fast mode. Before the peripheral is switched on by setting the RUN bit, the register should be configured.

FDIVCFG

Fractional Divider Configuration Register **(00018$_H$)** **Application Reset Value: 0000 0000$_H$**



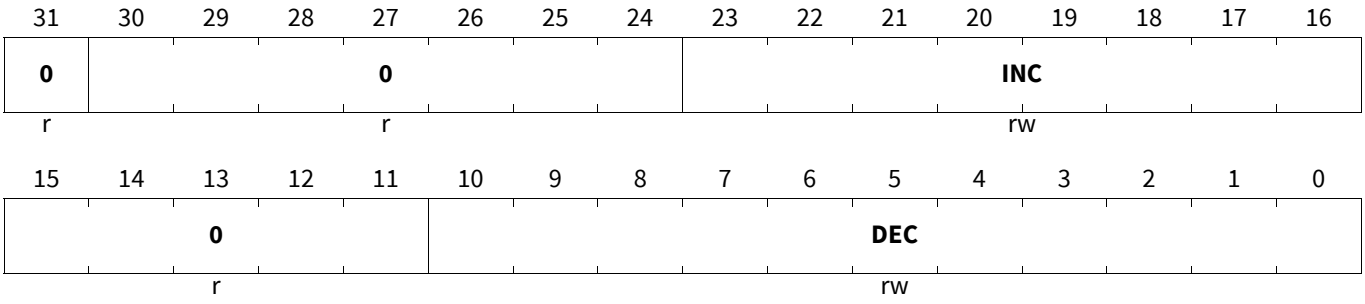
Field	Bits	Type	Description
DEC	10:0	rw	**Decrement Value of Fractional Divider** For standard/fast mode, see **Clock and Timing Control**.

Field	Bits	Type	Description
INC	23:16	rw	**Increment Value of Fractional Divider** For standard/fast mode, see **Clock and Timing Control**.
0	15:11, 31:24	r	**Reserved** Read as 0; should be written with 0.

Fractional Divider High-speed Mode Configuration Register

This configuration register is used to program the fractional divider of the I2C-bus for high-speed mode. Before the peripheral is switched on by setting the RUN bit, the register should be configured if high-speed mode is used.

FDIVHIGHCFG
Fractional Divider High-speed Mode Configuration Register($0001C_H$) Application Reset Value: $0000\ 0000_H$



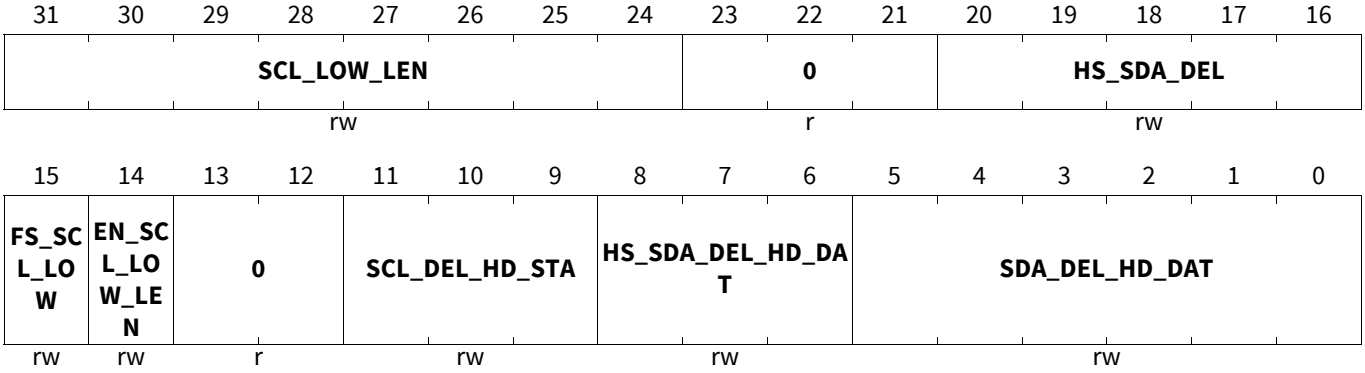
Field	Bits	Type	Description
DEC	10:0	rw	**Decrement Value of Fractional Divider** For high-speed mode, see **Clock and Timing Control**.
INC	23:16	rw	**Increment Value of Fractional Divider** For high-speed mode, see **Clock and Timing Control**.
0	15:11, 30:24, 31	r	**Reserved** Read as 0; should be written with 0.

Timing Configuration Register

This configuration register adjusts some timings of the I2C-bus signals SCL and SCA. The delays are given in kernel_clk cycles (denoted as stages below).

The delayed stages may have +/- 1 stage deviation.

TIMCFG
Timing Configuration Register **(00040_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SDA_DEL_HD_DAT	5:0	rw	**SDA Delay Stages for Data Hold Time in Standard and Fast modes** SDA delay stages for data hold time in standard and fast modes. *Note: SDA delay from SCL falling edge but will also affect SDA Setup time relative to next SCL rising edge* 00_H 3 stages delay … $3F_H$ 66 stages delay
HS_SDA_DEL_HD_DAT	8:6	rw	**SDA Delay Stages for Data Hold Time in High-speed Mode** SDA delay stages for data hold time in HS mode. *Note: SDA delay from SCL falling edge but will also affect SDA Setup time relative to next SCL rising edge* 000_B 3 stages delay … 111_B 10 stages delay
SCL_DEL_HD_STA	11:9	rw	**SCL Delay Stages for Hold Time Start (Restart) Bit** 000_B 2 stages delay … 111_B 9 stages delay
EN_SCL_LOW_LEN	14	rw	**Enable Direct Configuration of SCL Low Period Length in Fast Mode** 0_B SCL low period is a fixed part of the whole period, as defined by FS_SCL_LOW 1_B SCL low period is determined by the setting of SCL_LOW_LEN

Field	Bits	Type	Description
FS_SCL_LOW	15	rw	**Set Fast Mode SCL Low Period Timing** The internal duration of the SCL low time with respect to the period length as defined by the baudrate setting, can be enlarged for the Fast Speed Mode, in order to meet the asymmetric duty cycle requirements from the standard. The detailed formulas are given in the functional specification. 0_B Standard mode SCL low period timing. For INC = 1 it is 5/8 of period. 1_B Fast mode SCL low period timing. For INC = 1 it is 6/8 of period.
HS_SDA_DEL	20:16	rw	**SDA Delay Stages for Start/Stop bit in High-speed Mode** 00_H 3 stages delay … 07_H 10 stages delay
SCL_LOW_LEN	31:24	rw	**SCL Low Length in Fast Mode** If enabled by EN_SCL_LOW_LEN setting, this field determines the extension of the SCL low time. In case of INC = 1, the low time is extended by the number of kernel_clk cycles. In general, there is a more complex formula, as given in the functional specification. The total period time is not changed, i.e., the SCL high period is reduced accordingly. Setting SCL low time to period length or higher is not supported and would lead to unpredictable results.
0	13:12, 23:21	r	**Reserved** Read as 0; should be written with 0.

34.4.2 FIFO Registers

Transmission Data Register

The software has to write the characters to be transmitted into this register.

A larger address range (8000_H to $BFFC_H$) is reserved for the FIFO. Accessing any address in the defined range has the same effect as accessing the first address.

A read access to TXD register is not possible, it will return 0 in all cases. Reading has no effect on the FIFO

When using byte or half word access from the bus, the TX FIFO pointer will only be increased, if one of the following conditions is fulfilled:

- The most significant byte or half word of the FIFO stage is written
- The packet end is reached

TXD
Transmission Data Register (08000_H) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXD															
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXD															
w															

Field	Bits	Type	Description
TXD	31:0	w	**Transmission Data** Characters to be transmitted

Transmit Packet Size Control Register

This register is used to indicate the peripheral the size of the packet to be transmitted. Writing the packet size to this register if the FIFO controller is configured for flow controller mode initiates the data requests (BREQ, SREQ, ...).Writing to this register in configuration state has no impact.

TPSCTRL
Transmit Packet Size Control Register (00034_H) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		TPS													
r		rwh													

Field	Bits	Type	Description
TPS	13:0	rwh	**Transmit Packet Size** Length in characters of the transmit packet, write value range: 1 to 16383 Reading returns the written value as long as it is not loaded to an internal counter. After that, reading returns 0 and a new value can be written.
0	31:14	r	**Reserved** Read as 0; should be written with 0.

Reception Data Register

The software can read the received characters from this register.

A larger address range ($C000_H$ to $FFFC_H$) is reserved for the FIFO . Reading from any address in the defined range has the same effect as reading from the first address.

RXD

Reception Data Register **($0C000_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXD															
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXD															
rh															

Field	Bits	Type	Description
RXD	31:0	rh	**Reception Data** Received characters

Received Packet Size Status Register

This register indicates the size of the received data packet to the software. The software should read this register after the last request of a packet.

RPSSTAT

Received Packet Size Status Register **(00030_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		RPS													
r		rh													

Field	Bits	Type	Description
RPS	13:0	rh	**Received Packet Size** Length in characters of the received packet (0 to 16383)
0	31:14	r	**Reserved** Read as 0; should be written with 0.

FIFO Configuration Register

This configuration register is used to set up the FIFO before the peripheral is enabled and data is received or transmitted.

FIFOCFG
FIFO Configuration Register **(00028$_H$)** **Application Reset Value: 0000 0022$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													CRBC	TXFC	RXFC
r													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		TXFA		0		RXFA		0		TXBS		0		RXBS	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
RXBS	1:0	rw	**RX Burst Size** 00_B 1 word 01_B 2 words 10_B 4 words 11_B Do not use this combination
TXBS	5:4	rw	**TX Burst Size** 00_B 1 word 01_B 2 words 10_B 4 words 11_B Do not use this combination
RXFA	9:8	rw	**RX FIFO Alignment** Use byte alignment wherever it is possible. 00_B Byte aligned (character alignment) 01_B Half word aligned (character alignment of two characters) 10_B Word aligned (character alignment of four characters) 11_B Do not use this combination

Field	Bits	Type	Description
TXFA	13:12	rw	**TX FIFO Alignment** Use byte alignment wherever it is possible. 00_B Byte aligned (character alignment) 01_B Half word aligned (character alignment of two characters) 10_B Word aligned (character alignment of four characters) 11_B Do not use this combination
RXFC	16	rw	**RX FIFO Flow Control** 0_B RX FIFO not as flow controller 1_B RX FIFO as flow controller
TXFC	17	rw	**TX FIFO Flow Control** 0_B TX FIFO not as flow controller 1_B TX FIFO as flow controller
CRBC	18	rw	**Clear Request Behavior Configuration** Used to configure the clear request behavior for the FIFO data request. Can only be used for single request and must be set to “0” when burst accesses are used in the system (eg. when TX/RXBS > 0) 0_B Data request is cleared by Software. 1_B Data request is cleared automatically when Write/Read access to FIFO occurs.
0	3:2, 7:6, 11:10, 15:14, 31:19	r	**Reserved** Read as 0; should be written with 0.

Maximum Received Packet Size Control Register

This register is used to limit the received packet size. The register value may be changed in any state of the FIFO.

MRPSCTRL

Maximum Received Packet Size Control Register($0002C_H$) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		MRPS													
r		rwh													

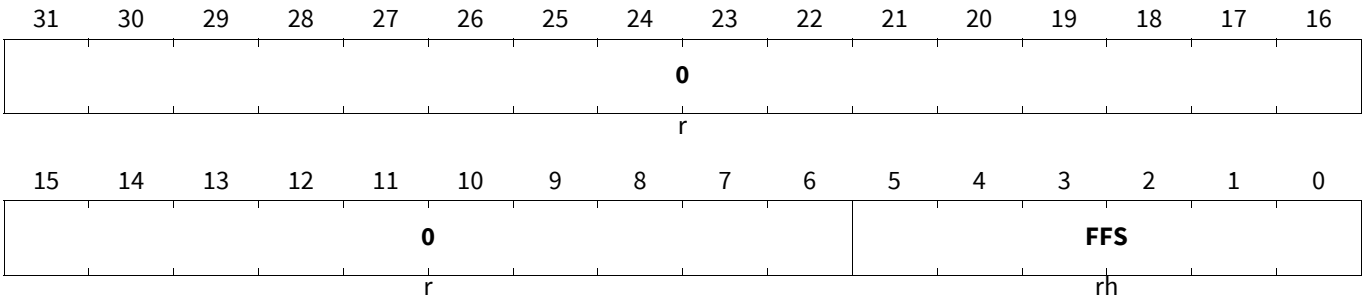
Field	Bits	Type	Description
MRPS	13:0	rwh	**Maximum Received Packet Size** Length in characters of packet to be received; write value range: 0 (unlimited size) to 16383 Reading returns the written value as long as the previous packet has not been read completely from the FIFO. After that, MRPS is loaded to an internal register, reading returns 0 and a new value can be written.
0	31:14	r	**Reserved** Read as 0; should be written with 0.

Filled FIFO Stages Status Register

This register is used to indicate the number of filled FIFO stages.

FFSTAT

Filled FIFO Stages Status Register **(00038_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FFS	5:0	rh	**Filled FIFO Stages** Number of filled FIFO stages (0 to 8)
0	31:6	r	**Reserved** Read as 0; should be written with 0.

34.4.3 Basic Interrupt Registers

For an overview of the Service Request Block (SRB) see **Section 34.3.1.7**.

Raw Interrupt Status Register

This read-only register returns the current raw status value (without reflecting the mask) of the interrupt request sources. One status bit is provided for each request. A write to this register has no effect. The status bits are set by hardware or software (via register **ISR**) and can be cleared by software (via register **ICR**).

RIS

Raw Interrupt Status Register **(00080_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										I2C_P_INT	I2C_ERR_INT	BREQ_INT	LBREQ_INT	SREQ_INT	LSREQ_INT
r										rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
LSREQ_INT	0	rh	**Last Single Request Interrupt** 0_B No interrupt request 1_B Interrupt request pending
SREQ_INT	1	rh	**Single Request Interrupt** 0_B No interrupt request 1_B Interrupt request pending
LBREQ_INT	2	rh	**Last Burst Request Interrupt** 0_B No interrupt request 1_B Interrupt request pending
BREQ_INT	3	rh	**Burst Request Interrupt** 0_B No interrupt request 1_B Interrupt request pending
I2C_ERR_INT	4	rh	**I2C Error Interrupt** This is the combined bit for indication of FIFO errors due to overflow and underflow. 0_B No interrupt request 1_B Interrupt request pending

Field	Bits	Type	Description
I2C_P_INT	5	rh	**I2C Protocol Interrupt** This is the combined bit for indication of a protocol event in the I2C kernel. 0_B No interrupt request 1_B Interrupt request pending
0	31:6	r	**Reserved** Read as 0; should be written with 0.

Interrupt Mask Control Register

A write of 1 to a particular bit of this register enables the corresponding interrupt request; a write of 0 disables it. A read to this register returns the current mask bits. After reset all requests are disabled.

IMSC
Interrupt Mask Control Register **(00084_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										I2C_P_INT	I2C_ERR_INT	BREQ_INT	LBREQ_INT	SREQ_INT	LSREQ_INT
r										rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
LSREQ_INT	0	rw	**Last Single Request Interrupt** 0_B Interrupt request disabled 1_B Interrupt request enabled
SREQ_INT	1	rw	**Single Request Interrupt** 0_B Interrupt request disabled 1_B Interrupt request enabled
LBREQ_INT	2	rw	**Last Burst Request Interrupt** 0_B Interrupt request disabled 1_B Interrupt request enabled
BREQ_INT	3	rw	**Burst Request Interrupt** 0_B Interrupt request disabled 1_B Interrupt request enabled
I2C_ERR_INT	4	rw	**I2C Error Interrupt** 0_B Interrupt request disabled 1_B Interrupt request enabled
I2C_P_INT	5	rw	**I2C Protocol Interrupt** 0_B Interrupt request disabled 1_B Interrupt request enabled

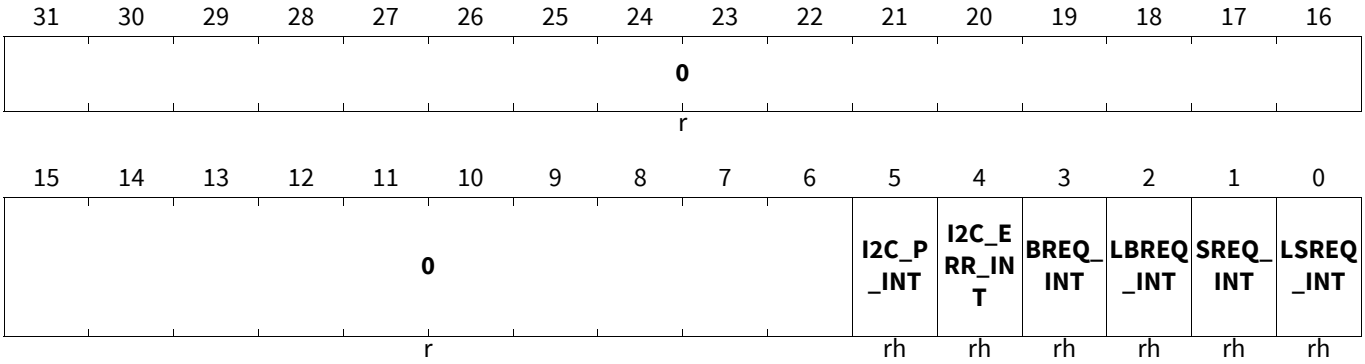
Field	Bits	Type	Description
0	31:6	r	**Reserved** Read as 0; should be written with 0.

Masked Interrupt Status Register

This read-only register returns the masked status value (derived from registers **RIS** and **IMSC**) of the corresponding interrupt requests. A write to this register has no effect.

MIS

Masked Interrupt Status Register **(00088_H)** **Application Reset Value: $0000\ 0000_H$**

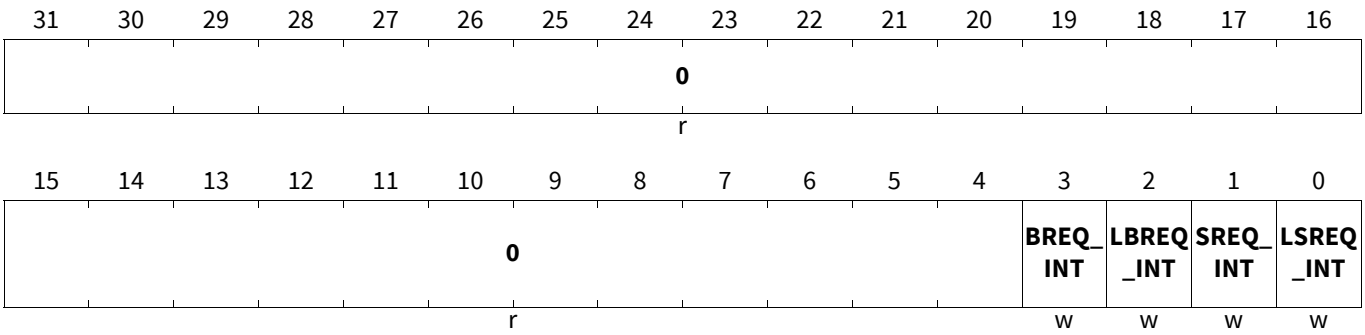


Field	Bits	Type	Description
LSREQ_INT	0	rh	**Last Single Request Interrupt** 0_B No interrupt request 1_B Interrupt request pending
SREQ_INT	1	rh	**Single Request Interrupt** 0_B No interrupt request 1_B Interrupt request pending
LBREQ_INT	2	rh	**Last Burst Request Interrupt** 0_B No interrupt request 1_B Interrupt request pending
BREQ_INT	3	rh	**Burst Request Interrupt** 0_B No interrupt request 1_B Interrupt request pending
I2C_ERR_INT	4	rh	**I2C Error Interrupt** This is the combined bit for indication of FIFO errors due to overflow and underflow. 0_B No interrupt request 1_B Interrupt request pending
I2C_P_INT	5	rh	**I2C Protocol Interrupt** This is the combined bit for indication of a protocol event in the I2C kernel. 0_B No interrupt request 1_B Interrupt request pending
0	31:6	r	**Reserved** Read as 0; should be written with 0.

Interrupt Clear Register

On a write of 1 to a particular bit of this write-only register, the corresponding interrupt request is cleared; a write of 0 has no effect. Reading the register returns 0.

ICR
Interrupt Clear Register **(0008C$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
LSREQ_INT	0	w	**Last Single Request Interrupt** 0_B No change 1_B Clear interrupt request
SREQ_INT	1	w	**Single Request Interrupt** 0_B No change 1_B Clear interrupt request
LBREQ_INT	2	w	**Last Burst Request Interrupt** 0_B No change 1_B Clear interrupt request
BREQ_INT	3	w	**Burst Request Interrupt** 0_B No change 1_B Clear interrupt request
0	31:4	r	**Reserved** Read as 0; should be written with 0.

Interrupt Set Register

On a write of 1 to a particular bit of this write-only register, the corresponding interrupt request is set; a write of 0 has no effect. Reading the register returns 0.

ISR

Interrupt Set Register **(00090$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										I2C_P_INT	I2C_ERR_INT	BREQ_INT	LBREQ_INT	SREQ_INT	LSREQ_INT
r										w	w	w	w	w	w

Field	Bits	Type	Description
LSREQ_INT	0	w	**Last Single Request Interrupt** 0_B No change 1_B Set interrupt request
SREQ_INT	1	w	**Single Request Interrupt** 0_B No change 1_B Set interrupt request
LBREQ_INT	2	w	**Last Burst Request Interrupt** 0_B No change 1_B Set interrupt request
BREQ_INT	3	w	**Burst Request Interrupt** 0_B No change 1_B Set interrupt request
I2C_ERR_INT	4	w	**I2C Error Interrupt** 0_B No change 1_B Set interrupt request
I2C_P_INT	5	w	**I2C Protocol Interrupt** 0_B No change 1_B Set interrupt request
0	31:6	r	**Reserved** Read as 0; should be written with 0.

34.4.4 Error Interrupt Source Registers

For an overview of the source register operation see **Section 34.3.1.7.2**.

Error Interrupt Request Source Mask Register

A write of 1 to a particular bit of this register enables the corresponding error interrupt request source; a write of 0 disables it. A read to this register returns the current mask bits. After reset all sources are enabled.

The interrupts are explained in detail in description of register **ERRIRQSS**.

ERRIRQSM

Error Interrupt Request Source Mask Register (00060$_H$) **Application Reset Value: 0000 000F$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												TXF_OFL	TXF_UFL	RXF_OFL	RXF_UFL
r												rw	rw	rw	rw

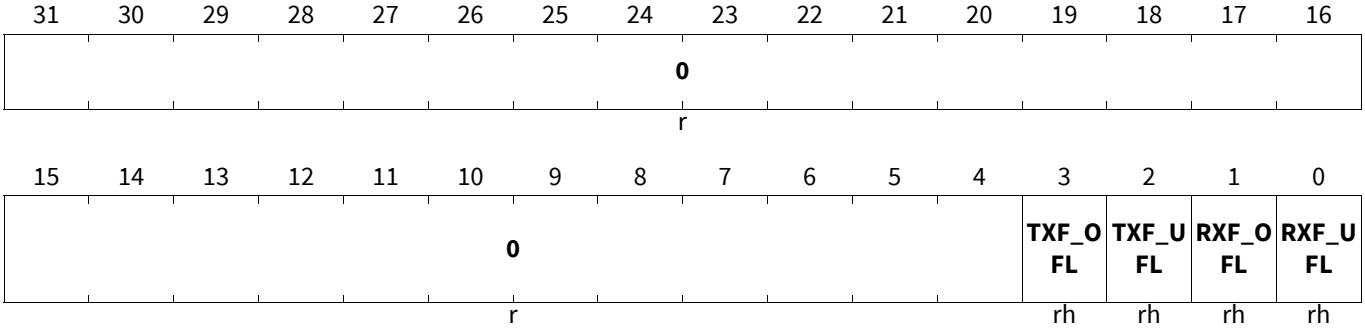
Field	Bits	Type	Description
RXF_UFL	0	rw	**RX FIFO Underflow** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
RXF_OFL	1	rw	**RX FIFO Overflow** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
TXF_UFL	2	rw	**TX FIFO Underflow** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
TXF_OFL	3	rw	**TX FIFO Overflow** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
0	31:4	r	**Reserved** Read as 0; should be written with 0.

Error Interrupt Request Source Status Register

This read-only register returns the current raw status value (without reflecting the mask) of the error interrupt request sources. A write to this register has no effect. The error status bits are set by hardware and can be cleared by software (via register **ERRIRQSC**).

ERRIRQSS

Error Interrupt Request Source Status Register (00064$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RXF_UFL	0	rh	**RX FIFO Underflow** The FIFO has detected an RX FIFO underflow. 0_B No interrupt request 1_B Interrupt request pending
RXF_OFL	1	rh	**RX FIFO Overflow** The I2C kernel has detected a RX FIFO overflow 0_B No interrupt request 1_B Interrupt request pending
TXF_UFL	2	rh	**TX FIFO Underflow** The I2C kernel has detected a TX FIFO underflow. 0_B No interrupt request 1_B Interrupt request pending
TXF_OFL	3	rh	**TX FIFO Overflow** The FIFO has detected a TX FIFO overflow. 0_B No interrupt request 1_B Interrupt request pending
0	31:4	r	**Reserved** Read as 0; should be written with 0.

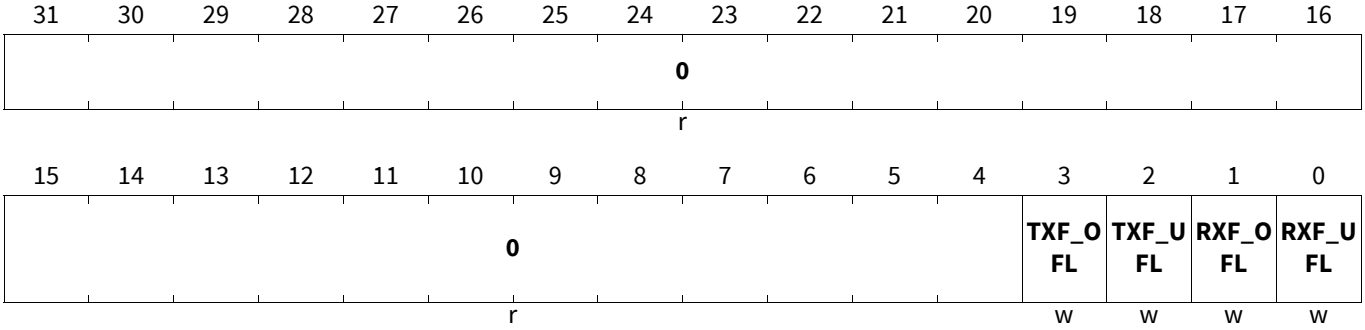
Error Interrupt Request Source Clear Register

On a write of 1 to a particular bit of this write-only register, the corresponding error interrupt request source is cleared and if no further error interrupt request sources are active, the whole error interrupt is cleared. If the corresponding bit is set by SW via the **ISR** register, then it can be cleared by setting any non-reserved bit of the interrupt request source clear register**.** A write of 0 has no effect. Reading the register returns 0.

The interrupts are explained in detail in description of register **ERRIRQSS**.

ERRIRQSC

Error Interrupt Request Source Clear Register (00068$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RXF_UFL	0	w	**RX FIFO Underflow** 0_B No change 1_B Clear interrupt request source
RXF_OFL	1	w	**RX FIFO Overflow** 0_B No change 1_B Clear interrupt request source
TXF_UFL	2	w	**TX FIFO Underflow** 0_B No change 1_B Clear interrupt request source
TXF_OFL	3	w	**TX FIFO Overflow** 0_B No change 1_B Clear interrupt request source
0	31:4	r	**Reserved** Read as 0; should be written with 0.

34.4.5 Protocol Interrupt Source Registers

For an overview of the source register operation see **Section 34.3.1.7.2**.

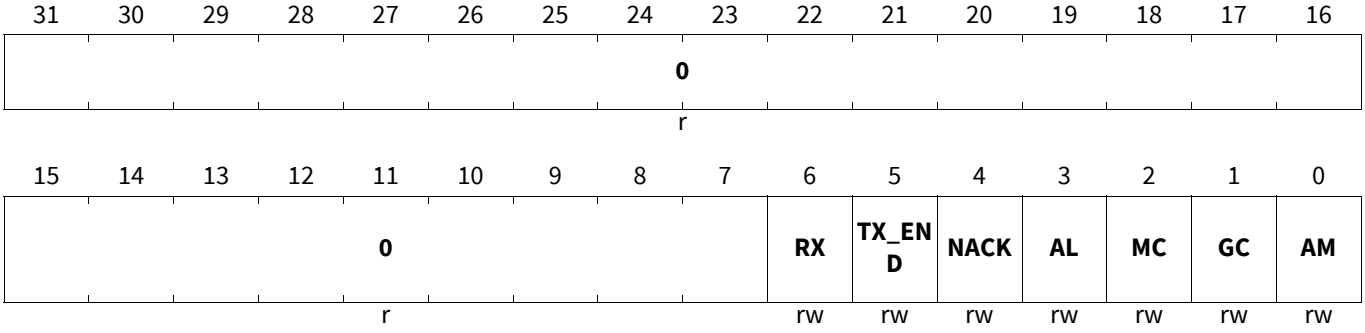
Protocol Interrupt Request Source Mask Register

A write of 1 to a particular bit of this register enables the corresponding protocol interrupt request source; a write of 0 disables it. A read to this register returns the current mask bits. After reset all sources are enabled.

The interrupts are explained in detail in description of register **PIRQSS**.

PIRQSM

Protocol Interrupt Request Source Mask Register(00070_H) **Application Reset Value: $0000\ 007F_H$**



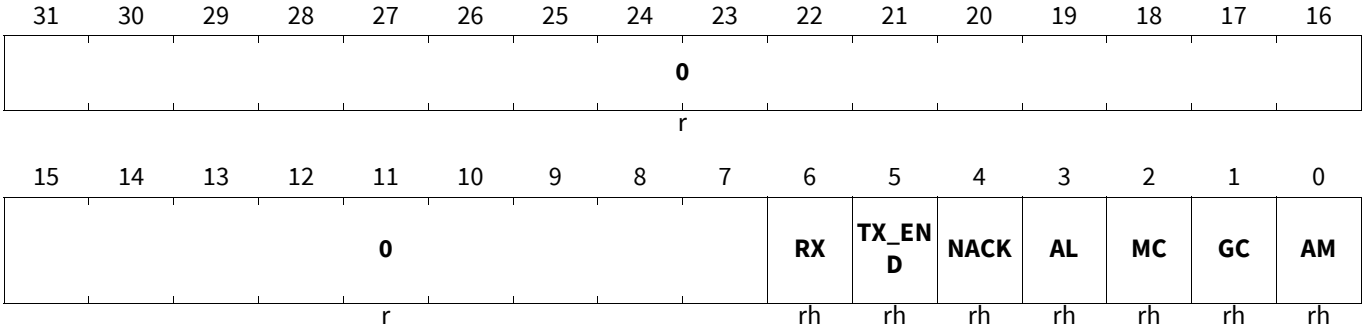
Field	Bits	Type	Description
AM	0	rw	**Address Match** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
GC	1	rw	**General Call** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
MC	2	rw	**Master Code** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
AL	3	rw	**Arbitration Lost** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
NACK	4	rw	**Not-acknowledge Received** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
TX_END	5	rw	**Transmission End** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
RX	6	rw	**Receive Mode** 0_B Interrupt request source disabled 1_B Interrupt request source enabled
0	31:7	r	**Reserved** Read as 0; should be written with 0.

Protocol Interrupt Request Source Status Register

This read-only register returns the current raw status value (without reflecting the mask) of the protocol interrupt request sources. A write to this register has no effect. The protocol interrupt status bits are set by hardware and can be cleared by software (via register **PIRQSC**).

PIRQSS

Protocol Interrupt Request Source Status Register(00074_H) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
AM	0	rh	**Address Match** Device (in master or slave mode) is addressed by remote master (matching device address). Accordingly, bit-field BS in register **BUSSTAT** is set to 11_B. 0_B No interrupt request 1_B Interrupt request pending
GC	1	rh	**General Call** Remote master has transmitted a general call. 0_B No interrupt request 1_B Interrupt request pending
MC	2	rh	**Master Code** Remote master has transmitted a master call. 0_B No interrupt request 1_B Interrupt request pending
AL	3	rh	**Arbitration Lost** Device (master mode) lost the control on the I2C-bus due to losing arbitration procedure. Accordingly, bit-field BS in register **BUSSTAT** is set to 01_B. 0_B No interrupt request 1_B Interrupt request pending
NACK	4	rh	**Not-acknowledge Received** When working as transmitter this interrupt indicates a not-acknowledge from the remote receiver. The SW has to decide what further steps have to be taken. 0_B No interrupt request 1_B Interrupt request pending

Field	Bits	Type	Description
TX_END	5	rh	**Transmission End** The device has ended the data transfer properly (after stop condition has been put on the bus or the MASTER RESTART state has been entered.) 0_B No interrupt request 1_B Interrupt request pending
RX	6	rh	**Receive Mode** I2C kernel indicates switching from transmitting data to receiving data. 0_B No interrupt request 1_B Interrupt request pending
0	31:7	r	**Reserved** Read as 0; should be written with 0.

Protocol Interrupt Request Source Clear Register

On a write of 1 to a particular bit of this write-only register, the corresponding protocol interrupt request source is cleared and if no further protocol interrupt request sources are active, the whole protocol interrupt is cleared. If the corresponding **RIS** bit is set by SW via the **ISR** register, then it can be cleared by setting any non-reserved bit of the interrupt request source clear register**.** A write of 0 has no effect. Reading the register returns 0.

The interrupts are explained in detail in description of register **PIRQSS**.

PIRQSC

Protocol Interrupt Request Source Clear Register(00078_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									RX	TX_END	NACK	AL	MC	GC	AM
r									w	w	w	w	w	w	w

Field	Bits	Type	Description
AM	0	w	**Address Match** 0_B No change 1_B Clear Interrupt source
GC	1	w	**General Call** 0_B No change 1_B Clear Interrupt source
MC	2	w	**Master Code** 0_B No change 1_B Clear Interrupt source

Field	Bits	Type	Description
AL	3	w	**Arbitration Lost** 0_B No change 1_B Clear Interrupt source
NACK	4	w	**Not-acknowledge Received** 0_B No change 1_B Clear Interrupt source
TX_END	5	w	**Transmission End** 0_B No change 1_B Clear Interrupt source
RX	6	w	**Receive Mode** 0_B No change 1_B Clear Interrupt source
0	31:7	r	**Reserved** Read as 0; should be written with 0.

34.5 Safety Measures

Description of safety mechanisms and conditions of use.

34.6 IO Interfaces

The table below lists all the interfaces of the I2C to other modules in the device.

Table 304 List of I2C Interface Signals

Interface Signals	I/O	Description
i2c_sfr		**FPI slave interface**
sx_irq_i2c		
DTR_INT	out	**I2C Data Transfer Request**
ERR_INT	out	**I2C Error Service Request**
P_INT	out	**I2C Kernel Service Request**
SDAA	in	**Serial Data Input 0** The last letter indicates the input of the input multiplexer.
SDAB	in	**Serial Data Input 1** The last letter indicates the input of the input multiplexer.
SDAC	in	**Serial Data Input 2** The last letter indicates the input of the input multiplexer.
SDAD	in	**Serial Data Input 3** The last letter indicates the input of the input multiplexer.
SDAE	in	**Serial Data Input 4** The last letter indicates the input of the input multiplexer.

Table 304 List of I2C Interface Signals (cont'd)

Interface Signals	I/O	Description
SDAF	in	**Serial Data Input 5** The last letter indicates the input of the input multiplexer.
SDAG	in	**Serial Data Input 6** The last letter indicates the input of the input multiplexer.
SDAH	in	**Serial Data Input 7** The last letter indicates the input of the input multiplexer.
SCL	out	**Serial Clock Output**
SDA	out	**Serial Data Output**
HS_MODE	out	**High-Speed-Mode Active**
HS_PAD_SWITCH	out	**Control Signal for High Speed Pad**
SLEEP	in	**Sleep Request**

34.7 Revision History

Table 305 Revision History

Reference	Change to Previous Version	Comment
V2.3.4		
–	No changes.	–
V2.3.5		
Page 51	3 questionmarks ("?") in footnote of register "Access Enable Register 0" removed.	
V2.3.6		
Page 9	Replaced list item.	
Page 59	Changed Reset value to 5705 hex for **Module Identification Register**.	

35 High Speed Serial Link (HSSL)

The HSSL module provides point-to-point communication of both single data values and of large data blocks, called streams. The communicating devices can be complex microcontrollers, or a microcontroller and a device with only basic execution capabilities. There are four channels to transfer single values to/from target. They support direct writing of 8/16/32 bit data from the initiator into a target's register, as well as reading a value from a target, performed by a modules internal master on the target side. For transferring large data blocks there is a channel containing FIFOs. The HSSL module implements Transport Layer tasks and hands over the data to another module which provides Data Link Layer and Physical Layer services, data serialization and transmission. All transfers are protected by safety features like CRC and timeout.

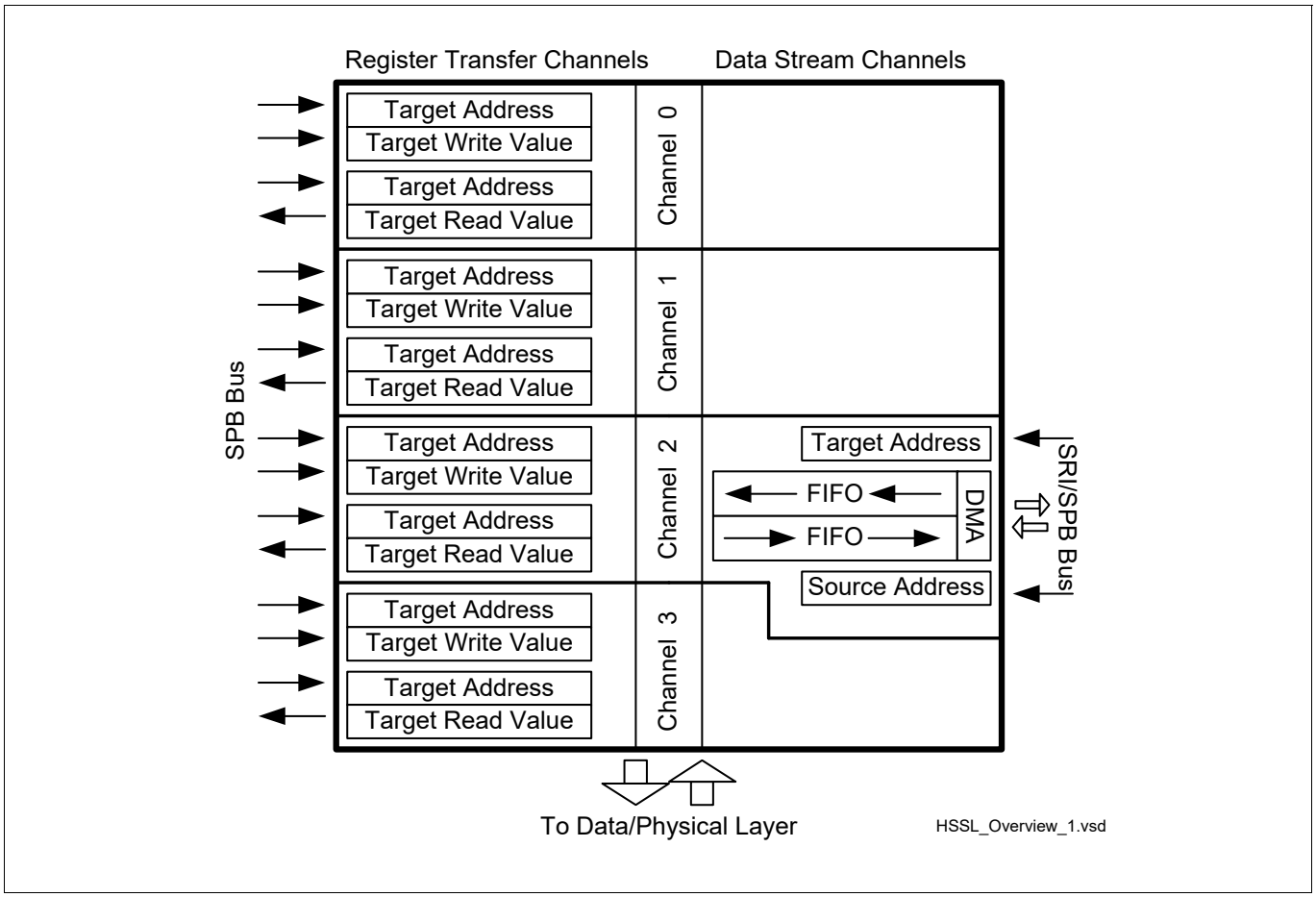


Figure 355 Block Diagram HSSL

35.1 Feature List

- Connects microcontroller to microcontroller or to any digital device
- Writing a single 8 / 16 / 32 bit data value into the register of a target device
- Reading single data from an 8 / 16 / 32 bit register of a target device
- Support of 32-bit address range
- Transfers protected by CRC16
- Programmable time outs for detection of blocked answer transfers
- Automatic frame transfer ID generation for detection of dropped frames
- Support of DMA driven multiple register write / read transfers
- Efficient transmission and reception of large data blocks / streams

- Acknowledge for command and stream frames to reduce latency of error detection
- Two stage FIFOs for transmitting and receiving streaming data
- Automatic FIFO flush when entering the run mode, for error handling
- Write protection by an external Memory Protection Unit MPU
- Remote trigger of event / interrupt in the target device by the initiator
- Identification of the target by the JTAG ID number
- Feature set identification of the HSSL module possible by using the JTAG ID number
- Access protection from an external master

Description

The HSSL module generates all necessary protocol formats in order to enable two devices to communicate to each other. Two basic modes are supported.
In register transfer mode the transmitter (“sending device”) provides an address and a data value by writing them into dedicated registers of the HSSL module. The HSSL module puts it into an envelop (called frame) and forwards it via physical layer to the target device. The target takes the information out of the frame and writes the received data into the address which came in the same frame as the data. If the sender wants to read the content of a register or a memory location, it simply sends only the address. The target device reads the value from the specified location and replies it to the sender by transferring it through an answer frame, which has a format of it’s own.

The transfer of register / memory contents is protected by several security levels. A 16 bit CRC value (called CRC16) is attached to the end of each frame. Each transfer process triggers a watchdog, which makes sure that the communication happens within a deterministic schedule. The target device itself offers the capability to define access windows, which allow reading or writing only from / into certain address ranges.

Each frame must be acknowledged by the target by sending an appropriate response frame. If the response frame does not come inside a predefined time window, the initiator module triggers a timeout interrupt.

In data stream mode data blocks of user defined size can moved from the memory of the sending device into the memory of the target device. A DMA master, which can be configured with any source address and source length, autonomously fetches data, breaks it into smaller blocks of programmable size, and puts it into a FIFO. The lower level companion module of the HSSL module empties the FIFO from the “other side” to the physical lines, which connect two devices. In the target device receives a structure, which is inverse to the sender system, the data blocks, puts them into a FIFO, which is in turn emptied by a DMA master into the target memory.

All streaming data transfers are subject to the protection features which are available for the register transfer mode.

After module reset, most registers are only released for read for remote access. They have to be unlocked by the local SW.

In order to ensure consistent initialization of the communication channel there is a possibility to determine the feature set of the module by using the JTAG ID of the device, which uniquely identifies the available HW options of the module. ASIC versions of the HSSL interface may only employ stripped down instances.

35.2 Overview

The following section gives an overview of the lower communication layers of the HSSL protocol.

35.2.1 Lower Communication Layers (HSCT, PLL, Pads)

The lower layer companion module and logic control the lower communication layers positioned between two devices, the data and the physical layer. They consist of High Speed Communication Tunnel (HSCT) module, a corresponding PLL and pads. For more information on the lower communication layers, see the HSCT chapter.

The chip to chip interface employs a digital interface for inter chip communication between a master IC and a slave IC. The interface is capable of running in a master or in a slave mode. During configuration phase the role of the Interface (master or slave) has to be defined. It is not intended to change the system role during an application.

The interface consists of a full-duplex RX and TX high-speed data interface based on double ended differential signals (in total 4 lines) and a master clock interface (SYS_CLK). The master IC owns the crystal and provides the clock to the slave. The interface reset is derived from the System reset and provided by chip internal reset signaling.

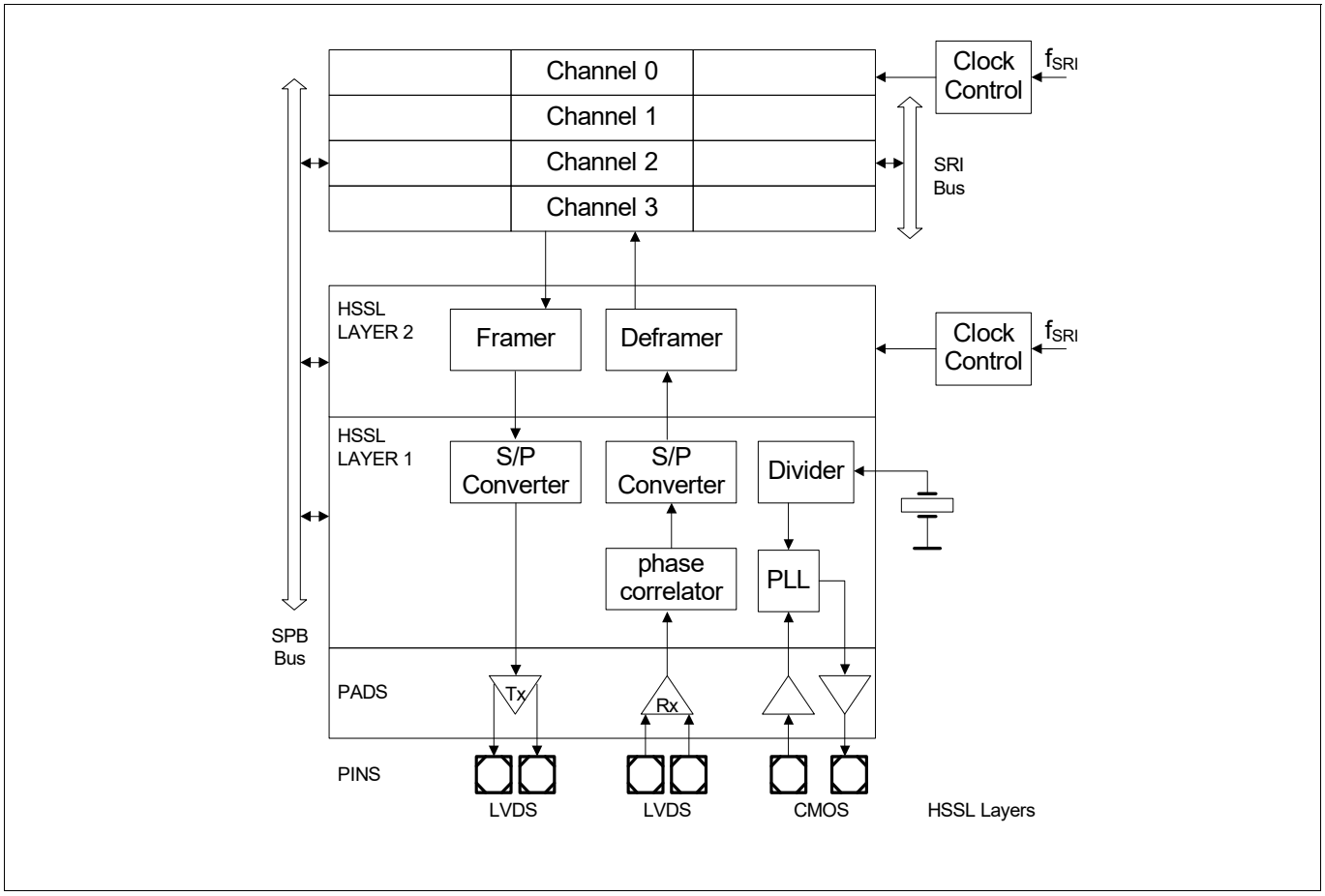


Figure 356 Physical Layer

Features:

- Transmission of symbol information from the master IC to the slave IC
- Receive symbol information sent from the slave IC to the master IC
- System Clock provided by the master IC (crystal owner) to the slave IC
- Exchange of control information in both directions
- Clear To Send indication in both directions
- Unsolicited status in both directions
- Regular data transfer in both directions based on data channels
- Master IC data transfer speed 5 MBaud and 320 MBaud
- Three slave IC data transfer speeds available based on 20 MHz SYS_Clk:
  5 MBaud (low speed)
  20 MBaud (medium speed)
  320 MBaud (high speed)
- Two slave IC data transfer speeds available based on 10 MHz SYS_Clk (lower EMI):
  5 MBaud (low speed)
  320 MBaud (high speed)
- The interface is based on IEEE 1596.3 LVDS input-output pads
- To reduce the voltage swing a configuration option is available

35.3 Functional Description

This section describes the HSSL protocol and it's implementation.

35.3.1 HSSL Protocol Definition

This section describes the high level protocol definition of the HSSL interface.

The HSSL communication normally consists of two stage transactions, sending a command and an receiving a response. It involves two participants:

- an initiator, which starts a transaction by sending a command
- a target, which responds to the command

35.3.1.1 List of Abbreviations, Acronyms, and Term Definitions

ACK - Acknowledge Frame

NACK - Not Acknowledge Frame (Target Error Frame)

CRC - Cyclic Redundancy Check

HSSL - High Speed Serial Link

TO - Time Out

TT - Transaction Tag

TTERR - Transaction Tag Error

SPB - Serial Peripheral Bus

SRI - Shared Resource Interconnection

SoC - System on Chip

Initiator - Device that starts a HSSL transaction by sending a command frame

Target - Device that responds to a command frame sent by an initiator

35.3.1.2 Frame Types

The HSSL module generates a serial frame by:

- taking the payload delivered by a CPU or a DMA
- first wrapping it in a HSSL frame by adding header and CRC fields, and then
- wrapping the HSSL frame in a HSCT frame by adding Sync, Header and End bit fields (see **Figure 357**)

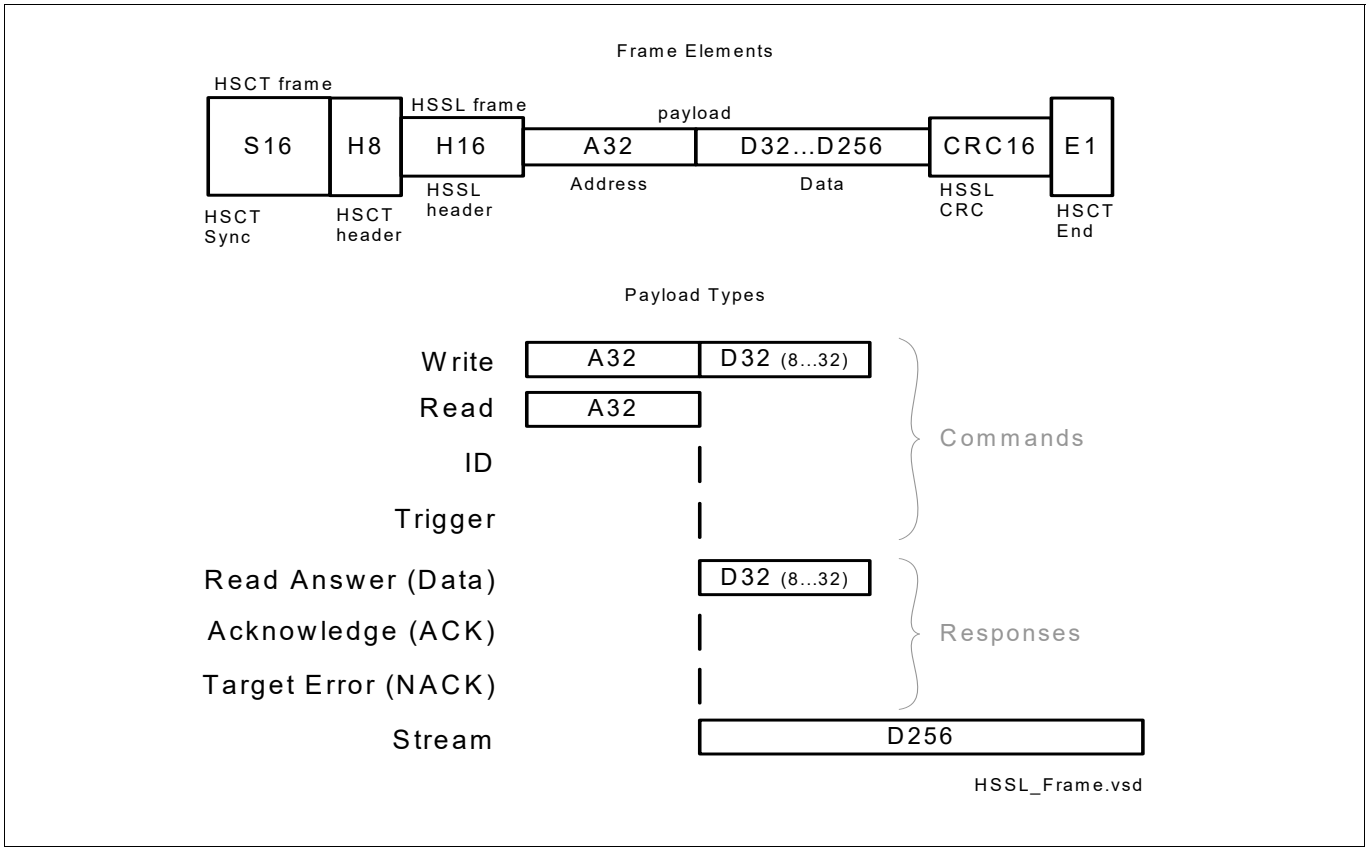


Figure 357 HSSL Frame Types

There are several types of frames:

Command Frames

Command frames are sent by the initiator to request an action by the target. There are four types of command frames:

- **Write Frame**
  - Used by the initiator to request the target controller to write the data to the address, both provided by the initiator.
- **Read Frame**
  - Used by the initiator to request the target controller to read and deliver a content from an address from its address space.
- **Read ID Frame**
  - Used by the initiator to request the target controller to read and deliver a content uniquely defining the target device. The answer is a standard Read Data Answer Frame.
- **Trigger Frame**
  - Used by the initiator to trigger the trigger interrupt in the target module.

In this document, the command frames are sometimes referred to as WRT frames.

Response Frames

Response frames are sent by the target either to deliver the requested data or to report a successful or unsuccessful completion of a request. There are three types of response frames:

- **Read Data Answer Frame**

  - Used by the target to deliver the data, requested by the initiator with a read command frame. It is frequently referred to as a Data frame.
- **Acknowledge Frame**
  - Used by the target to confirm the completion of a write or trigger command. It is frequently referred to as an ACK frame.
- **Target Error Frame**
  - Used by the target to report the unsuccessful attempt to complete a write or read command. It is frequently referred to as a NACK frame.

Stream Frame

Used by the initiator controller for sending data streams. It behaves similar to a write command frame which contains long 256-bit data block, is acknowledged in the same way, but the acknowledge uses sliding window with depth of 2 (instead of 1 as for the simple command frames).

35.3.1.2.1 Frame and Payload Lengths

Each type of frame has a single data length associated with it. The length of the frame depends on the length of the payload. Different frame types have different payload lengths, but the payload length for one type is constant for all variations of the frame type.

For example all data frames have the same length, although they can carry 8, 16 or 32 bit data. This is due to the fact that the data field in the frame is always 32 bit long, and if the data itself is shorter, it is copied several times to fill in the whole field. The same behavior is valid for the read answer frames.

The rest of the frame, consisting of sync and header fields, has always the same length (see **Figure 361**).

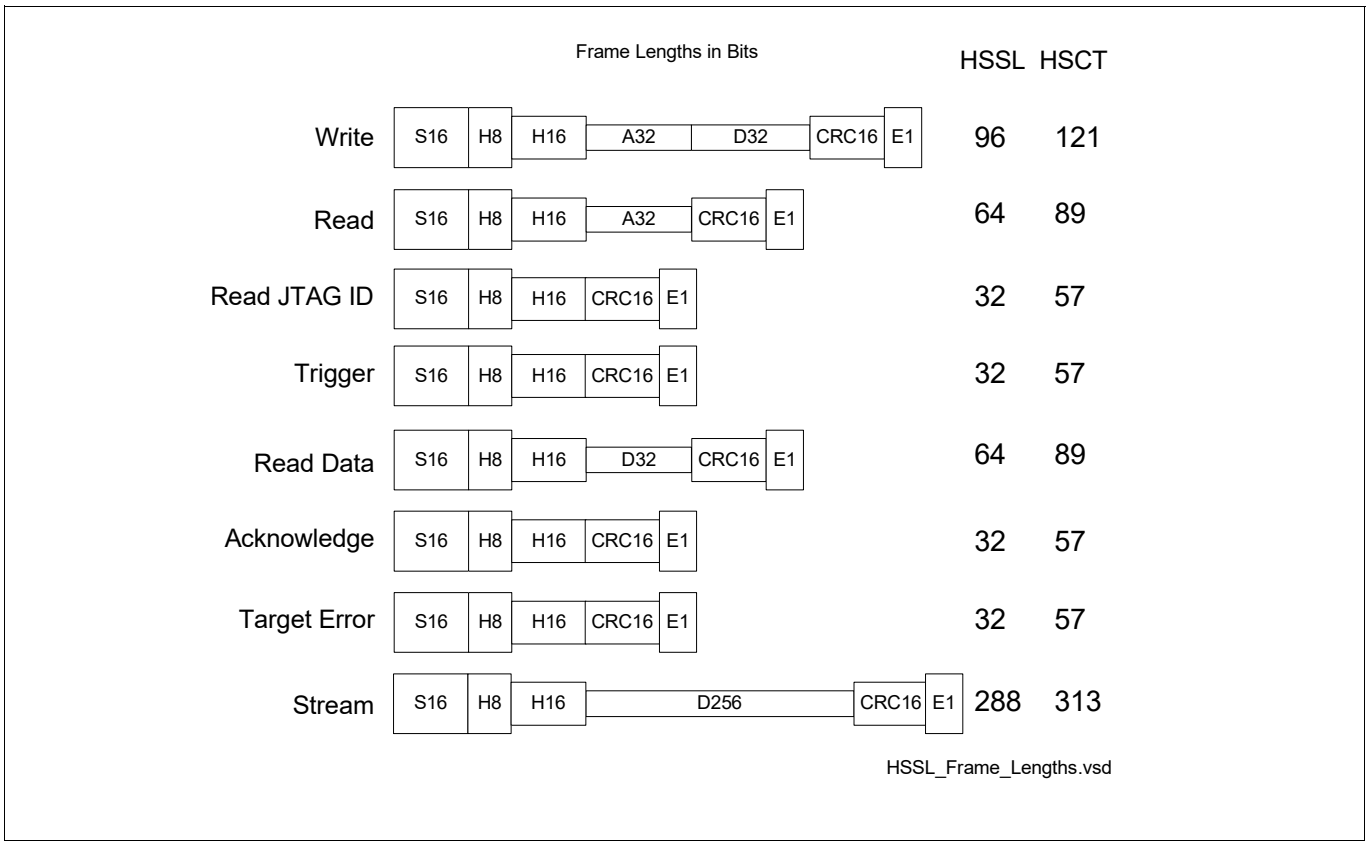


Figure 358 Frame Lengths

35.3.1.2.2 Data Types

There are four types of HSSL data:

- 8-bit
- 16-bit
- 32-bit
- 256-bit stream block

The in case of 8/16/32 bit data types, the transmitted number of bits is always 32, where the shorter data types are copied several times to fill up this length (see **Figure 359**).

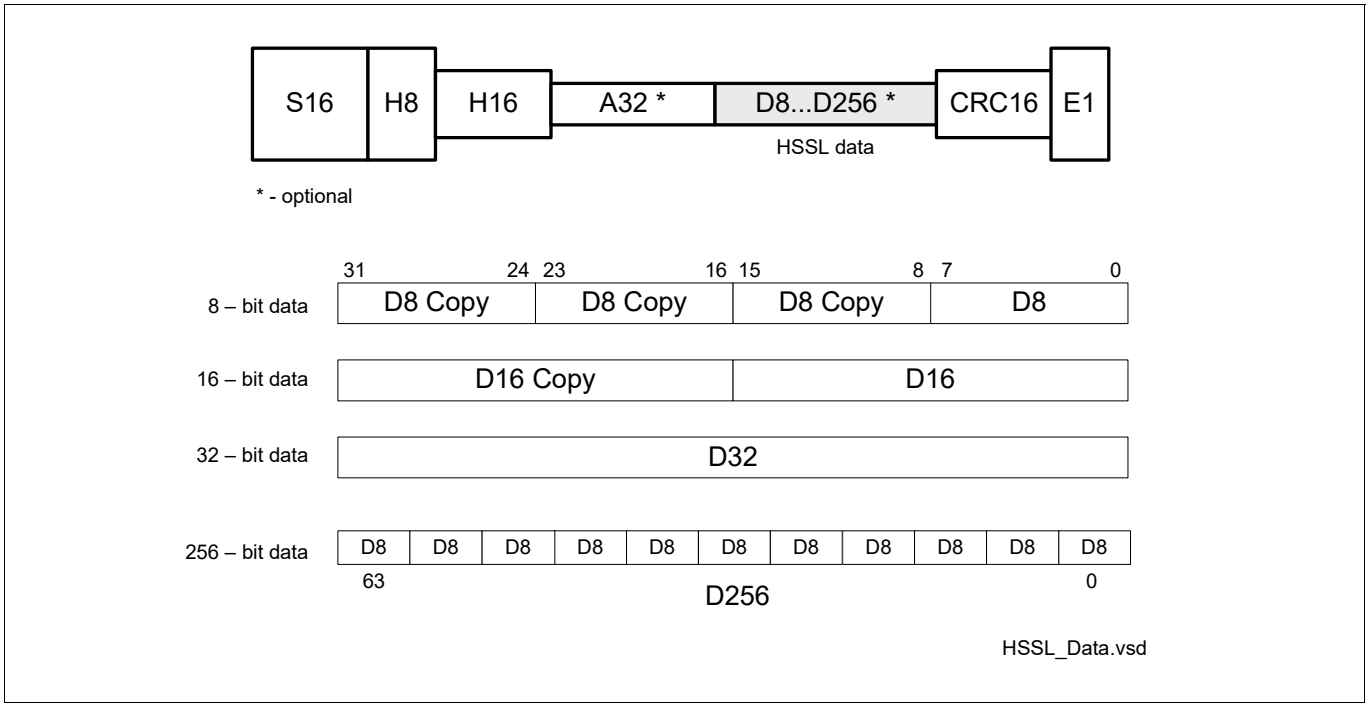


Figure 359 Data Types

35.3.1.2.3 Cyclic Redundancy Check Field - CRC

The HSSL protocol uses the CRC-16-CCITT polynomial: $x^{16} + x^{12} + x^5 + 1$ with the following standard properties:

- seed: 0xFFFF
- calculation direction: MSB first (header msb to lsb, then payload msb to lsb)
- CRC result direction: MSB first

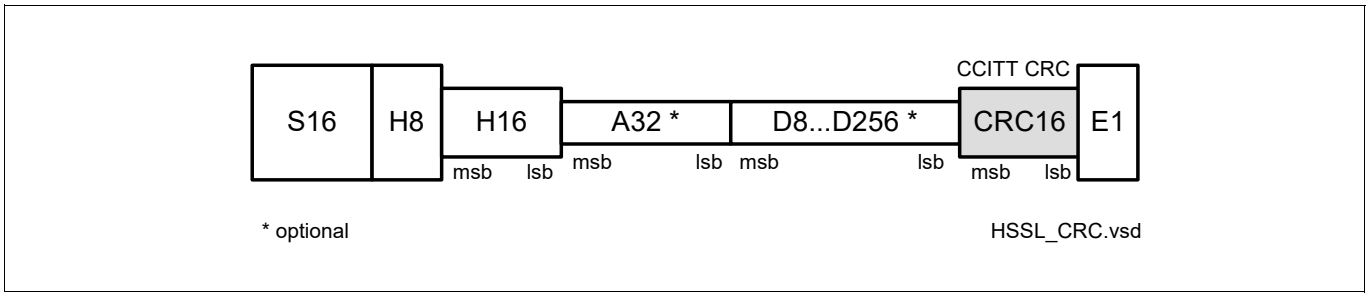


Figure 360 CRC Field

35.3.1.2.4 Header Structure

The HSSL header contains the protocol information that is required additionally to the raw payload data. The HSSL header describes the payload data regarding the length and the type, and carries additional information like channel number code and transaction tag. It also carries stand-alone information in case of frames without payload, like acknowledge and trigger frames.

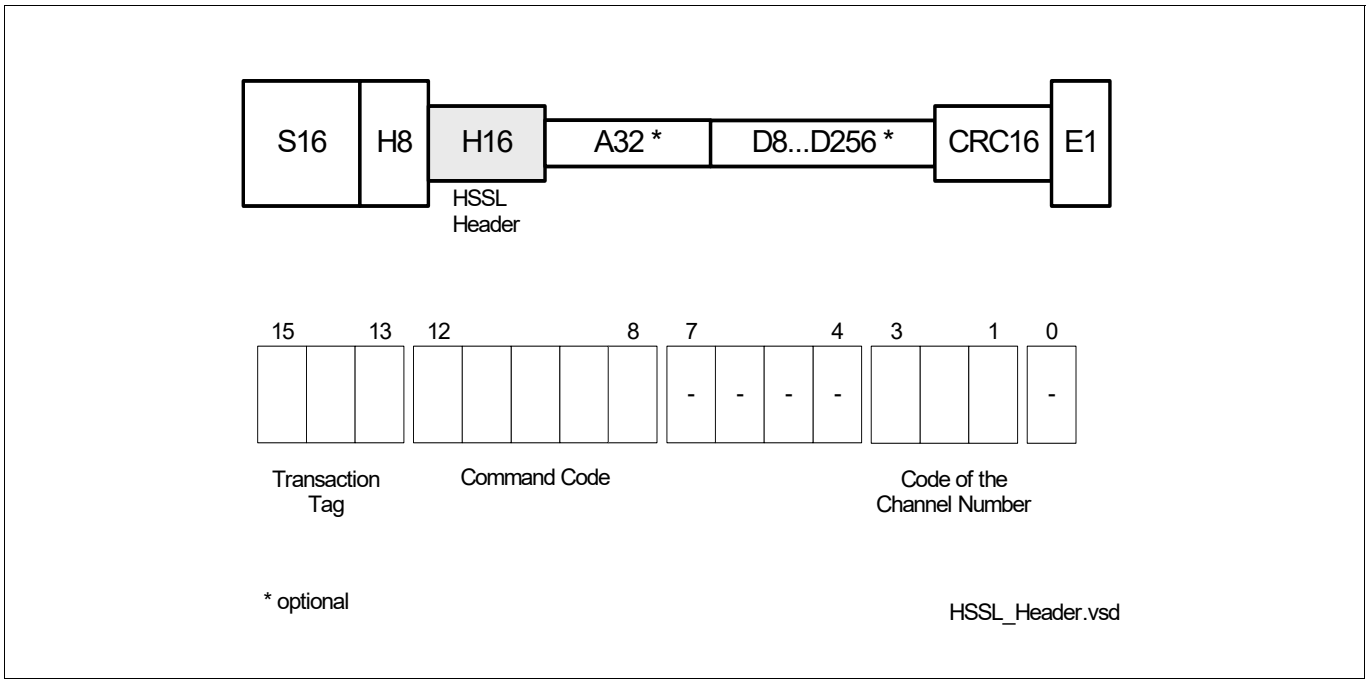


Figure 361 Header Types

Note: The unused bits in the header are padded with 0.

Table 306 Mapping of HSSL to HSCT channel codes

HSSL Channel Number	HSSL Channel Number, Binary Code	HSSL Channel Number, Special Code	HSCT Channel	HSCT Channel Code
0	000	100	A	0100
1	001	101	B	0101
2	010	110	C	0110
3	011	111	D	0111
4 (reserved)	100	000	E	1000
5 (reserved)	101	001	F	1001
6 (reserved)	110	010	G	1010
7 (reserved)	111	011	H	1011

Note: The HSSL module supports four channels, 0 to 3.

Table 307 List of Command Codes

Command Code	Command Description	HSSL Frame Size
00000	Read 8 bit	64
00001	Read 16 bit	64
00010	Read 32 bit	64
00011	RFU (Reserved for Future Use)	-
00100	Write 8 bit with ACK (Acknowledge)	96
00101	Write 16 bit with ACK	96
00110	Write 32 bit with ACK	96
00111	RFU	-
01000	ACK OK	32
01001	ACK Target Error	32
01010	Read Answer OK	64
01011	RFU	-
01100	Trigger with ACK	32
01101	RFU	-
01110	RFU	-
01111	RFU	-
10000	RFU	-
10001	RFU	-
10010	Read JTAG ID 32-bit	32
10011	RFU	-
10100	RFU	-
10101	RFU	-
10110	RFU	-
10111	Stream Data with ACK (32 Byte Data)	288

Table 307 List of Command Codes (cont'd)

Command Code	Command Description	HSSL Frame Size
11000	RFU	-
11001	RFU	-
11010	RFU	-
11011	RFU	-
11100	RFU	-
11101	RFU	-
11110	RFU	-
11111	RFU	-

35.3.1.3 Single and Block Transfers

Single frame transfers are always performed expecting an acknowledge.

35.3.1.4 Streaming Interface

Stream frame transfers are always performed with acknowledge.

There are only write streams possible. Read streams are not possible.

35.3.1.5 Sliding Window Protocol

For flow control the HSSL module implements two variants of the "Sliding Window" protocol regarding the sliding window depth:

- depth of one used for command frames
  - The depth of one means that a single timer tracks the arrival of the corresponding response frame. A new command can be sent only after the response to the previous command has been received, or a timeout occurred
- depth of two used for streaming frames
  - the depth of two means that there are two timers tracking the acknowledge times for the two latest streaming frames. In this case, a second stream frame can be sent although an acknowledge / timeout to the previous stream frame has not been received yet. After the second stream frame, no third stream frame can be sent before the first one has been acknowledged

An example of communication using sliding window with depth of one protocol is shown in **Figure 362**. Here, the initiator sends commands originating from one and the same channel, for example channel 0 (labeled with the capital letter "A"), and the corresponding acknowledge frames from the target are labeled with the small letter "a".

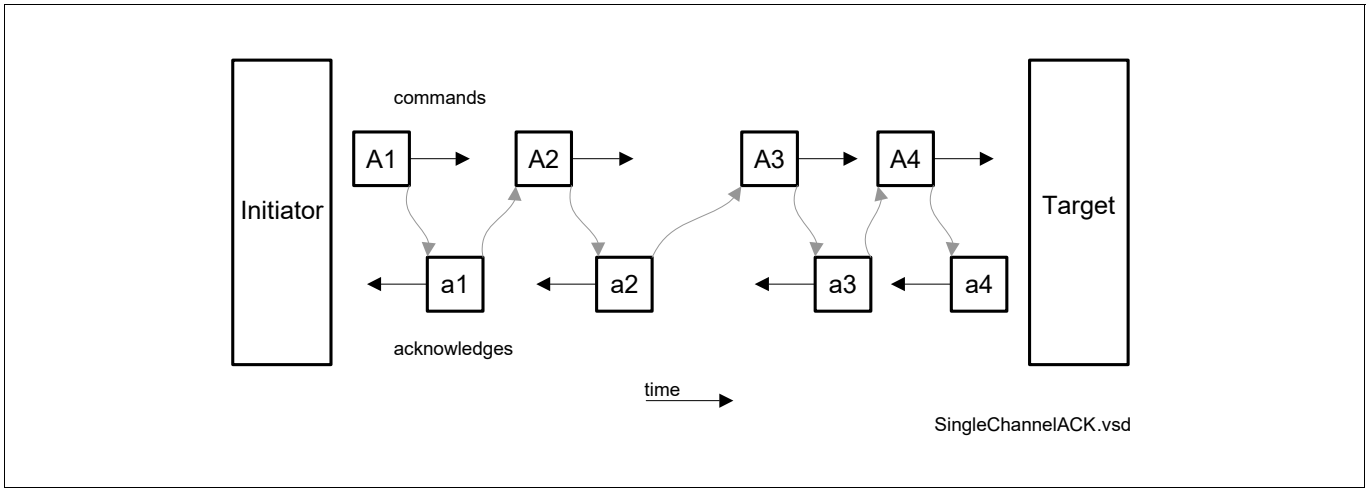


Figure 362 Flow Control, Single Channel, Sliding Window With Depth of One

The sliding window protocol requires capability of sending acknowledge frames in the target and timeout timers in the initiator. Every time the target receives a frame without detecting an error, it sends a response / acknowledge to the initiator. If the target detects a frame with a CRC error, it does not send an acknowledge. The missing acknowledge causes a timeout event in the initiator channel.

If the command frames are longer than the acknowledge frames plus the reaction time, like for example in case of streaming, back to back transfers are easily possible, see **Figure 363**.

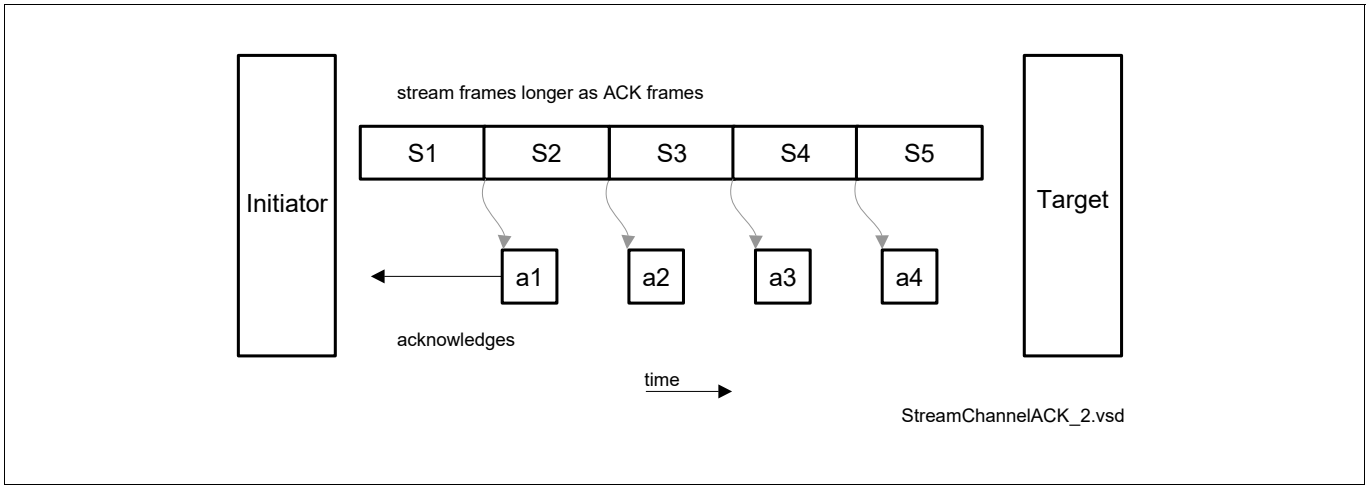


Figure 363 Flow Control, Streaming Frames

35.3.1.6 Error Management

The HSSL protocol defines four types of errors:

- Time Out
- Transaction Tag Error
- CRC Error
- Target Error

Time Out Error

A time out error is detected at the initiator side, if the expected ACK frame was not received within the expected time window. This can occur if a frame had been sent by the initiator, and the target detected an CRC error and

did not answer with an acknowledge, or the connection between the initiator and the target is physically damaged in one or the other direction.

Transaction Tag Error

A transaction tag error occurs at the initiator side, if instead the expected ACK frame with the expected TAG number, an acknowledge with an unexpected transaction tag was received. This would indicate a missing frame or missing acknowledge. Transaction tag error generate frames that pass the CRC checking stage.

CRC Error

A CRC Error can occur:

- at the target side, in which case:
  - the CRC error flag is set
  - the received command frame with a CRC error is discarded
  - no acknowledge frame is sent and
  - a channel unspecific EXI error interrupt is generated, if enabled.
- at the initiator side, in which case:
  - the CRC error flag is set
  - the received response frame with a CRC error is discarded
  - a channel unspecific EXI error interrupt is generated, if enabled

Both scenarios lead to a time out at the initiator side.

In both cases the CRC error flag is set at the side receiving the erroneous frame (either initiator or target) and an interrupt is generated, if enabled.

For the purpose of CRC error injection, a bit field with an XOR mask used for toggling the bits of the calculated CRC is provided, see **CRC**.XORMASK. In order to switch on this feature, the E (Endinit) protected bit must be set by the application software. The error injection only influences the generation of the transmitted CRC.

Target Error

A Target Error can occur at the target side, when accessing the target memory a bus error or memory protection error occurs. In such a case, the target responds with an NACK frame indicating the error.

35.3.1.7 Shift Direction

In general, the HSSL interface makes a copy of a memory location or block from the address space of one microcontroller into the address space of the other microcontroller.

The HSSL module delivers 32-bit parallel data.

The transfer over HSCT serial link is done by using MSB first shifting at 32-bit level.

The overview of the path from the memory to the serial bus is shown in the **Figure 364**.

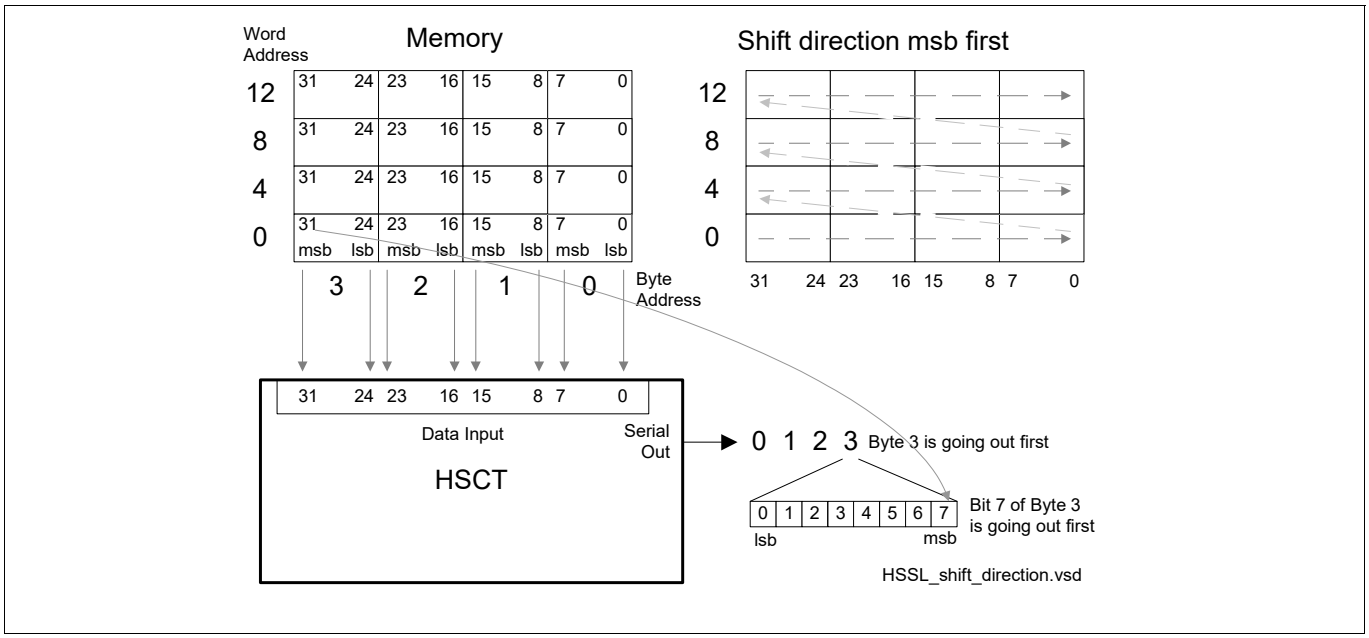


Figure 364 Shift Direction in Case of HSCT Layer

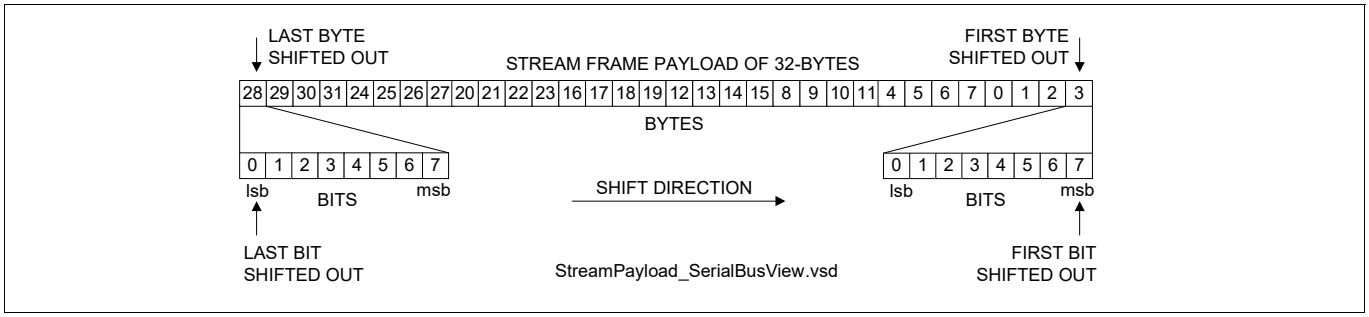


Figure 365 Stream Frame in Case of HSCT Layer

35.3.2 HSSL Implementation

This section describes the implementation of the HSSL protocol.

35.3.3 Overview

The HSSL module is connected to the SRI bus using a master interface and to the SPB bus using a BPI slave interface.

Additionally, it is connected to the SPB bus using a master interface.

The SRI master interface is capable of performing single and burst reads and writes on the SRI bus. Additionally, the HSSL kernel provides logic for performing streaming communication. The write and read accesses are always performed in User Mode.

The SPB BPI slave interface is used by an SPB master (DMA, CPU, HSSL SPB Master) for writing the HSSL registers, thus configuring the module and performing single read and write operations.

The SPB master interface is capable of performing single read operations, write operations, and streaming communication using BTR4 burst accesses on the SPB bus. The SPB master accesses are always performed in User Mode.

The HSSL module to use the SRI or the SPB master depending on the address of the access. Accesses to the SPB address space are automatically routed to the SPB master, accesses to the SRI address space are automatically routed to the SRI master. The address window F000 0000_H - F7FF $FFFF_H$ is used for the SPB bus, otherwise SRI.

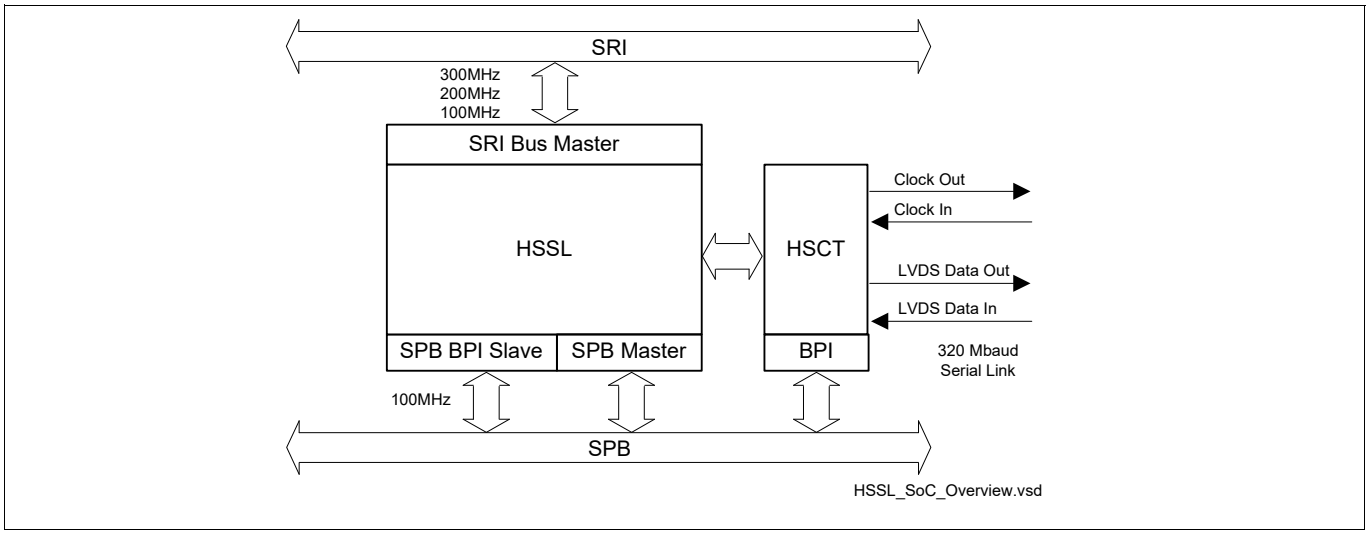


Figure 366 HSSL Subsystem Overview

35.3.3.1 HSSL Module Operation

The HSSL module consists of separate transmitter and receiver, each containing four channels.

The transmitter contains an arbiter block and a wrapper block, which are common for all channels. The arbiter provides fixed arbitration between the channels which are ready to send when the serial link is available, with channel 0 having the highest priority, and the channel 3 having the lowest priority. The wrapper block generates the CRC.

The receiver unwraps the HSSL frame, discards the frames with a CRC error, and distributes the error free commands and acknowledges to the channels.

A common prescaler generates the time basis for all channels, in particular for their timeout timers.

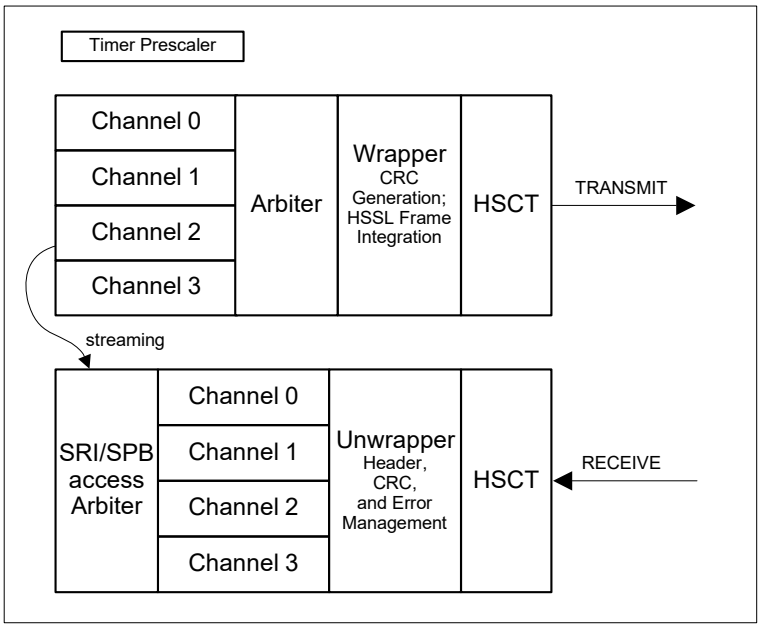


Figure 367 HSSL Layer Architecture

35.3.3.1.1 Frame Transmission Priorisation

When more than one frames are pending, an arbitration is performed between them.

Each channel activates two flags: initiator command flag and response flag: channel 0 I0 and R0, channel 1 I1 and R1, … .

Between the response and command frame request types, the response request type has higher priority. Among the requests of a same type, lower channel number has higher priority, see **Figure 368**. A pending frame is indicated with a corresponding request flag in the register **QFLAGS**.

The SRI/SPB bus master sets the response requests (“R” flags).

The software write accesses via the SPB bus set the initiator requests (“I” flags).

The transmit arbiter clears the “I” and “R” flags of the arbitration winner.

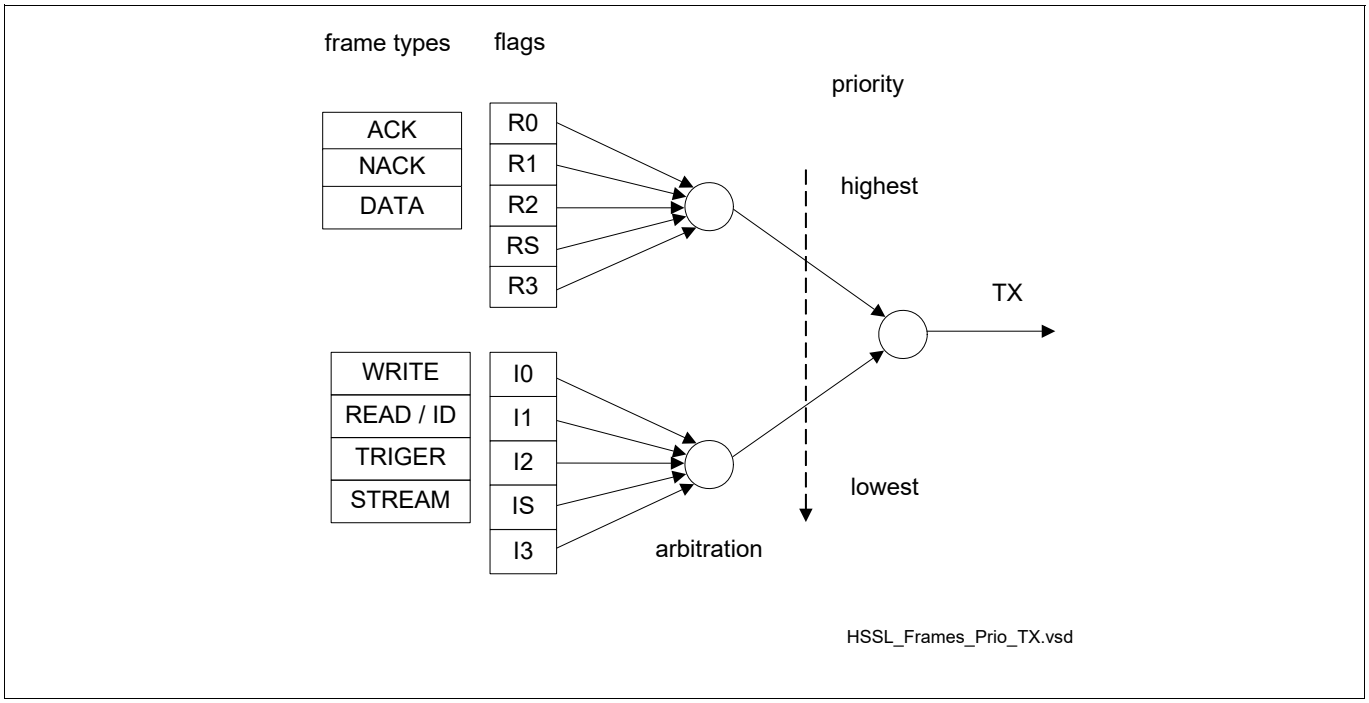


Figure 368 Frame Priorities in the HSSL module

35.3.3.1.2 Received Frame Management and Command Execution

After a frame passes the CRC check, it is checked for its type. A received command frame sets a request flag for the SRI/SPB master, a received response frame resets the expect flag and stops the appropriate channel timeout, see **Figure 369**.

The target flags “T” are set by the frame distributor, and cleared by the arbiter of the SRI/SPB master.

The expect flags “E” are set by the Tx arbiter, and reset by the Rx distributor.

The ISB and ISF flags operate together to control the streaming process. The ISB flag (Initiator Stream Block flag) is set by the software to enable the streaming. The ISF flag (Initiator stream FIFO flag) is set/cleared by the TXFIFO according to its filling level.

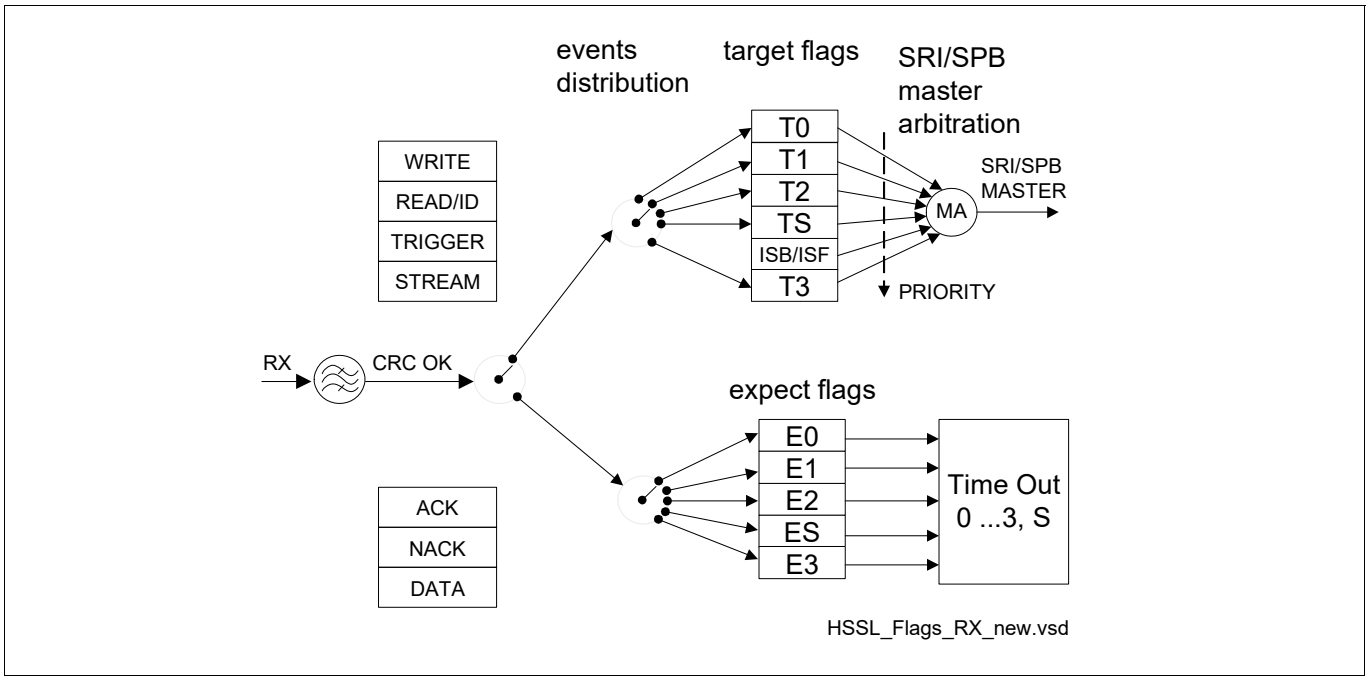


Figure 369 Received Frames Management

The CRC and Transaction Tag error flags are located in the **MFLAGS** register.

An overview of a full communication cycle over all sections of a channel (initiator, target or receive, transmit) is shown in **Figure 370**.

See also the **QFLAGS** register description.

If the channel does not have a command pending and does not expect a response, it is ready.

Note: In case of at trigger command frame, the target command distributor sets the corresponding Rx flag, and not the corresponding Tx flag, due to the fact that the service request is triggered directly, and not by the SRI/SPB master,

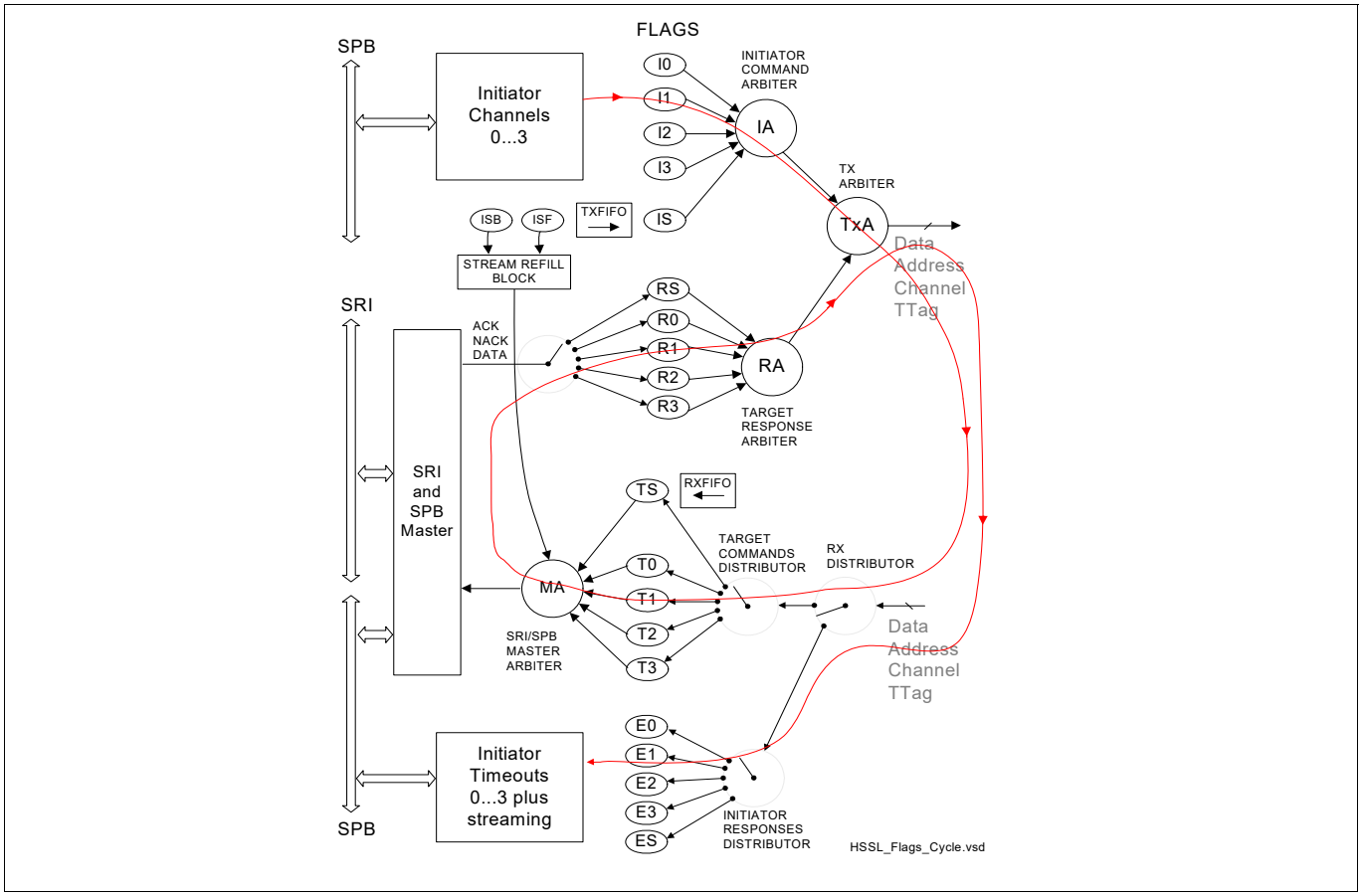


Figure 370 Request Flags Overview and a Communication Cycle Example

Table 308 Set and Clear of Request Flags

	I	R	T	E	ISB	ISF
SPB Software Write	set	-	-	-	set	-
Tx Arbiter	clear	clear	-	set	-	-
Rx Distributor	-	-[1)]	set	clear	-	-
SRI / SPB Master	-	set	clear	-	-	-
Stream Refill Block	-	-	-	-	clear	clear
TXFIFO	-	-	-	-	-	set

1) In case of a trigger command, an R flag is set by the Rx Distributor instead of a T Flag, because a trigger command is not executed by the SRI/SPB master.

35.3.3.2 HSSL Channel Architecture

The HSSL module contains four channels. The functionality of each channel can be subdivided in two ways:

- transmitter and receiver functionality
  - The transmit functionality generates an appropriate header for each frame
  - The receiver generates an appropriate acknowledge header with the tag and channel number code. The read and writes are performed by the SRI/SPB master. Therefore, the module can send back to the transmitter a target error frame if an SRI/SPB bus error occurs and the transfer on the bus can not be executed.
  - The timeout monitoring on one hand and the acknowledge management on the other hand involve both the transmitter and receiver channels
- initiator and target functionality
  - The initiator functionality transmits write, read or trigger command frames in random sequence, after the previous command has been responded to.
  - Generation/comparison of a Transaction Tag per channel is an initiator functionality
  - The timeout management is a pure initiator functionality.
  - The target functionality is to receive commands, executes them, and generates and transmits responses.

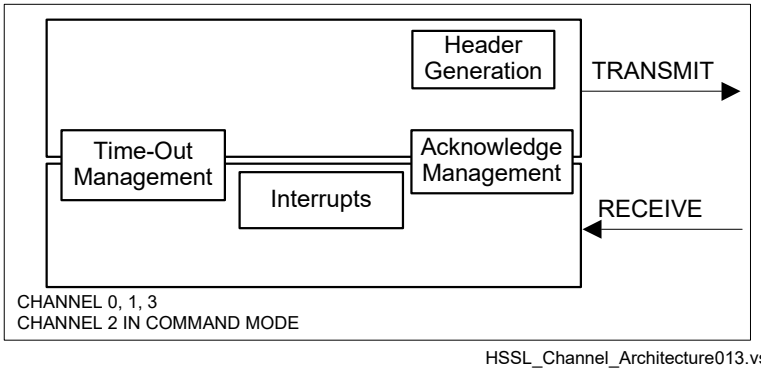


Figure 371 Architecture of Channels 0,1, 3 and Channel 2 in Command Mode

Channel 2 can be switched between command and streaming mode of operation. In single mode a single data register is used and in streaming mode a FIFO is used.

Channel 2 contains one initiator and one target address counter which are used to fill a memory range with data. Each counter contains two start address registers and one stream frame count register.
At reaching the predefined frame count, a wrap around is automatically executed.

The granularity of the start address is 256-bit.

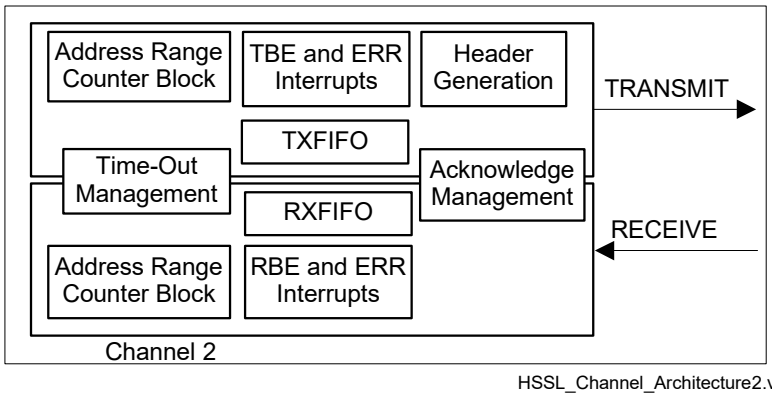


Figure 372 HSSL Architecture of Channel 2

A classification of the various channel functions is shown in **Figure 373**.

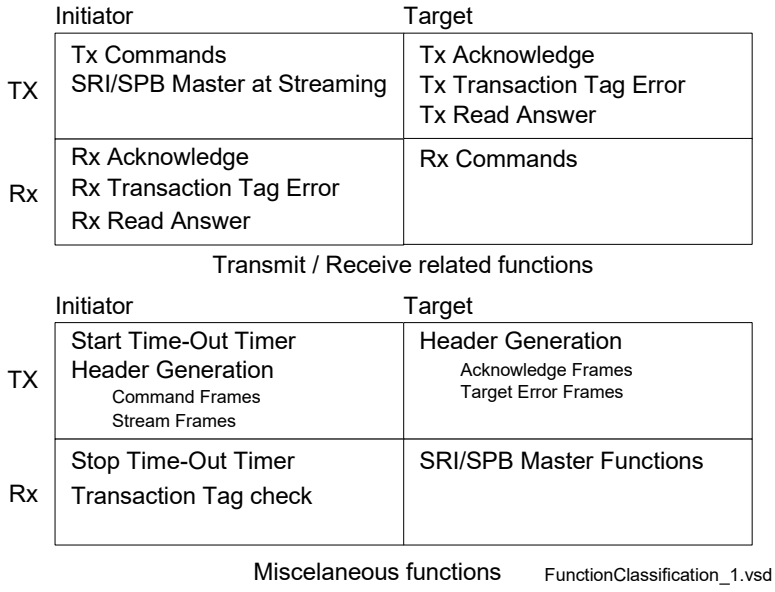


Figure 373 Overview of Channel Functions

35.3.3.3 Acknowledge Responses

A read or write command is executed by the SRI / SPB master. The SRI / SPB master reads or writes to the SRI / SPB bus and signals either a successful transaction or an error. The are the following cases:

- When the SRI master writes or reads an SRI memory, both cases are acknowledged so that the HSSL module receives a feedback in any case.
- When the SPB master writes or reads an SPB module or memory, both cases are acknowledged so that the HSSL module receives a feedback in any case.

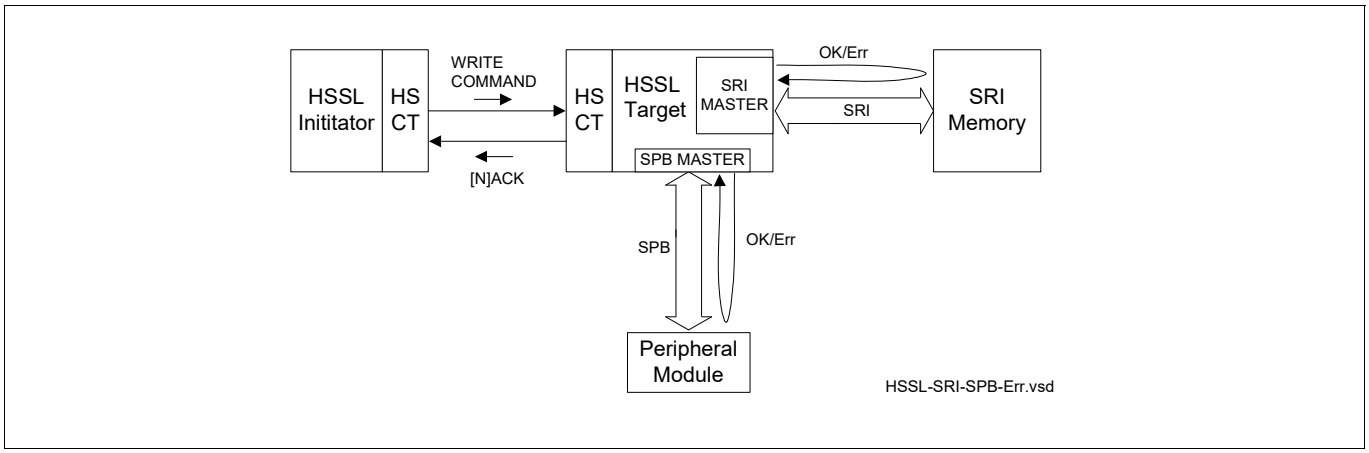


Figure 374 Bus Error Feedback Paths

35.3.3.4 Cross Dependencies Between the Frame Types

For a channel in a target role, there are cross-dependencies between command frame arrival and triggering of an ACK or NACK frame for a channel. Each command frame received with correct CRC of types Write, Stream, or Trigger Frame will result in either ACK or NACK (target error) response frame. Each Read Frame received with correct CRC will result either with Data Frame (Read Response Frame) or with NACK (Target Error Frame), see **Figure 375**.

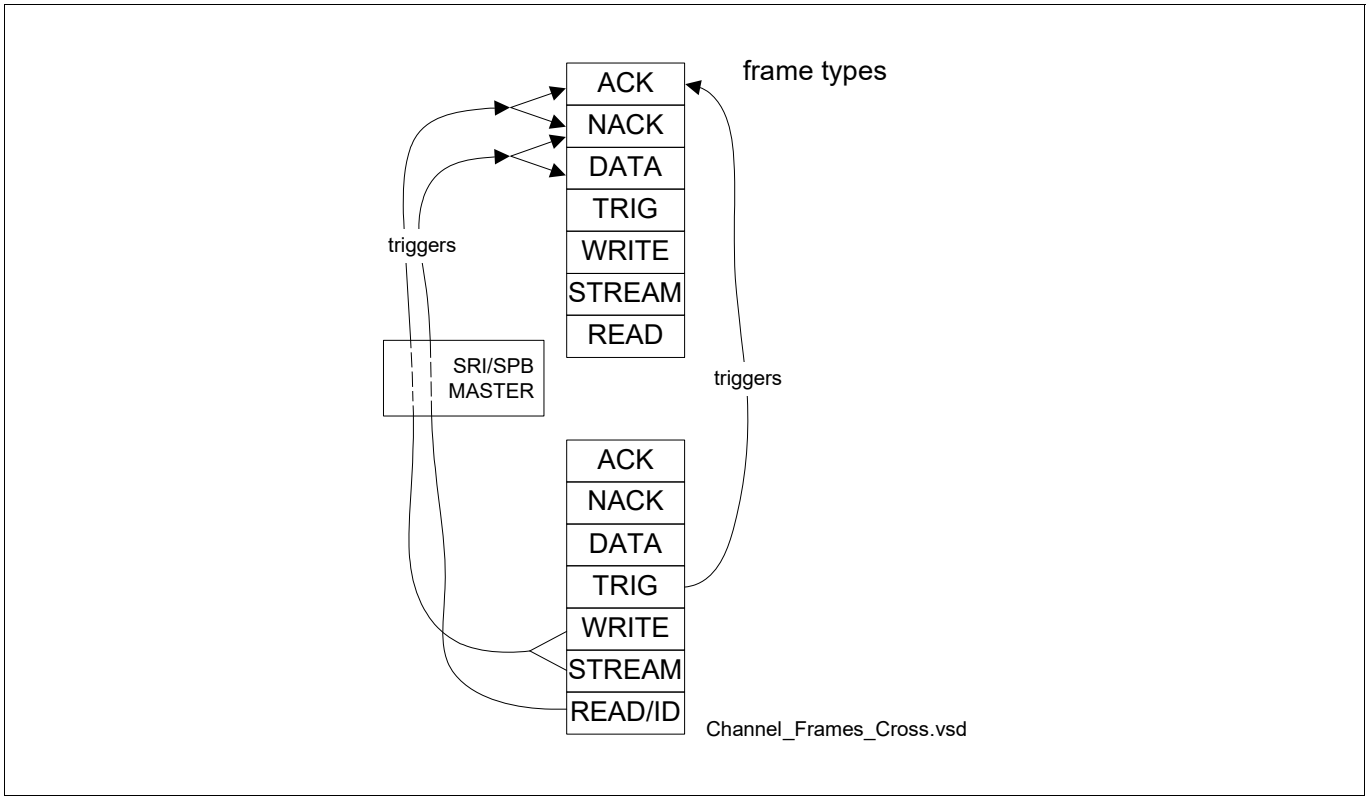


Figure 375 Channel 2 Cross Dependencies Between Stream and Command Frames and their Responses in a Target Role

Additionally, generating a response frame involves copying of the transaction tag of the command frame into the response frame. The response frame is triggered by an SRI/SPB master signalling either a successful SRI/SPB bus transaction or an SRI/SPB bus error. Only a Trigger Command Frame triggers immediate acknowledge, because it results in an interrupt flag being set, and not in a bus transaction.

On the initiator side, there is a forward looking cross dependency between a command frame and the expected response, see **Figure 376**. Setting a transmission request flag for a certain command is accompanied in parallel with:

- capturing the current transaction tag in the bit field **ICONx (x=0-3)**.CETT.
- generating the next transaction tag by incrementing a three bit counter per channel, the **ICONx (x=0-3)**.ITTAG.
- starting the timeout counter at the moment when the command wins the arbitration
- setting an expect tag indicating that the timeout is running and a response frame is expected. See **MFLAGS**.Ex bit fields.

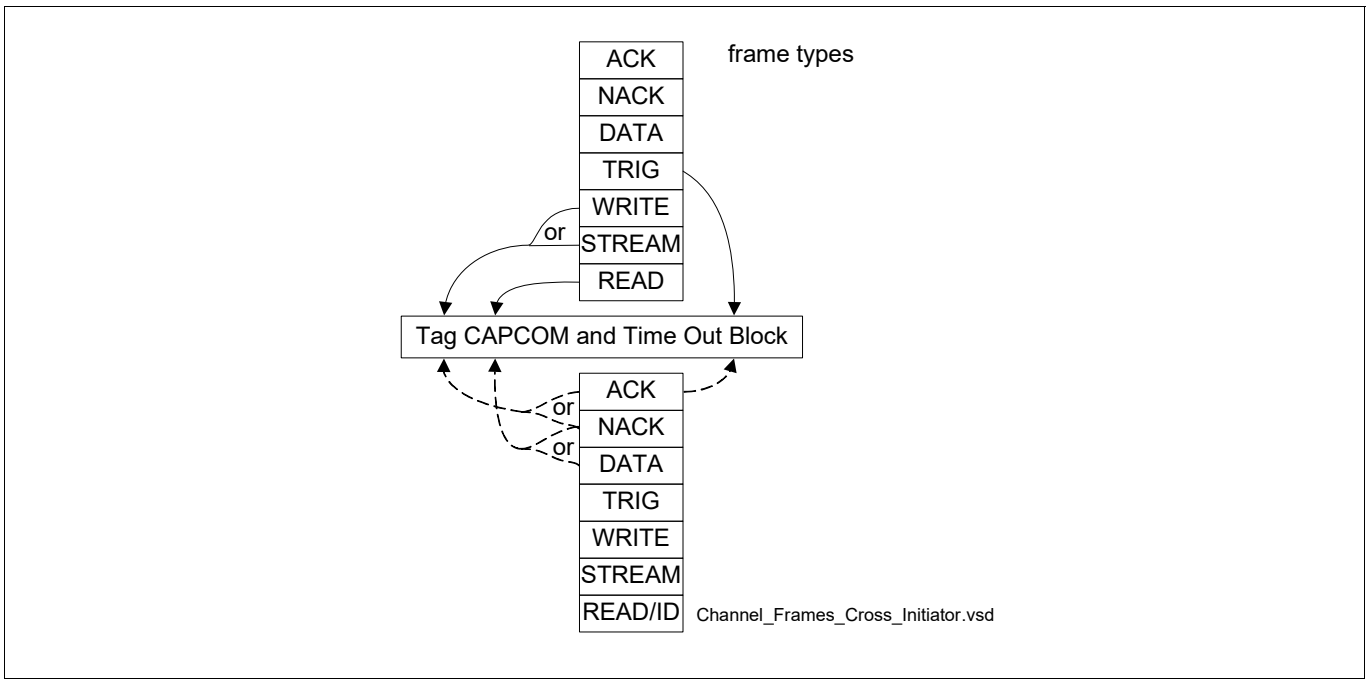


Figure 376 Channel 2 Cross Dependencies Between Stream and Command Frames and their Responses in an Initiator Role

35.3.3.5 Command Timeout

A timeout timer is started when a frame wins the arbitration and is delivered by the channel for transmission. The timeout timer is stopped if an appropriate response frame has arrived in time (without a CRC error). If an appropriate response has not arrived, and the timeout timer reaches zero, the timer is stopped, a timeout error is triggered and the expect flag is cleared.

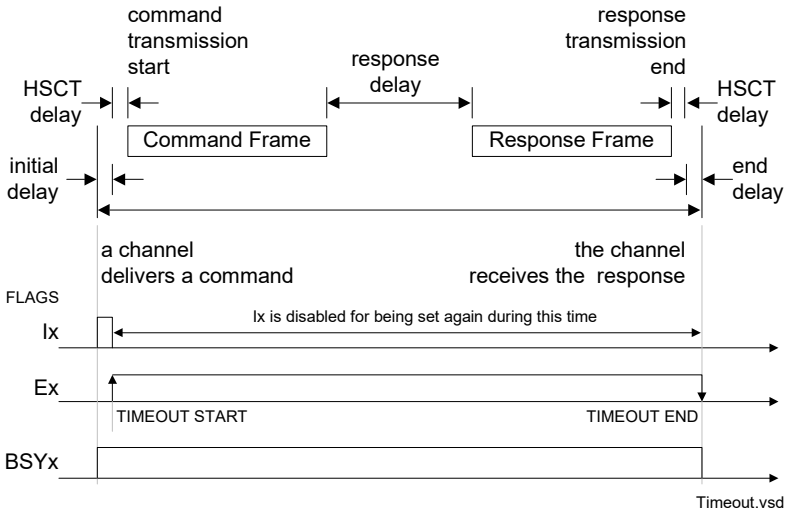


Figure 377 Command Timeout Measurement

35.3.3.5.1 Command Timeout Operation

The HSSL module contains one prescaler common to all channels, and one channel specific timer block per channel. The timer block generates the transaction tag for the channel by incrementing a three bit counter, captures the current transaction tag and compares it with the arrived response. In case of an error sequence CRC and timeout error, the timeout block generates a timeout signal. The timeout timer is reloaded and starts down-counting when the command transmission process starts.

The current timeout timer current value is indicated in the bit field **ICONx (x=0-3)**.TOCV, and the timeout reload value is configured in the bit field **ICONx (x=0-3)**.TOREL.

The currently expected transaction tag is indicated in the bit field **ICONx (x=0-3)**.CETT, the captured value of the latest erroneous transaction tag is indicated in the bit field **ICONx (x=0-3)**.LETT.

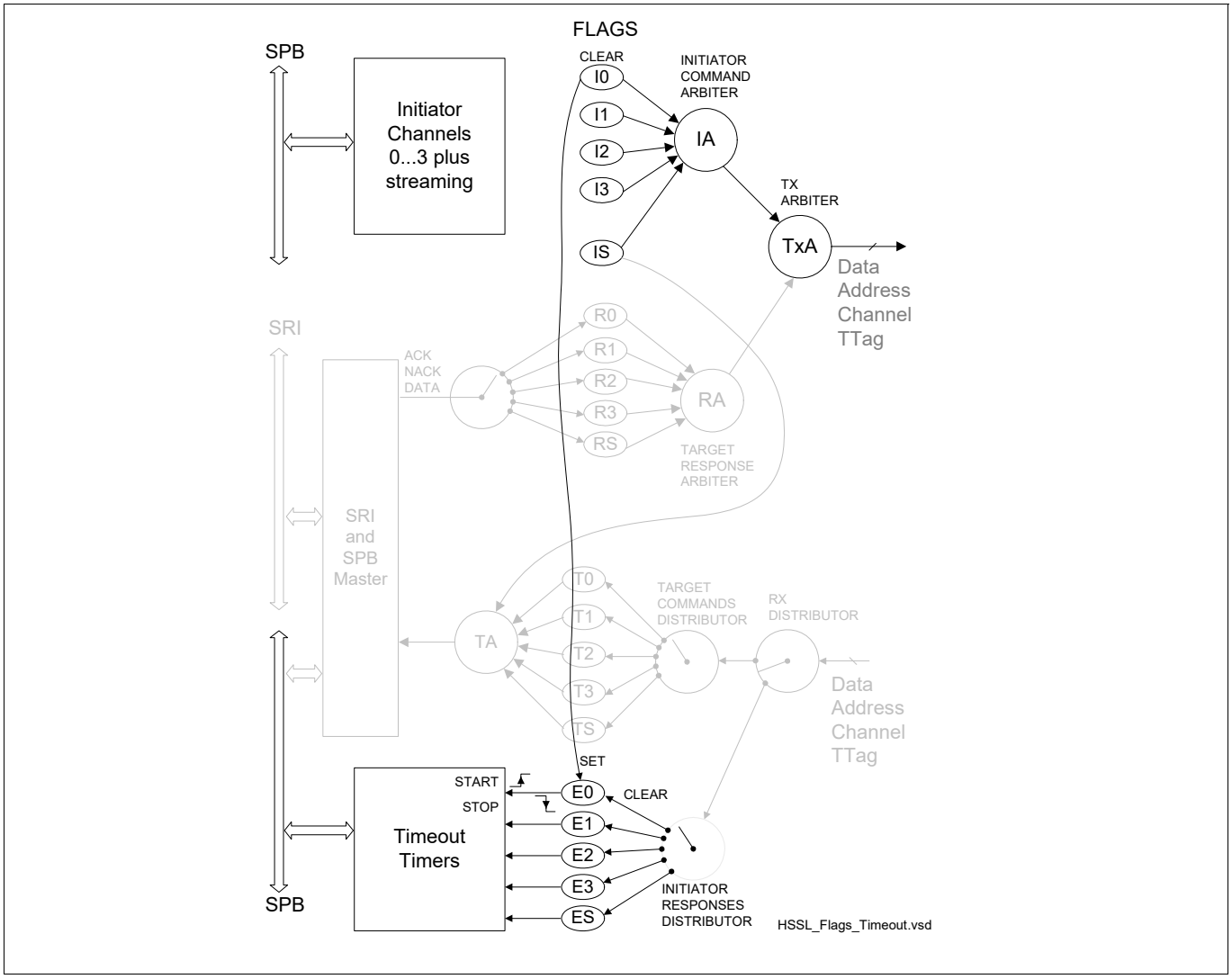


Figure 378 Request Flags Overview and a Communication Cycle Example

From the point of view of the flags, the timeout starts with the rising edge of the **QFLAGS**.Ex flag and ends with its falling edge, see **Figure 376**. The Ex flag is set by the falling edge of the Ix flag, which is reset when its command request wins the transmit arbitration, see **Figure 378**.

35.3.3.6 Stream Timeout

A free timeout timer block is started when a frame wins an arbitration round and is delivered by the channel for transmission. The timeout timer block is reset to zero and stopped if an appropriate response frame has arrived in time (without a CRC error). If an appropriate response has not arrived, a timeout error is triggered and the timeout timer block is reset.

The stream initiator keeps track of the expected acknowledges using a virtual expect fifo having a virtual filling level EXFL. Each new stream frame transmission increment the EXFL, receiving a correct acknowledge decreases the EXFL. As long as the expect fifo is full (filling level 2) no new transmit request can be issued. In order a new stream frame request to be issued, which means the flag IS to be set by the module, three conditions must be met:

- EXFL<2 - the expect level must be below two,
- TXFL>0 - the TXFIFO must be not-empty and
- ISB = 1 - the bit ISB bit must have been set by the software.

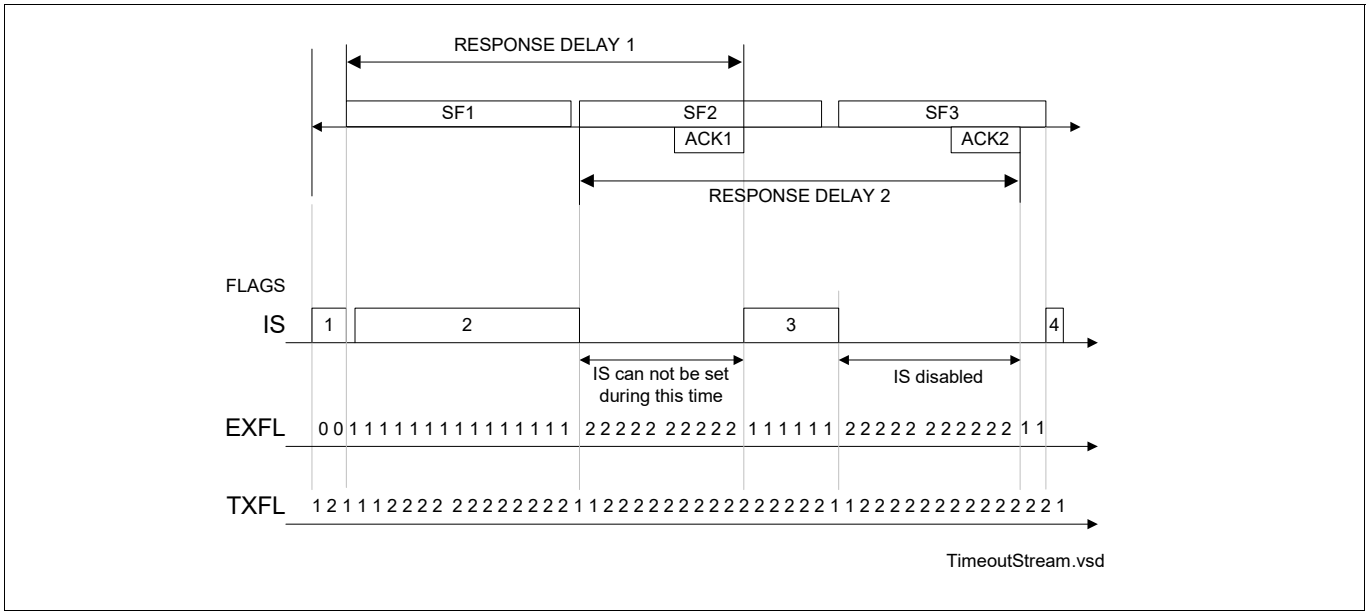


Figure 379 Stream Timeout Measurement

35.3.3.6.1 Stream Timeout Operation

The stream channel monitors the arrivals of the stream frame responses by using two timeout timer blocks. A single timer block is shown in **Figure 380**.

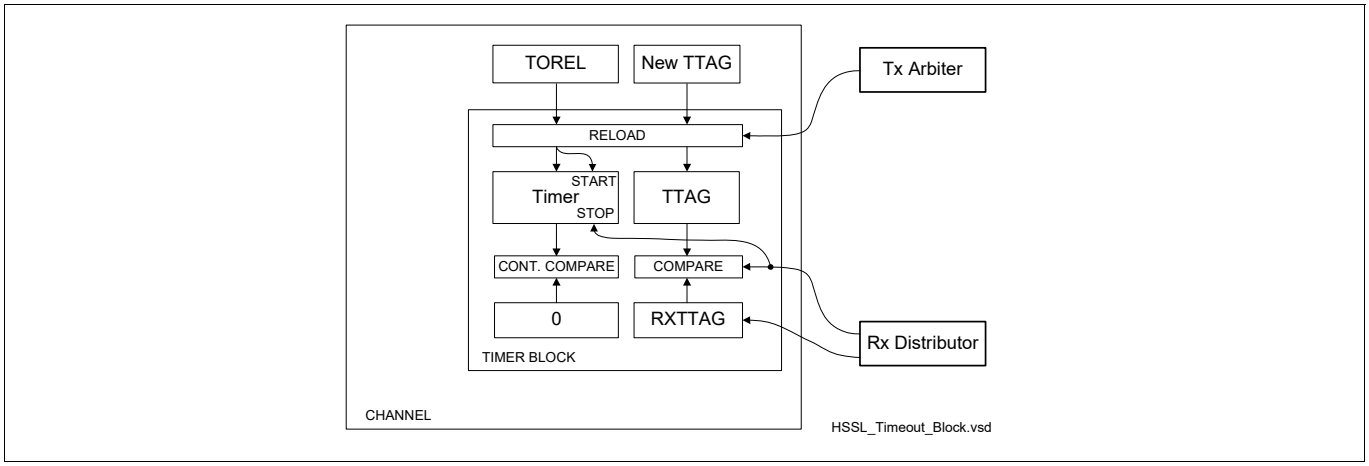


Figure 380 Single Timeout Block

The new initiator TTAG value which will be used for the next frame is visible in the bit field **ICONx (x=0-3)**.ITTAG. The **ICONx (x=0-3)** register also contains the TOREL and the TOCV bit fields.

The streaming functionality uses two timers sharing the same reload value, as defined by the **ICONx (x=0-3)**.TOREL, i=2. The timeout timer blocks simulate the behavior of a FIFO (named Expect FIFO), by having additionally a “write” and “read” pointer:

- the “write” or transmit pointer points to the timer block where the next command frame will trigger the writing of the timer reload value and the transaction tag,
- the “read” or receive pointer points to the block which performs the transaction tag comparison with the currently arrived stream acknowledge TTAG.
- The EXFL or virtual filling level indicating how many timers are active 0, 1 or 2

The EXFL is visible in the register **SFSFLAGS**.

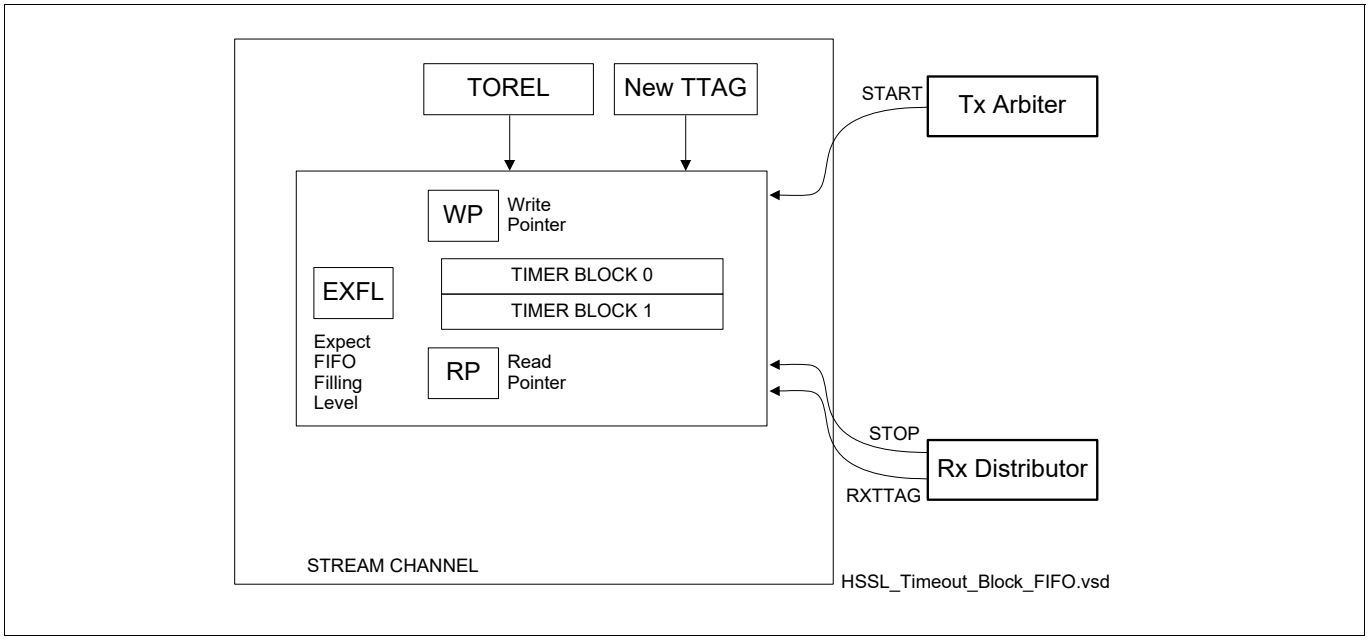


Figure 381 Stream Timeout Block

35.3.3.7 Data FIFOs of the Streaming Channel 2

Each stream frame delivers 256 bit data. This data is transferred by the SRI/SPB bus master in a memory by using BTR4 bursts. The start address and the end address is aligned at 256-bit block border, see **ISSAx (x=0-1)**, **ISFC**, **TSSAx (x=0-1)**, **TSFC**.

The channel expects one full streaming frame of 256 bit to be delivered before triggering either the SRI/SPB master or the HSCT module for fetching the data. The streaming is full duplex capable, which means one FIFO for each direction is available, and also Priorisation for the requests to the SRI/SPB master.

Emptying the RXFIFO has higher priority than filling the TXFIFO.

A service request to the SRI/SPB master is generated when the complete payload of a streaming frame is available in a FIFO.

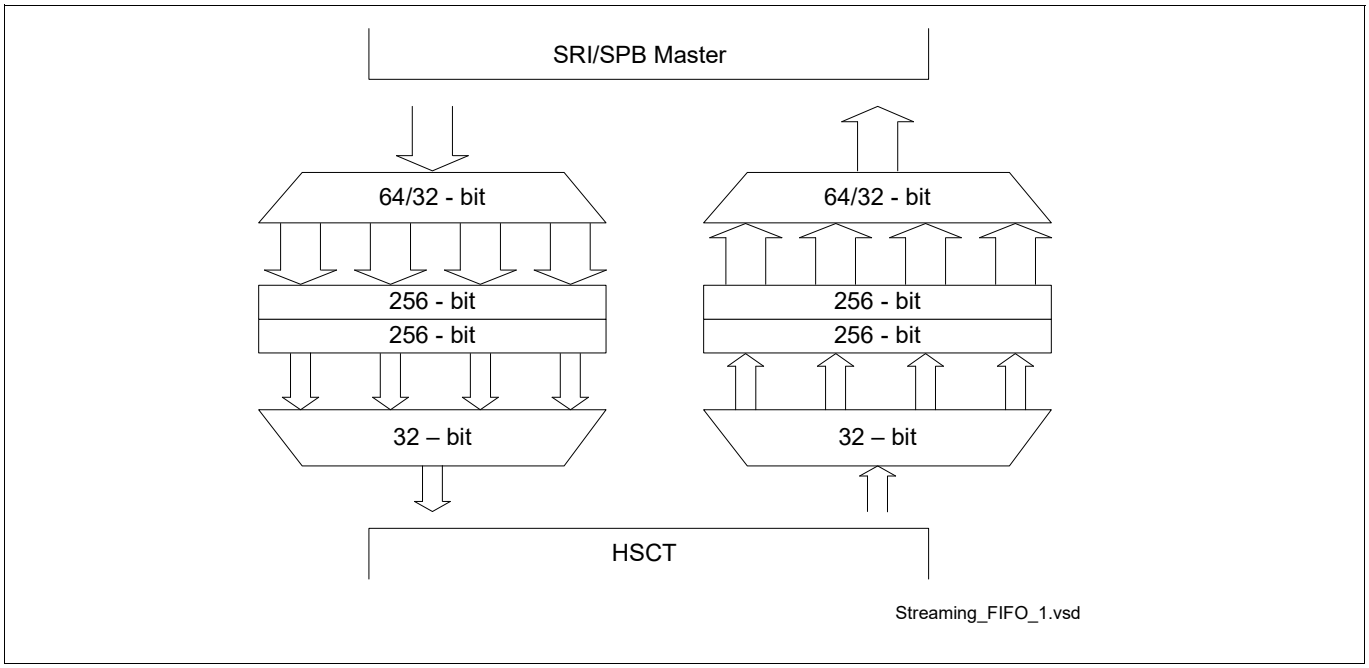


Figure 382 Streaming FIFO

35.3.4 Modes of Operation

The HSSL module has the following modes:

- Disabled mode (software controlled via DISR bit)
- Initialize mode (software via INI bit, sleep & suspend signals)
- Run mode (INI bit, sleep and suspend signals)
- OCDS soft suspend mode (OCDS suspend signal)
- Sleep mode (sleep signal)

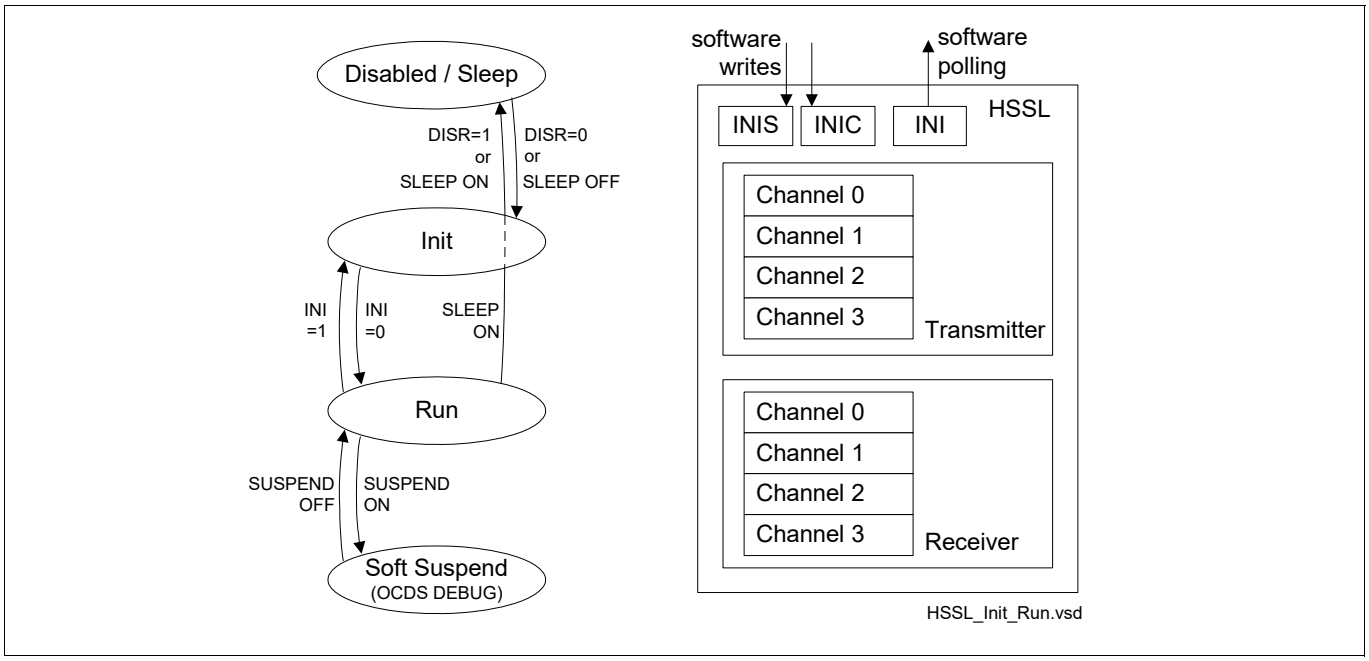


Figure 383 Modes of operation

After INIT state has been requested by using the **MFLAGSSET**.INIS bit, the HSSL module immediately stops starting new SRI/SPB transactions and new command, response and stream frame transmissions by resetting all request bits (I, T and R) in the **QFLAGS** register and disabling these bits for being set again. The already started transactions/transmissions will be finished. Afterwards, it waits for all pending response frames to arrive or the corresponding timeout events to be raised. This condition is achieved when all expect bits in the **QFLAGS** register has been cleared by the hardware, and there are no ongoing transmission and SRI/SPB master activities. Subsequently the module enters the Init state and the **MFLAGS**.INI flag is set. The CTS signal is deactivated in order to inform the peer module about the Init state.

During the transition from run to sleep state, triggered by the sleep signal, the **MFLAGS**.INI bit is automatically set by the hardware (behaves as INI bit was set by software). When the sleep period is over and the sleep signal is deactivated, the module goes into the Init state and must be set to Run state by software, by writing **MFLAGSCL**.INIC=1.

At the state transition of Init -> Run the timeout timers and all the flags are reset. The RUN state starts with all channels inactive.

At the state transition SoftSuspend -> Run the timers are not reset but continue to run and the flags states are preserved, because the module continues operating from the point at which it was suspended.

When leaving the Run state towards Sleep the CTS signal is deactivated and the transmission of the pending commands is stopped.

When a soft suspend is requested, the module stops the transmission of the pending command and stream frames, waits until all expect flags are cleared, then deactivates the CTS signal and at the end acknowledges the

soft suspend request and sets the bit **OCS**.SUSSTA. Incoming target frames are served until the CTS signal stops the other side from sending new command and stream frames.

Hard Suspend request causes immediate switching off of the module clocks. After ending the Hard Suspend state both HSSL and HSCT modules must be reset by the application software and communication restarted from reset state.

Note: Reading and writing of registers is possible but will enable the kernel clock f_{CLC} for a few cycles. Attention: register accesses with clocking in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so a HSSL kernel reset might not be sufficient to bring the system into a defined state.

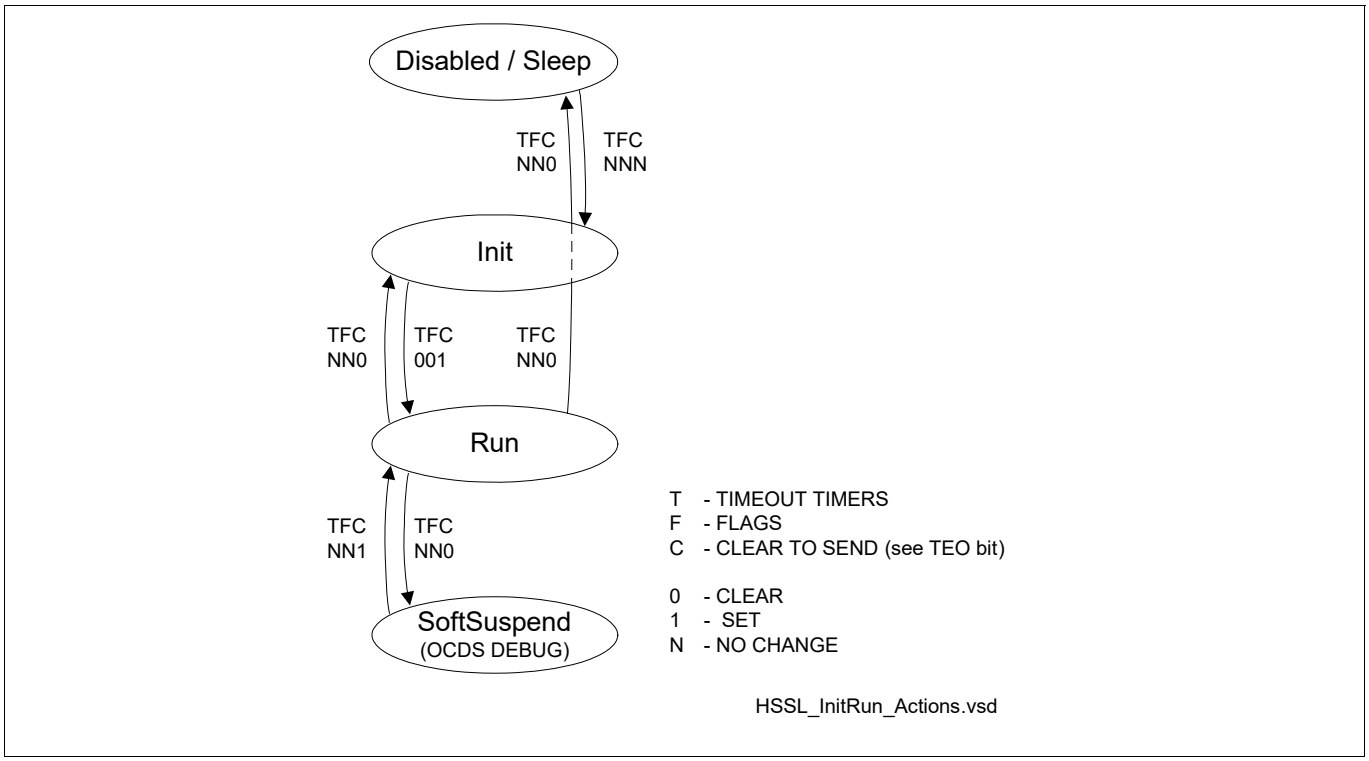


Figure 384 Actions on State Transitions

35.3.5 Interrupts

The HSSL module generates four types of interrupts per channel and one channel unspecific module interrupt.

Command Channel Interrupts

Each HSSL command channel generates four interrupts:

- Command OK interrupt COK
- Read Data interrupt RDI
- Error interrupt ERR
- Trigger interrupt, triggered by a trigger command frame TRG

The names of the interrupt signals are COK_INT, RDI_INT, ERR_INT and TRG_INT correspondingly.

An arrival of an error free frame response (ACK frame) triggers a COK interrupt, otherwise an ERR interrupt is triggered. An arrival of a read response frame triggers at the initiator side, additionally to the COK interrupt, an RDI interrupt.

An arrival of a trigger frame at the target side triggers a TRG interrupt there.

ERR interrupt is disjunct to COK and RDI the , meaning that for one frame, either ERR or COK and optionally RDI can occur. After an ERR interrupt, normal transmission must be resumed by software, because an optional DMA would remain not retriggered and would wait for COK indefinitely.

Stream Channel Interrupts

The HSSL stream transmit sub-channel generates three interrupts:

- TBE - Transmit Block End interrupt, shared with the COK interrupt of the command mode of the channel 2
- error interrupt ERR, shared with the ERR interrupt of the command mode of the channel 2
- RBE - Receive Block End interrupt, shared with the TRG interrupt of the command mode of the channel 2.

Exception Interrupt EXI

If the receive stage of the HSSL module detects a CRC error or any inconsistency in the received data the global EXI Interrupt is activated, which is not channel specific.

Note: The HSCT module uses channels A to D for the HSSL channels 0 to 3. So, HSCT channel 0100_B corresponds to 000_B (binary code) or 100_B (special code) in the HSSL header.

The EXI interrupt is also raised on an edge of the TEI signal (HSCT CTS output).

In total, the HSSL module provides 4*4 + 1 = 17 interrupt lines.

35.3.6 Operating a Command Channel

All HSSL command channel provide identical functionality.

35.3.6.1 Initiating a Single Write Command

In order to initiate a write command, the software must

- configure the data width and the type of the request which will follow (read or write)
- provide the new data (most frequently, but not necessary)
- provide new address

Write to the address register requests the predefined type of the transfer

35.3.6.2 Initiating a Single Read Command

In order to initiate a read command, the software must

- configure the data width and the type of the request which will follow (read or write)
- skip providing of the data step, or provide dummy data
- provide new address

Write to the address register requests the predefined type of the transfer

35.3.6.3 Initiating a Single Trigger Command

In order to initiate a trigger command, the software can either

- request a trigger type of frame and provide dummy data and address
- directly request a trigger frame by writing **ICONx (x=0-3)**.TQ

35.3.6.4 DMA Operated Command Queues

It is possible to use DMA to initiate lists of commands, by moving memory blocks containing the command lists to the HSSL module. In order to support such a way of operation, each channel provides three registers arranged in a particular way: first **IWDx (x=0-3)**, followed by **ICONx (x=0-3)** and **IRWAx (x=0-3)** at the end. For executing a list of write commands all three registers must be written for each command; for executing a list of read commands the first register **IWDx (x=0-3)** is optional (dummy write possible) but the last two **ICONx (x=0-3)** and **IRWAx (x=0-3)** are mandatory; for executing a trigger command only writing **ICONx (x=0-3)**.TQ is mandatory, but optionally even all three registers can be written: dummy write to **IWDx (x=0-3)**, write to **ICONx (x=0-3)**.RWT and write to **IRWAx (x=0-3)** to issue the request.

The answer data to a read command is available in the **IRDx (x=0-3)** register.

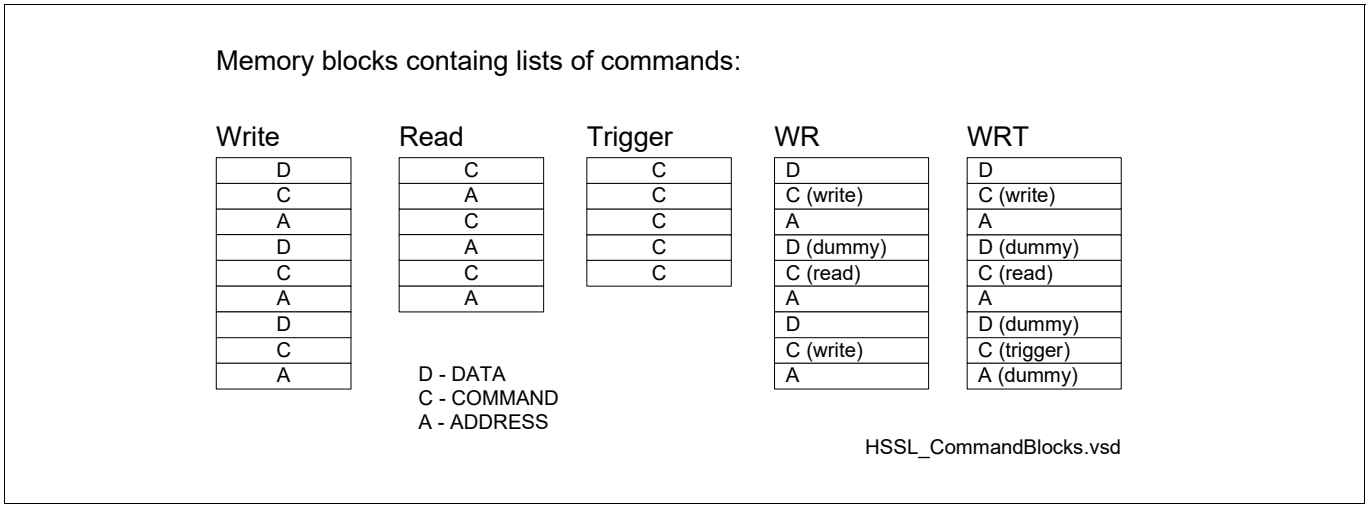


Figure 385 Command Queues

35.3.6.5 Receiver Error Handling

The HSSL receiver generates three error signals, two channel specific and one general:

- channel specific timeout and transaction tag errors and
- channel unspecific CRC error
- command frames, which pass the CRC check, containing an RFU command code are ignored
- Command frames, which pass the CRC check, containing invalid (greater than 3) channel number are ignored.

- If a command frame is received and the acknowledge of the previous command has not been sent yet, the new command frame is discarded.
- If a response frame comes, and the channel does not expect any more a response (the E flag is 0), than an incoming response is discarded without any error interrupts, additional to the timeout interrupt. This scenario can occur if a response is received after a timeout, and before the next command has been sent.
- If a channel in command mode receives a stream frame, the stream frame will be ignored.
- if a channel in streaming mode receives a command frame, the command frame will be ignored.

35.3.6.5.1 Timeout Error

Error triggered by the initiator.

35.3.6.5.2 Transaction Tag Error

Error triggered by the initiator.

35.3.6.6 Global Error

Global, channel not specific error is triggered if a CRC error, inconsistency between the HSCT and HSSL header, and SRI/SPB error occurs.

CRC error is triggered:

- by the target: when erroneous command frames has been received and
- by the initiator for erroneous response frames.

There is a dedicated interrupt named EXI which signals the CRC error. There is no indication if the CRC error was caused by a command or response frame. The CRC error caused by a response frame will be followed by a timeout interrupt. The received data is discarded; it is not available for the software in any register(s).

Additionally, the CRC error interrupt signals an inconsistency between the data length and channel number code delivered by the HSCT module and the information contained in the HSSL header. The differentiation between the two types of error can be done by reading the **MFLAGS**.CRCE, PIE1 and PIE2 flags. All three error types can occur and are detected independently of each other, so that more than one flag can be set at the same time.

An SRI error interrupt (EXI) is triggered if there was a transaction ID error, ECC error or SRI error acknowledge on the SRI bus. The flag **MFLAGS**.SRIE indicates this type of error. In case of an SRI read data ECC error, the READ_PH_ERR_ALARM signal reports this event to the SMU module.

An SPB error also triggers the global error interrupt (EXI) and sets the **MFLAGS**.SRIE flag.

35.3.7 Memory Block Transfer Modes of the Stream Channel

HSSL streaming channel is capable of transmitting one stream and receiving one stream in parallel. Streaming operates either in single block mode or continuous mode.

In single block mode, after triggering/enabling the streaming by using the **MFLAGS**.ISB/TSE bit, the SRI/SPB master of the HSSL module transmits/receives the preconfigured memory block, generates an interrupt signal and waits for the next trigger.

In continuous mode, after triggering/enabling the streaming by using the **MFLAGS**.ISB/TSE bit, the SRI/SPB master of the HSSL module transmits/receives two memory blocks of the same size in a round robin fashion indefinitely, or after being stopped by the software.

The streaming mode of the initiator/transmitter is configured by using the **CFG**.SMT bit. The streaming mode of the target/receiver is configured by using the **CFG**.SMR bit.

Triggering/enabling a block or continuous stream is done by using the Stream Block Request bit **MFLAGS**.ISB/TSE bit. In single block mode, the hardware resets this bit after performing a block transfer. In continuous mode, the transfer is ongoing continuously (**MFLAGS**.ISB/TSE remains set) and a stop of the transfer must be requested by writing one to **MFLAGSCL**.ISBC/TSEC bit.

On the initiator side, after completing the transfer of the current memory block that is currently read from the memory, the transmitter will be stopped and the **MFLAGS**.ISB will be cleared.

On the target side, after completing the transfer of the current memory block that is currently written to the memory, the receiver will be stopped and the **MFLAGS**.TSE will be cleared.

SRI/SPB error on the target triggers an error interrupt and the hardware stops. The TSE bit (and the RXFIFO) will be cleared by the hardware and the incoming frames are ignored.

Any error on the initiator side (like SRI/SPB error, target error, timeout, transaction tag error) triggers an error interrupt and the hardware stops streaming. The bit ISB (and the TXFIFO) will be cleared by hardware. All the incoming frames will be ignored, including the acknowledge frames of the previous frames.

Error can also occur at the beginning of a new memory block, while the last frames of the previous memory block are still in the TXFIFO. These frames are also lost, and the previous memory block remains incompletely received.

After an error while streaming in one direction the software must restart the streaming in that direction at both sides. The streaming will restart with a new block.

An error while streaming in most cases leaves the target waiting in the middle of a block. The target receiver can be stopped immediately (and afterwards restarted) by clearing the **MFLAGS**.TSE only when **TSFC**.CURCOUNT equals **TSFC**.RELCOUNT. Normally, CURCOUNT equals RELCOUNT before the start of a transfer of a block or between block transfers. Otherwise, while waiting in the middle of a block transfer, RELCOUNT can be made equal to CURCOUNT by writing the CURCOUNT value to RELCOUNT. This can be done either locally, by the target itself, or remotely, by the initiator using HSSL command frames.

The two initiator start addresses are configured in **ISSAx (x=0-1)**. The single block size is configured in the **ISFC** register. The current address being transferred is indicated in the read only (rh) **ISCA** register. The bit **MFLAGS**.IMB selects the corresponding **ISSAx (x=0-1)** register. This bit is not modified by the hardware in single block mode, and is toggled by the hardware in the continuous mode. At the start of the transfer, the selected start address **ISSAx (x=0-1)** is copied to the **ISCA** register.

The two target start addresses are configured in **TSSAx (x=0-1)**. The single block size is configured in the **TSFC** register. The current address being transferred is indicated in the read only (rh) **TSCA** register.The bit **MFLAGS**.TMB selects the corresponding **TSSAx (x=0-1)** register. This bit is not modified by the hardware in single block mode, and is toggled by the hardware in the continuous mode. At the start of the transfer, the selected start address **TSSAx (x=0-1)** is copied to the **TSCA** register.

The HSSL stream transmitter generates one dedicated interrupt and several error events:

- transmit block end interrupt, generated after the acknowledge of the last frame has been received.
- error events that can trigger a channel error interrupt (target, timeout, transaction tag, unexpected error)
- a global SRI/SPB error interrupt (check if the ISB bit is cleared by hardware)

The HSSL stream receiver generates one dedicated interrupt and several error events:

- receive block end interrupt event, generated after the last stream frame in a block has been written to the memory and the SRI/SPB master has received a confirmation (an acknowledge, but not an error).
- error event (target error that triggers an NACK frame)
- a SRI/SPB global error interrupt (check if the TSE bit is cleared by hardware)

As long as the ISB bit is low, the TXFIFO and the expect FIFO are cleared and kept in the empty state.

As long as the TSE bit is low, the RXFIFO is cleared and kept in the empty state.

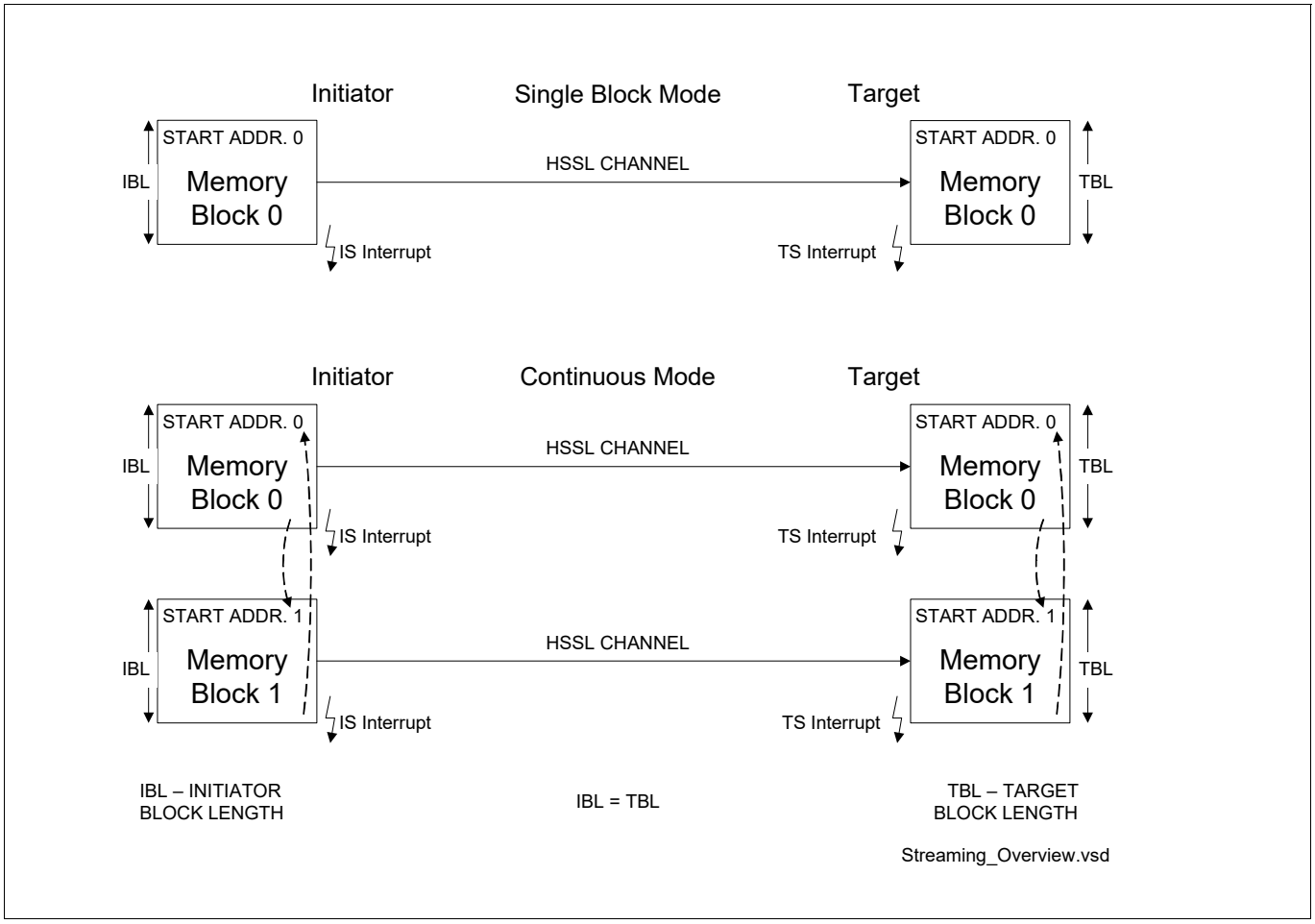


Figure 386 Stream Modes of Operation

35.3.8 HSSL Reset

Both HSCT and the HSSL modules must be reset together.

35.3.9 OCDS SRI / SPB Master Suspend

The SRI /SPB master access on the bus can be suspended by using the OCDS suspend request. When the master reaches idle state the suspend acknowledge signal is activated.

35.3.10 OCDS Trigger Sets

In order to support the debugging activities, the HSSL module provides a set of internal signals to the on-chip debug system OCDS. An overview of this feature is shown in **Figure 387**. Its configuration is done by using the bits **OCS**.TGS and TGB.

An edge on any module internal signal belonging to the selected set going out on one of the two OTGB busses triggers an action of the OCDS system.

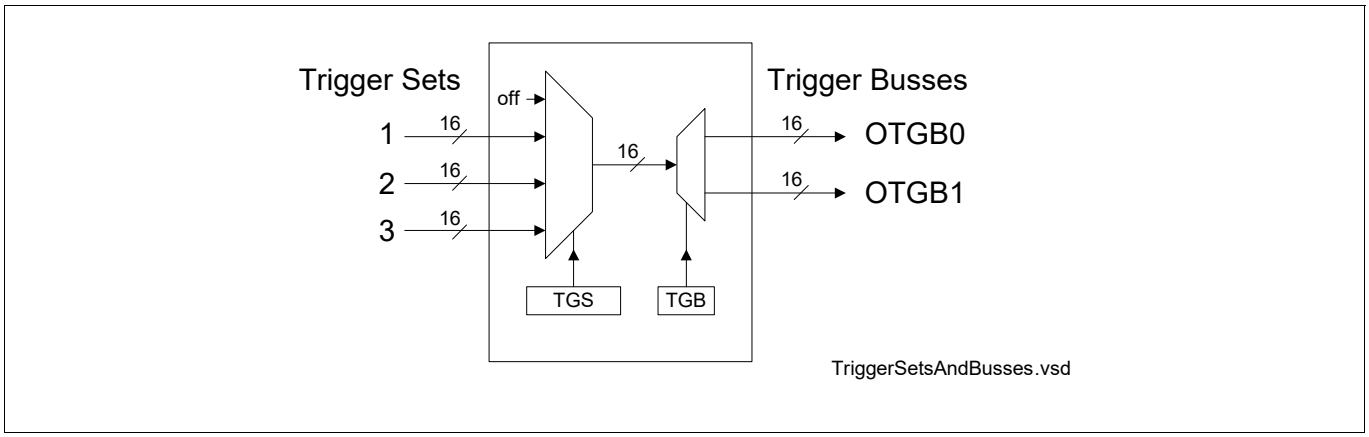


Figure 387 Overview of the Trigger Sets and Busses

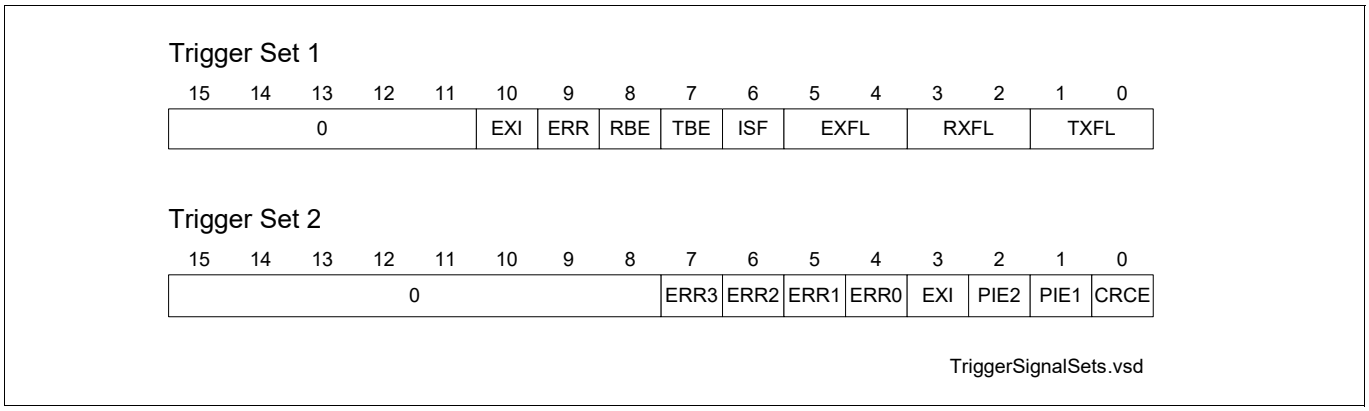


Figure 388 Overview of the Trigger Signal Sets

Table 309 HSSL Trigger Sets

Name	Description
TS16_STR	Streaming Channel Trigger Set (Trigger Set 1)
TS16_ERR	Error Trigger Set (Trigger Set 2)

The trigger sets 1 and 2 are used, the set 3 is not used (padded with 0).

Set 1, streaming channel debugging:

- 2 bits: TXFIFO filling level **SFSFLAGS**.TXFL
- 2 bits: RXFIFO filling level **SFSFLAGS**.RXFL
- 2 bits: Expect FIFO filling level **SFSFLAGS**.EXFL
- 1 bit: Initiator Stream Frame Request **SFSFLAGS**.ISF
- 4 bits: streaming channel interrupt signals TBE, RBE, ERR, EXI, see **Interrupts**.

- others: not used

Set 2, timeout errors caused by unspecific errors debugging:

- 1 bit: CRC error **MFLAGS**.CRCE
- 1 bit: Phy Inconsistency error, inconsistent channel number code, **MFLAGS**.PIE1
- 1 bit: Phy Inconsistency error, inconsistent data length, **MFLAGS**.PIE2
- 5 bits: error interrupt signals 1 x EXI and 4 x ERR, see **Interrupts**.
- others: not used

Note: The signal lists from above are mapped to the trigger sets in the following way: first item from the list to the trigger signal 0,… to 15.

35.3.11 Access Protection

The HSSL module provides memory access protection in form of four memory access windows. Each window can be located anywhere within the address space, having an arbitrary size with the granularity of 256 bytes.

Each window defines a memory range where an access is allowed. All four windows create a sort of an access filter that protects the memory from external writing or reading. A window can be a read only, write only or read and write window.

Each window is defined with:

- window start address, see register **AWSTARTi (i=0-3)**
- window end address, see register **AWENDi (i=0-3)** and
- access rule: read only, write only or read and write, see register **AR**

If two access windows overlap, one read only and one write only, the common address range is read and write accessible. If no access window overlaps with r, w or rw window, the r, w or rw window wins. No access window means window disabled.

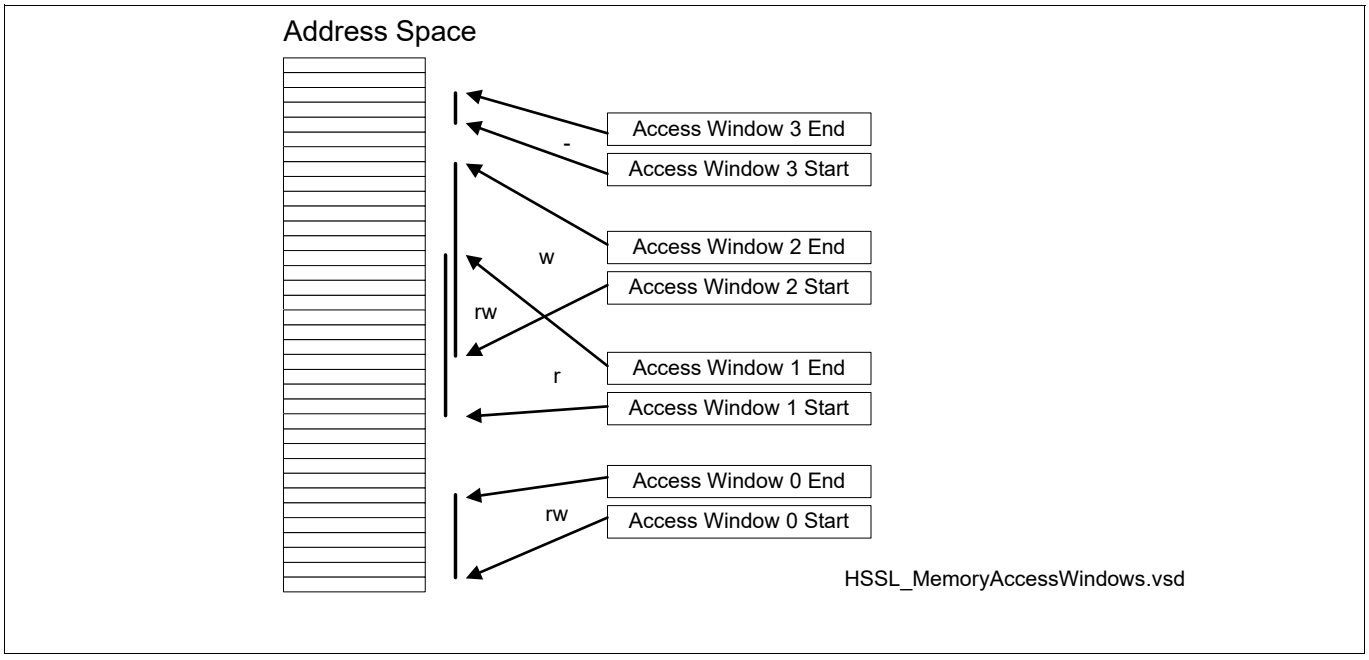


Figure 389 HSSL Access Windows

Normally, it is expected that the End Address is higher then or equal to the Start Address. Otherwise, the window is invalid and no commands can pass through the up-side-down range.

The **AWSTARTi (i=0-3)** registers define the first address of the first 256-byte block of an address window. The **AWENDi (i=0-3)** registers define the first address of the last 256-byte block of an address window. The following diagram shows the details of the address window definition.

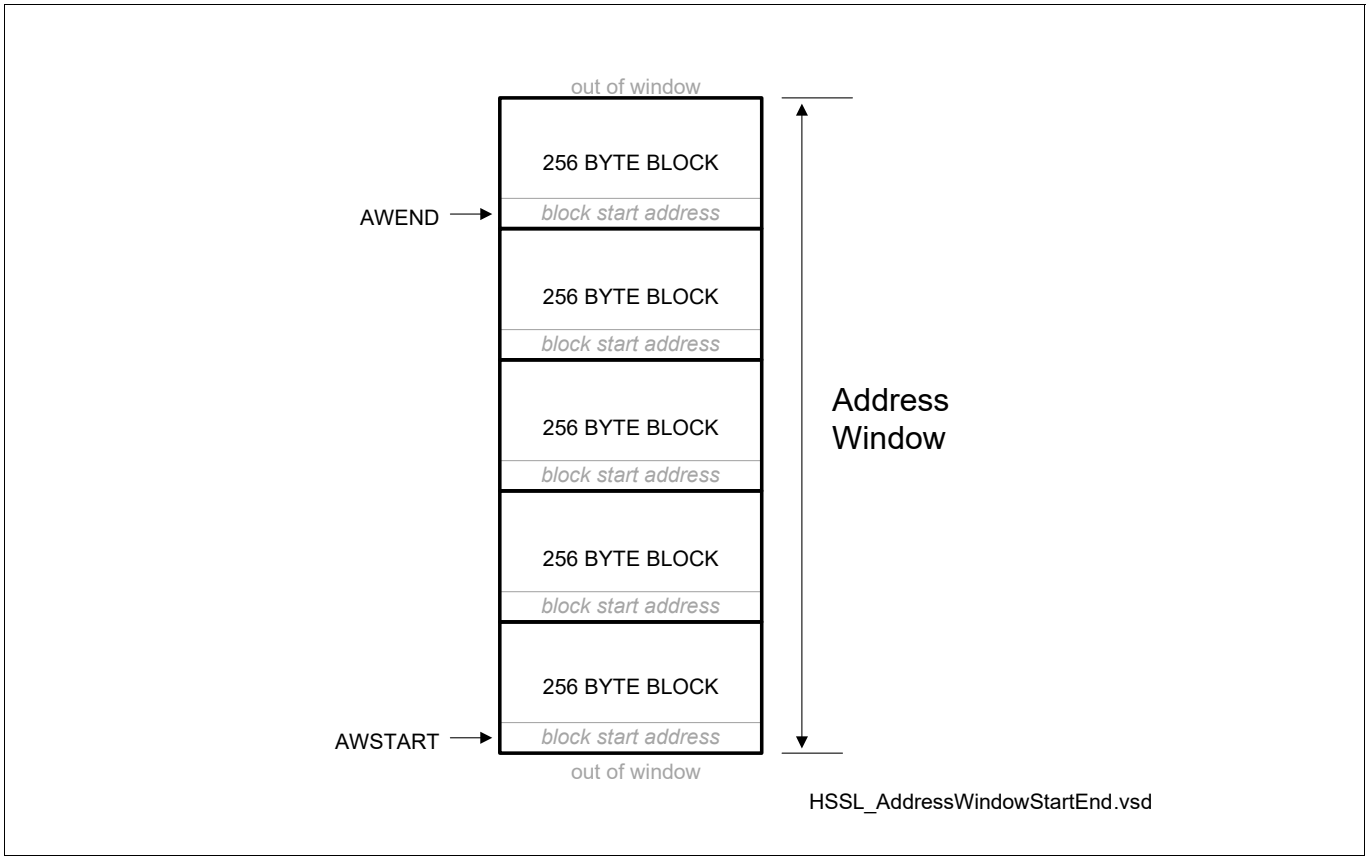


Figure 390 Access Windows Definition

These command frames are subject to filtering:

- Register Read
- Register Write and
- Stream Frame

These command always pass through:

- Read Answer
- ACK
- NACK
- Trigger
- ID Request

Command Frames Access Protection

If a HSSL target receives a read or write command frame with an address which passes through one of the windows, the command is executed and a response frame is sent back.

If a HSSL target receives a command frame with an address which does not pass through any windows, the command is not executed, a NACK response is sent back. The memory access violation flag **MFLAGS**.MAV is set, the access violating channel is captured in the bit field **AR**.MAVCH, and the EXI interrupt is triggered in the target.

Streaming Access Protection

In contrast to the read and write command frames, the stream frames do not contain an address information. There are two cases to distinguish, target side and initiator side.

At the target side, when the streaming starts, the initiator has already programmed the target addresses in the **TSSAx (x=0-1)** registers by using write frames, which pass through and carry the stream address in their data field. Therefore, the memory filtering uses the current access address, visible in the **TSCA** register, and if the current streaming address passes through the access protection filter, the read or write is executed, else a NACK frame is sent as a response.

If a HSSL target receives a stream frame and the current access address does not pass through any windows, the access is not executed, a NACK response is sent back. The TSE bit will be cleared by hardware and the incoming frames will be ignored. The memory access violation flag **MFLAGS**.MAV is set, the access violating channel is captured in the bit field **AR**.MAVCH, and the EXI interrupt is triggered in the target.

At the initiator side, the **ISSAx (x=0-1)** registers, which are externally accessible, define the memory range to be transmitted. Therefore, the access filtering at the initiator side covers the streaming read accesses by using the current access address, visible in the **ISCA** register.

At the initiator side, access violation stops the streaming, and the ISB bit is cleared by hardware. The memory access violation flag **MFLAGS**.MAV is set, the access violating channel is captured in the bit field **AR**.MAVCH, and the EXI interrupt is triggered.

Public and Private Registers

The HSSL module contains two types of registers:

- public registers, which can be both read and written by the HSSL module itself
- private registers, which can be only read by the HSSL module itself

This behavior of the HSSL module registers is implemented by using the User Mode (U) and Supervisor Mode (SV) of writing registers. The private registers are writable in SV mode, the public registers are writable in U and SV mode.

The memory access protection registers **AWSTARTi (i=0-3)**, **AWENDi (i=0-3)** and **AR** including the **TIDADD** register are all writable only in Supervisor Mode, and are consequently private. The HSSL itself makes only User Mode accesses, so that it can not write these registers.

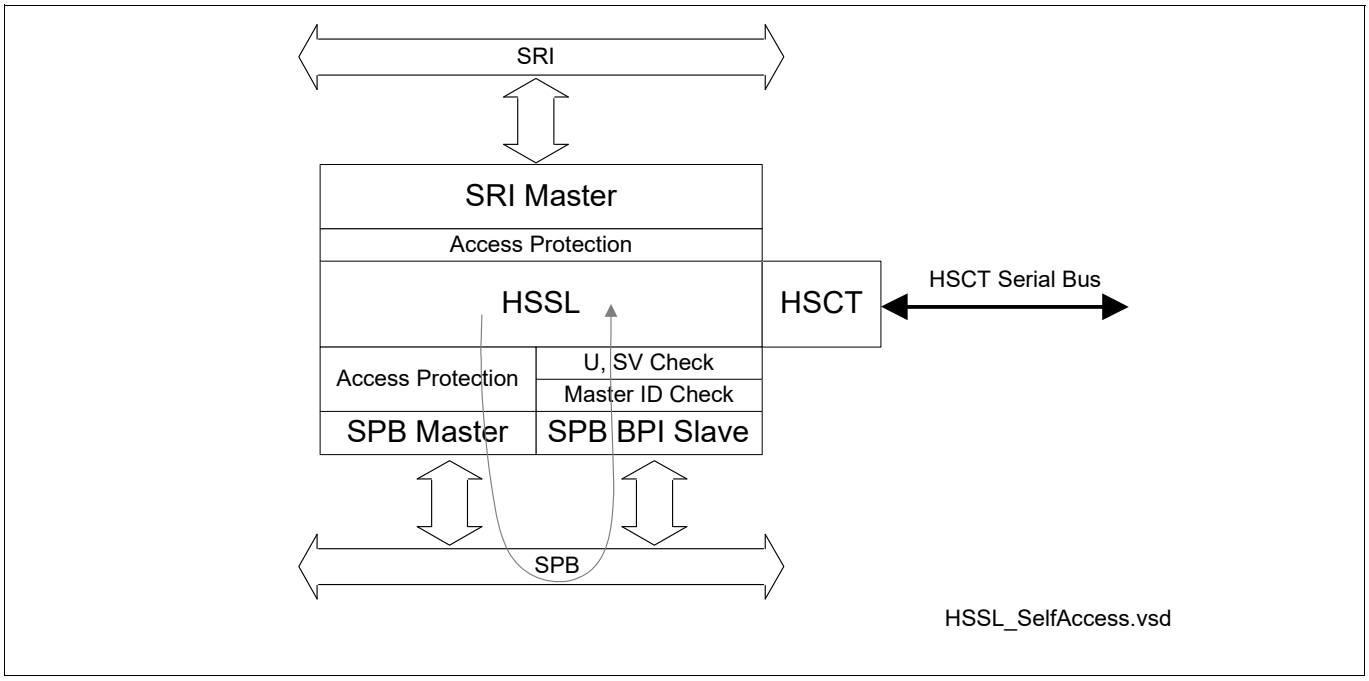


Figure 391 HSSL access to its own registers

35.3.12 Multi-Slave Operation

In multi slave use-cases, one HSSL master can communicate with up to three slaves. The master generates the reference clock from the HSCT module point of view and selects the active counterpart for the connection. The slaves receive the reference clock and are being selected by the master one at a time.

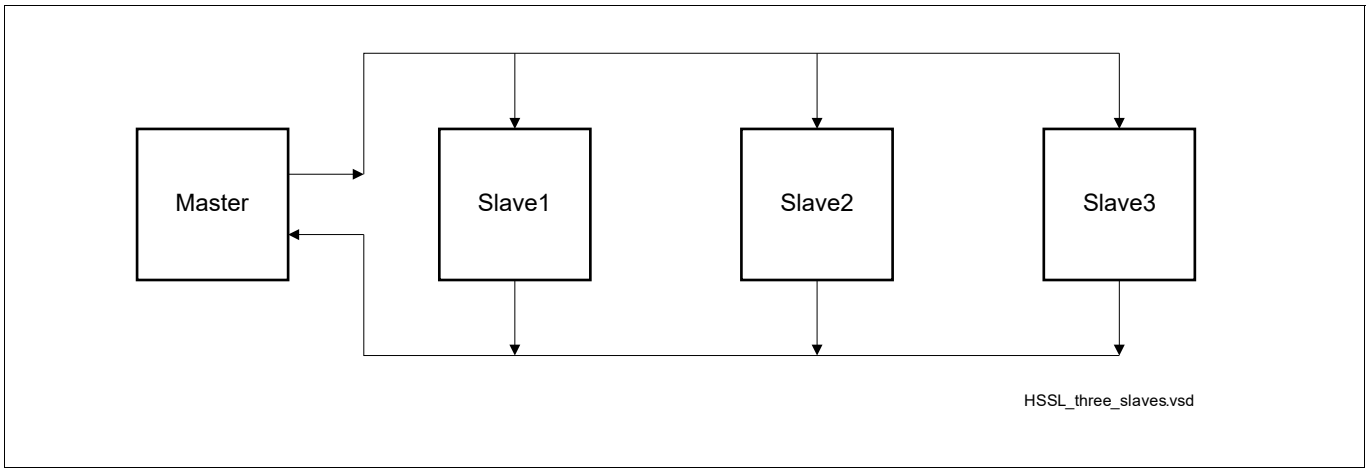


Figure 392 Three Slave HSSL Connection

The communication is connection based - at any point in time there can be only one active connection (or zero) between the master and one slave. There are no broadcast or multicast messages on the HSSL level. There is a defined procedure for terminating a connection with one slave and activating a connection with another slave. During an active connection both master and slave can be initiators and targets.

After power-on reset, the transmit outputs of the slaves and the transmit output of the master are high impedance. Up to three slaves can be enabled for reception by software.

35.3.12.1 Slave Tag and Slave Control

The multi slave operation uses the identical set of command and stream frames as the standard HSSL two-device operation. The slaves are selected with two bits in the HSSL header. These bits are used to filter the frames which are relevant for a slave by comparing them with the slave internal two bit tags, see **MSCR**.SLAVETAG.

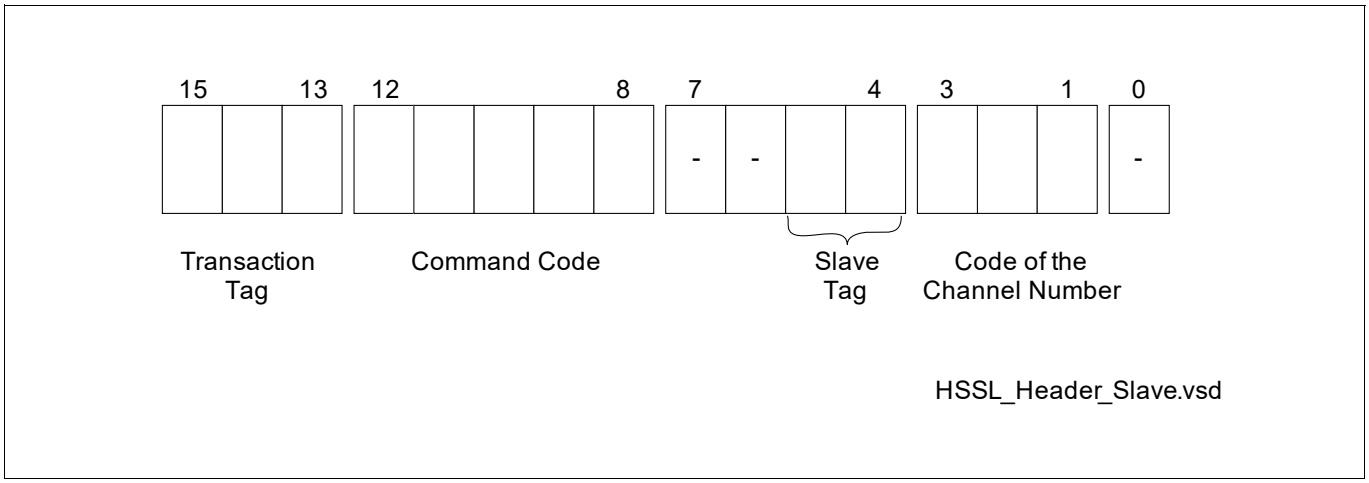


Figure 393 HSSL Header with Slave Tag

35.3.12.2 Slave Tag Frame Filter and Translator

The filter is located after the CRC block. In receive direction, if there is a match, the frame is passed through and handled in a standard HSSL two-device-connection way. If no match, the frames are discarded. In transmit direction, the headers of transmitted frames get the appropriate tag inserted. In case of a CRC error in a frame sent by the master, all slaves may detect the CRC error. The CRC checker discards the frames with CRC error, and even the addressed slave does not get the frame with a CRC error delivered. Frames with PIE1 and PIE2 error are also discarded. The corresponding error interrupts can be triggered if enabled.

The slaves have their tags fixed in the initialization phase and they never change. The software of the master changes its slave tag when establishing connection with different slave. First the HSCT link must be established. The master must have its **MSCR**.SLAVETAG configured before starting with sending HSSL commands or stream frames to the appropriate slave.

The **MSCR**.EN and **MSCR**.SLAVETAG bit fields are propagated to the HSCT module in order to ensure consistent operation of the whole communication channel.

If the multi slave operation is disabled (**MSCR**.EN=0), then both the injection of the slave tag in transmit direction and the filtering of the slave frames in the receive direction are not performed. The header remains unchanged and all the incoming frames pass through.

It is not intended to set the **MSCR**.ITXSTOP bit in the master.

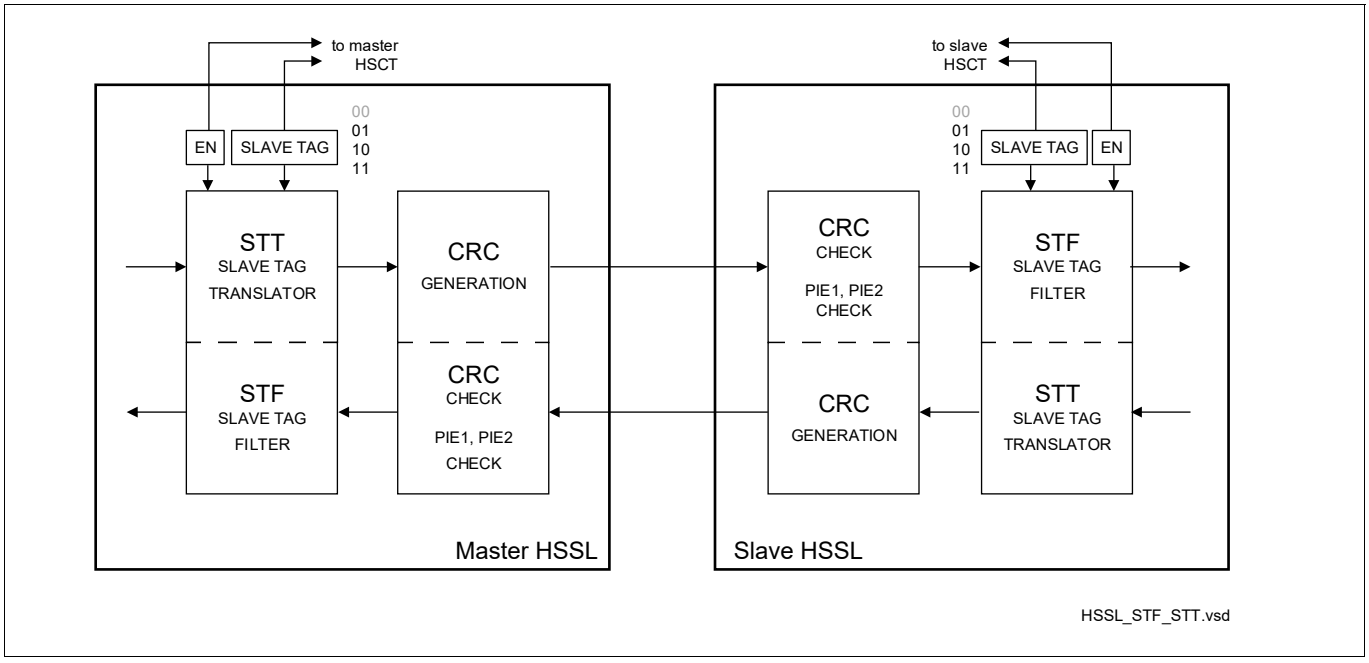


Figure 394 Slave Frame Filter

35.3.12.3 Activating a Slave

Activating a slave is done in two steps:

- The master enables the LVDS transmitter pads of one slave by using slave specific HSCT command. The Tx enable HSCT commands for the slaves 1, 2 and 3 have payloads 51_H, 52_H and 53_H correspondingly.
- The second step is either soft or hard HSSL activation:
  - Soft activation - the master notifies the slave by using standard HSSL commands, according to a pre-defined procedure. For example, master can issue a write command to a certain location, send an interrupt trigger command frame, and a slave having a CPU can start transmitting using the HSSL module.

– Hard activation - the master clears the **MSCR**.ITXSTOP bit by using standard HSSL 16-bit write command frame (an appropriate address window for the HSSL registers must have been already set up). In the next step, the master notifies the slave, for example with a trigger command, or enables the operation of the slave by setting an appropriate bit(s). The slave continues the operation from the point its initiator transmission has been stopped.

35.3.12.4 Deactivating a Slave

Master can deactivate a slave in two ways, soft and hard.

Soft Deactivation

Soft deactivation procedure uses software handshake procedure to bring the slave in a state where all ongoing slave initiator transactions has been completed, including streams, no new slave transactions are initiated and the slave notifies the master that the safe deactivation state has been reached. Subsequently, the master disables the LVDS pad.

The soft deactivation sequence consists of:

- Soft HSSL deactivation - using software handshake with standard HSSL commands, like writing to memory location and triggering interrupts
- Switching off the LVDS Tx pads with slave unspecific (broadcast) HSCT command with payload 32_H

Hard Deactivation

Hard deactivation is intended to provide faster deactivation with bounded reaction time, primarily for slaves without CPU and use-cases without streaming.

The hard deactivation sequence consists of:

- Hard HSSL deactivation
  - the master sets the **MSCR**.ITXSTOP bit by using standard HSSL 16-bit write command frame (an appropriate address window for the HSSL registers must have been already set up)
  - The master must assure that slave initiator commands have been processed by itself and that the acknowledge frames have been sent to the slave. For example, the master polls the slave E (expect) flags by using standard HSSL 16-bit read commands until it reads back 0.
  - Additionally, the master must assure that its command frames have been processed and acknowledged by the slave. For example, the master can poll its own expect flags until it reads 0.
- Switching off the LVDS Tx pads with slave unspecific (broadcast) HSCT command with payload 32_H

The slave HSSL module initiator transmit path remains inactive and later, after being re-activated by the master, continues the operation from the point where it has been stopped. The master is not allowed to send commands to a deactivated slave.

35.3.12.5 MSCR HSSL to HSCT Connections

The HSSL module provides the **MSCR**.EN and **MSCR**.SLAVETAG bit fields at the module border as outputs. These signals are connected to the HSCT module and provide single point of configuring the behavior of both HSSL and HSCT regarding multi slave behavior.

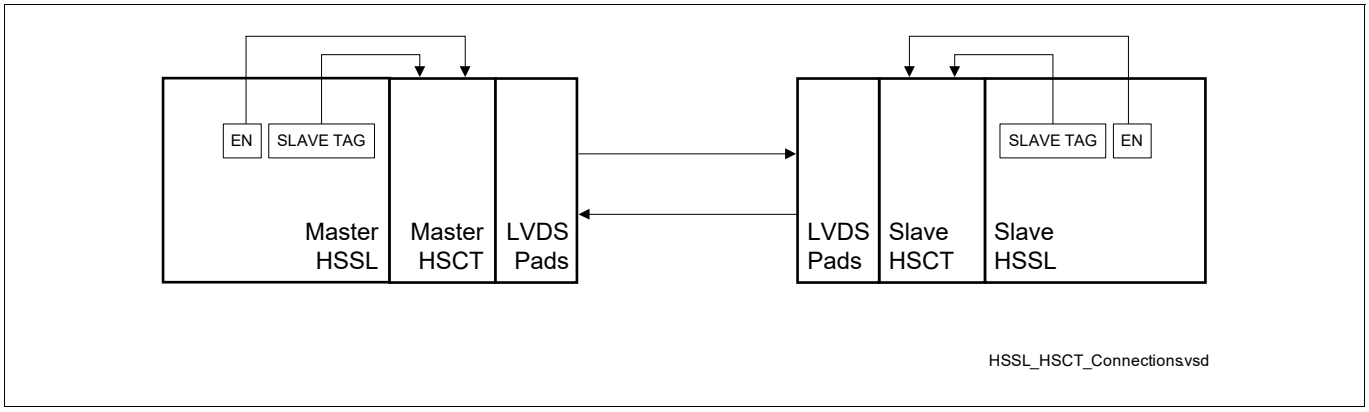


Figure 395 HSSL to HSCT Connections

35.4 Registers

This section describes the registers of the HSSL module.

Note: The HSSL module occupies 1KByte of address space, although only the first 256 bytes are used. Accessing a not used address location generates a bus error.

There are no registers in the HSSL with the destructive read property, where a read access would generate some action in the hardware.

Table 310 Register Overview - HSSL (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	000_H	U,SV	SV,E,P	Application Reset	77
ID	Module Identification Register	008_H	U,SV	BE	Application Reset	50
CRC	CRC Control Register	$00C_H$	U,SV	SV,E,P	Application Reset	50
CFG	Configuration Register	010_H	U,SV	U,SV,P	Application Reset	51
QFLAGS	Request Flags Register	014_H	U,SV	U,SV,P	Application Reset	52
MFLAGS	Miscellaneous Flags Register	018_H	U,SV	U,SV,P	Application Reset	54
MFLAGSSET	Miscellaneous Flags Set Register	$01C_H$	U,SV	U,SV,P	Application Reset	56
MFLAGSCL	Miscellaneous Flags Clear Register	020_H	U,SV	U,SV,P	Application Reset	58
MFLAGSEN	Flags Enable Register	024_H	U,SV	U,SV,P	Application Reset	61
SFSFLAGS	Stream FIFOs Status Flags Register	028_H	U,SV	U,SV,P	Application Reset	63
IWDx	Initiator Write Data Register x	030_H+x*10_H	U,SV	U,SV,P	Application Reset	64
ICONx	Initiator Control Data Register x	034_H+x*10_H	U,SV	U,SV,P	Application Reset	64
IRWAx	Initiator Read Write Address Register x	038_H+x*10_H	U,SV	U,SV,P	Application Reset	65
IRDx	Initiator Read Data Register x	$03C_H$+x*10_H	U,SV	U,SV,P	Application Reset	66
TCDi	Target Current Data Register i	070_H+i*8	U,SV	U,SV,P	Application Reset	67
TCAi	Target Current Address Register i	074_H+i*8	U,SV	U,SV,P	Application Reset	67

Table 310 Register Overview - HSSL (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
TSTAT	Target Status Register	090_H	U,SV	U,SV,P	Application Reset	67
TIDADD	Target ID Address Register	094_H	U,SV	SV,P	Application Reset	68
SEC	Security Control Register	098_H	U,SV	U,SV,P	Application Reset	75
MSCR	Multi Slave Control Register	$09C_H$	U,SV	U,SV,P	Application Reset	76
ISSAx	Initiator Stream Start Address Register	$0A0_H$+x*4	U,SV	U,SV,P	Application Reset	69
ISCA	Initiator Stream Current Address Register	$0A8_H$	U,SV	U,SV,P	Application Reset	69
ISFC	Initiator Stream Frame Count Register	$0AC_H$	U,SV	U,SV,P	Application Reset	70
TSSAx	Target Stream Start Address Register x	$0B0_H$+x*4	U,SV	U,SV,P	Application Reset	71
TSCA	Target Stream Current Address Register	$0B8_H$	U,SV	U,SV,P	Application Reset	71
TSFC	Target Stream Frame Count Register	$0BC_H$	U,SV	U,SV,P	Application Reset	72
AWSTARTi	Access Window Start Register i	$0C0_H$+i*8	SV	SV,P	Application Reset	73
AWENDi	Access Window End Register i	$0C4_H$+i*8	SV	SV,P	Application Reset	73
AR	Access Rules Register	$0E0_H$	SV	SV,P	Application Reset	74
OCS	OCDS Control and Status	$0E8_H$	U,SV	SV,P	Debug Reset	78
KRSTCLR	Kernel Reset Status Clear Register	$0EC_H$	U,SV	SV,E,P	Application Reset	81
KRST1	Kernel Reset Register 1	$0F0_H$	U,SV	SV,E,P	Application Reset	81
KRST0	Kernel Reset Register 0	$0F4_H$	U,SV	SV,E,P	Application Reset	80
ACCEN1	Access Enable Register 1	$0F8_H$	U,SV	SV,SE	Application Reset	79
ACCEN0	Access Enable Register 0	$0FC_H$	U,SV	SV,SE	Application Reset	79

Note: ***AWSTARTi (i=0-3)***, ***AWENDi (i=0-3)***, ***AR****: read access mode is SV only; these registers deliver 0 to a read access in User Mode, but BE is not triggered.*

List of Access Protection Abbreviations

U - User Mode

SV - Supervisor Mode

BE - Bus Error

nBE - no Bus Error

P - Access Protection, as defined by the ACCEN Register

E - ENDINIT

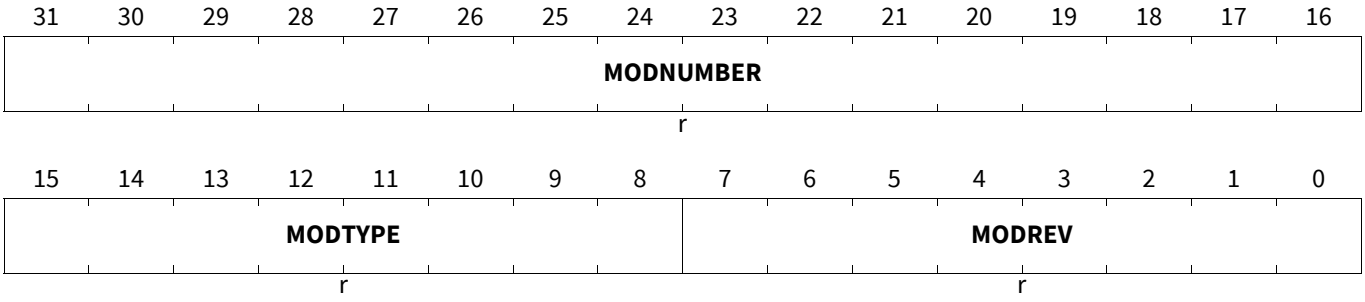
SE - Safety ENDINIT

35.4.1 Global Registers

Module Identification Register

The Module Identification Register ID contains read-only information about the module version.

ID
Module Identification Register **(008_H)** **Application Reset Value: 00CB C0XX$_H$**



Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** MODREV defines the module revision number. The value of a module revision starts with 01_H (first revision).
MODTYPE	15:8	r	**Module Type** This bit field is $C0_H$. It defines a 32-bit module.
MODNUMBER	31:16	r	**Module Number Value** This bit field together with MODTYPE uniquely identifies a module.

CRC Control Register

This register only influences the generation of the CRC in the transmit direction.

CRC
CRC Control Register **($00C_H$)** **Application Reset Value: 0000 0000$_H$**

Field	Bits	Type	Description
XORMASK	15:0	rw	**Value to be XORed with the Calculated CRC** Used for error injection / test purposes.
XEN	16	rw	**Enable the Error Injection via XORMASK** 0_B disabled 1_B enabled

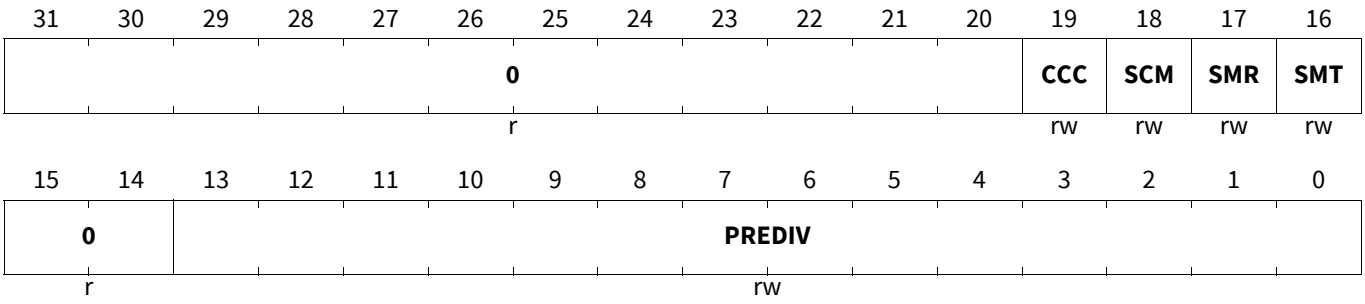
Field	Bits	Type	Description
0	31:17	r	**Reserved**

Configuration Register

The CFG register is used to configure the HSSL module. It shall only be written in the Initialize Mode.

Note: The duration of a timeout interval may vary for an amount of one predivider clock period. Therefore the predivider should be set as low as possible, and the timeout length as many predivider periods as possible.

CFG
Configuration Register **(010_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
PREDIV	13:0	rw	**Global Predivider** Defines the down-scaled module clock to be used by all channel timeout timers. 0000_H1 0001_H2 0002_H3 **others**, corresponding down-scale factor
SMT	16	rw	**Streaming Mode Transmitter** 0_B Continuous 1_B Single
SMR	17	rw	**Streaming Mode Receiver** 0_B Continuous 1_B Single
SCM	18	rw	**Streaming Channel Mode** Defines if the channel 2 is used in a streaming or command mode. 0_B Command 1_B Streaming
CCC	19	rw	**Channel Code Control** Defines the coding of the channel number in the HSSL header. 0_B Binary 1_B Special
0	15:14, 31:20	r	**Reserved**

35.4.2 Channel.Flags Registers

Request Flags Register

This register contains flags indicating if an appropriate action request is pending or not.

QFLAGS

Request Flags Register **(014_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ES	**TS**	**RS**	**IS**	**0**				**E3**		**E2**		**E1**		**E0**	
rh	rh	rh	rh	r				rh		rh		rh		rh	

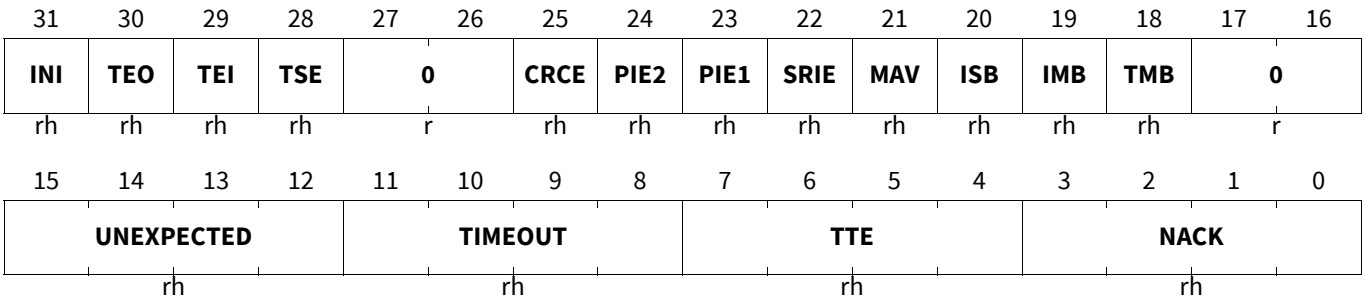
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**R**				**T**				**I**			
r				rh				rh				rh			

Field	Bits	Type	Description
I	3:0	rh	**Request Flags for Initiated Commands** These flags are set by the corresponding channel when a WRTS command is initiated. The WRT commands are initiated via the SPB bus. The S(tream) commands are initiated by the module internally, by the TXFIFO, except for the first stream frame start, which is done via the SPB bus. See MFLAGS.ISB and ISF flags. 0_H No request pending 1_H Channel 0 request pending 2_H Channel 1 request pending 4_H Channel 2 request pending 8_H Channel 3 request pending **others**, corresponding combination of requests
T	7:4	rh	**Request Flags for Commands Arrived at Target** These flags are set by the hardware according to the header information of a frame arrived at the target without a CRC error. They are used by the arbiter of the SRI/SPB master and cleared when the appropriate command is executed. 0_H No request pending 1_H Channel 0 request pending 2_H Channel 1 request pending 4_H Channel 2 request pending 8_H Channel 3 request pending **others**, corresponding combination of requests

Field	Bits	Type	Description
R	11:8	rh	**Request Flags for Response Frames at the Target** After a command has been executed by the SRI/SPB master, an appropriate flag is being set which indicates that an ACK/NACK/DATA is pending. 0_H No request pending 1_H Channel 0 request pending 2_H Channel 1 request pending 4_H Channel 2 request pending 8_H Channel 3 request pending **others**, corresponding combination of requests
E0	17:16	rh	**Expect Flags for Activated Timeout Timer 0** An appropriate two bit flag is set by the hardware when a timeout timer for a channel is started. The hardware clears the appropriate flag when any response frame arrives at the initiator. In case of an unexpected response the flag UNEXPECTED is additionally set. 00_B No request pending 01_B write request pending 10_B read request pending 11_B trigger request pending
E1	19:18	rh	**Expect Flags for Activated Timeout Timer 1**
E2	21:20	rh	**Expect Flags for Activated Timeout Timer 2**
E3	23:22	rh	**Expect Flags for Activated Timeout Timer 3**
IS	28	rh	**I Flag for Stream Frames** See the “I” flag description above. 0_B No request pending 1_B Request pending
RS	29	rh	**R Flag for Stream Frames** See the “R” flag description above. 0_B No request pending 1_B Request pending
TS	30	rh	**T Flag for Stream Frames** See the “T” flag description above. 0_B No request pending 1_B Request pending
ES	31	rh	**E Flag for Stream Frames** See the “E” flag description above. 0_B No request pending 1_B Request pending
0	15:12, 27:24	r	**Reserved**

Miscellaneous Flags Register

MFLAGS
Miscellaneous Flags Register **(018_H)** **Application Reset Value: $8000\ 0000_H$**



Field	Bits	Type	Description
NACK	3:0	rh	**Not Acknowledge Error - Target Error** Indicates for each channel that a target error frame has been received.
TTE	7:4	rh	**Transaction Tag Error** Indicates for each channel if a CRC correct acknowledge frame with an unexpected transaction tag number has been received.
TIMEOUT	11:8	rh	**Timeout Error** Indicates for each channel if an timeout event has occurred.
UNEXPECTED	15:12	rh	**Unexpected Type of Frame Error** Indicates for each channel if an unexpected or inappropriate response is received. For example a NACK for a Trigger frame or DATA for WRITE frame.
TMB	18	rh	**Target Memory Block** Selects the currently active memory block used by the target as a target for the streaming data, with its start address and frame counter. Switching the active block in the middle of a block transfer is not allowed. 0_B Memory block 0 1_B Memory block 1
IMB	19	rh	**Initiator Memory Block** Selects the currently active memory block used by the initiator as a source for the streaming data, with its start address and frame counter. Switching the active block in the middle of a block transfer is not allowed. 0_B Memory block 0 1_B Memory block 1
ISB	20	rh	**Initiator Stream Block Request** Indicates if stream block request is pending. Set by the software to start a stream block transfer by using the MFLAGSSET.ISBS bit; clear by the software possible (if needed) by using the MFLAGSCL.ISBC bit; cleared by hardware at the end of the current block transfer in single mode, but not in continuous mode. 0_B No request or streaming ongoing 1_B Streaming ongoing

Field	Bits	Type	Description
MAV	21	rh	**Memory Access Violation** Indicates a memory access violation. 0_B No violation 1_B Violation
SRIE	22	rh	**SRI/SPB Bus Access Error** Indicates an error on the SRI bus - transaction ID, ECC error or error acknowledge. Indicates an error on the SPB bus, error acknowledge or timeout. 0_B No error 1_B Error
PIE1	23	rh	**PHY Inconsistency Error 1(Channel Number Code Error)** Indicates if HSCT to HSSL channel number code comparator has detected an inconsistency error. 0_B No error 1_B Error
PIE2	24	rh	**PHY Inconsistency Error 2(Data Length Error)** Indicates if HSCT to HSSL data length comparator has detected an inconsistency error. 0_B No error 1_B Error
CRCE	25	rh	**CRC Error** Indicates if CRC checker has detected a CRC error. 0_B No error 1_B Error
TSE	28	rh	**Target Stream Enable** Used by the hardware to handle the single and continuous streaming. In single mode, cleared by hardware after the current block transfer ends. The module ignores afterwards the incoming steam frames. 0_B Disabled 1_B Enabled
TEI	29	rh	**Transmit Enable Input** Indicates the state of the TEI input signal of the HSSL module, which is driven by the CTS output signal of the HSCT module. Any edge on this signal triggers an EXI interrupt. This low level signal stops the transmission of both command and response frames.
TEO	30	rh	**Transmit Enable Output** Indicates the state of the TEO output signal of the HSSL module, which drives thee CTS input signal of the HSCT module. This bit is cleared by hardware at entering the INIT and Soft Suspend state.
INI	31	rh	**Initialize Mode** Indicates if the module is in the Initialize or Run mode. 0_B Run mode 1_B Initialize mode
0	17:16, 27:26	r	**Reserved**

Miscellaneous Flags Set Register

Note: Read access to this register returns all zero.

MFLAGSSET
Miscellaneous Flags Set Register (01C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INIS	TEOS	0	TSES	0		CRCES	PIE2S	PIE1S	SRIES	MAVS	ISBS	IMBS	TMBS	0	
w	w	r	w	r		w	w	w	w	w	w	w	w	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNEXPECTEDS				TIMEOUTS				TTES				NACKS			
w				w				w				w			

Field	Bits	Type	Description
NACKS	3:0	w	**NACK Flags Set** Writing 1 sets the corresponding bit in the MFLAGS register and triggers the ERR interrupt for the corresponding channel, if enabled in the MFLAGSEN register. Writing 0 has no effect. 0_H no action 1_H set channel 0 2_H set channel 1 4_H set channel 2 8_H set channel 3 **others**, set the corresponding combination of channels
TTES	7:4	w	**Transaction Tag Error Flags Set** Writing 1 sets the corresponding bit in the MFLAGS register and triggers the ERR interrupt for the corresponding channel, if enabled in the MFLAGSEN register. Writing 0 has no effect. 0_H no action 1_H set channel 0 2_H set channel 1 4_H set channel 2 8_H set channel 3 **others**, set the corresponding combination of channels
TIMEOUTS	11:8	w	**Timeout Error Flags Set** Writing 1 sets the corresponding bit in the MFLAGS register and triggers the ERR interrupt for the corresponding channel, if enabled in the MFLAGSEN register. Writing 0 has no effect. 0_H no action 1_H set channel 0 2_H set channel 1 4_H set channel 2 8_H set channel 3 **others**, set the corresponding combination of channels

Field	Bits	Type	Description
UNEXPECTEDS	15:12	w	**Unexpected Error Flags Set** Writing 1 sets the corresponding bit in the MFLAGS register and triggers the ERR interrupt for the corresponding channel, if enabled in the MFLAGSEN register. Writing 0 has no effect. 0_H no action 1_H set channel 0 2_H set channel 1 4_H set channel 2 8_H set channel 3 **others**, set the corresponding combination of channels
TMBS	18	w	**Target Memory Block Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Set
IMBS	19	w	**Initiator Memory Block Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Set
ISBS	20	w	**Initiator Stream Block Request Set** Writing 1 sets the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Set
MAVS	21	w	**MAV Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register, and generates an EXI interrupt if enabled in the corresponding bit of the MFLAGSEN register. Writing 0 has no effect 0_B No action 1_B Set
SRIES	22	w	**SRI/SPB Bus Access Error Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register, and generates an EXI interrupt if enabled in the corresponding bit of the MFLAGSEN register. Writing 0 has no effect 0_B No action 1_B Set
PIE1S	23	w	**PIE1 Error Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register, and generates an EXI interrupt if enabled in the corresponding bit of the MFLAGSEN register. Writing 0 has no effect 0_B No action 1_B Set

Field	Bits	Type	Description
PIE2S	24	w	**PIE2 Error Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register, and generates an EXI interrupt if enabled in the corresponding bit of the MFLAGSEN register. Writing 0 has no effect. 0_B No action 1_B Set
CRCES	25	w	**CRC Error Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register, and generates an EXI interrupt if enabled in the corresponding bit of the MFLAGSEN register. Writing 0 has no effect. 0_B No action 1_B Set
TSES	28	w	**Target Stream Enable Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Set
TEOS	30	w	**Transmit Enable Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Set
INIS	31	w	**Initialize Mode Flag Set** Writing 1 sets the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Set
0	17:16, 27:26, 29	r	**Reserved**

Miscellaneous Flags Clear Register

Note: Read access to this register returns all zero.

MFLAGSCL
Miscellaneous Flags Clear Register **(020_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INIC	**TEOC**	**0**	**TSEC**	**0**		**CRCEC**	**PIE2C**	**PIE1C**	**SRIEC**	**MAVC**	**ISBC**	**IMBC**	**TMBC**	**0**	
w	w	r	w	r		w	w	w	w	w	w	w	w	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNEXPECTEDC				**TIMEOUTC**				**TTEC**				**NACKC**			
w				w				w				w			

Field	Bits	Type	Description
NACKC	3:0	w	**NACK Flags Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_H no action 1_H clear channel 0 2_H clear channel 1 4_H clear channel 2 8_H clear channel 3 **others**, clear the corresponding combination of channels
TTEC	7:4	w	**Transaction Tag Error Flags Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_H no action 1_H clear channel 0 2_H clear channel 1 4_H clear channel 2 8_H clear channel 3 **others**, clear the corresponding combination of channels
TIMEOUTC	11:8	w	**Timeout Error Flags Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_H no action 1_H clear channel 0 2_H clear channel 1 4_H clear channel 2 8_H clear channel 3 **others**, clear the corresponding combination of channels
UNEXPECTEDC	15:12	w	**Unexpected Error Flags Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_H no action 1_H clear channel 0 2_H clear channel 1 4_H clear channel 2 8_H clear channel 3 **others**, clear the corresponding combination of channels
TMBC	18	w	**Target Memory Block Flag Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Clear
IMBC	19	w	**Initiator Memory Block Flag Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Clear

Field	Bits	Type	Description
ISBC	20	w	**Initiator Stream Block Request Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Clear
MAVC	21	w	**MAV Flag Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect 0_B No action 1_B Clear
SRIEC	22	w	**SRI/SPB Bus Access Error Flag Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect 0_B No action 1_B Clear
PIE1C	23	w	**PIE1 Error Flag Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect 0_B No action 1_B Clear
PIE2C	24	w	**PIE2 Error Flag Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Clear
CRCEC	25	w	**CRC Error Flag Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Clear
TSEC	28	w	**Target Stream Enable Flag Clear** Writing 1 sets the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Clear
TEOC	30	w	**Transmit Enable Flag Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Clear
INIC	31	w	**Initialize Mode Flag Clear** Writing 1 clears the corresponding bit in the MFLAGS register. Writing 0 has no effect. 0_B No action 1_B Clear

Field	Bits	Type	Description
0	17:16, 27:26, 29	r	**Reserved**

Flags Enable Register

MFLAGSEN
Flags Enable Register **(024$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**TEIEN**		**0**		**CRCEEN**	**PIE2EN**	**PIE1EN**	**SRIEEN**	**MAVEN**	**0**				
r		rw		r		rw	rw	rw	rw	rw	r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UNEXPECTEDEN				**TIMEOUTEN**				**TTEEN**				**NACKEN**			
rw				rw				rw				rw			

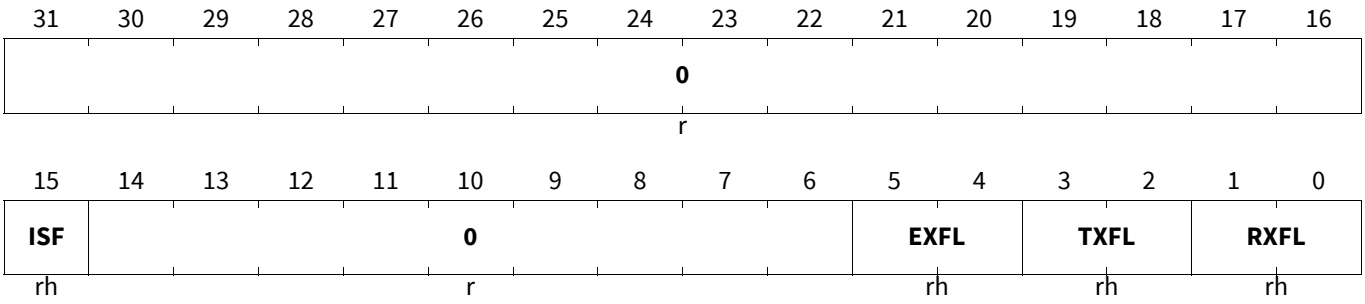
Field	Bits	Type	Description
NACKEN	3:0	rw	**Not Acknowledge Error Enable Bits** Used to enable the error interrupt associated to the corresponding bit in the MFLAGS register. 0_H all disabled 1_H enable channel 0 2_H enable channel 1 4_H enable channel 2 8_H enable channel 3 **others**, enable the corresponding combination of channels
TTEEN	7:4	rw	**Transaction Tag Error Enable Bits** Used to enable the error interrupt associated to the corresponding bit in the MFLAGS register. 0_H all disabled 1_H enable channel 0 2_H enable channel 1 4_H enable channel 2 8_H enable channel 3 **others**, enable the corresponding combination of channels
TIMEOUTEN	11:8	rw	**Timeout Error Enable Bits** Used to enable the error interrupt associated to the corresponding bit in the MFLAGS register. 0_H all disabled 1_H enable channel 0 2_H enable channel 1 4_H enable channel 2 8_H enable channel 3 **others**, enable the corresponding combination of channels

Field	Bits	Type	Description
UNEXPECTED EN	15:12	rw	**Unexpected Error Enable Bits** Used to enable the error interrupt associated to the corresponding bit in the MFLAGS register. 0_H all disabled 1_H enable channel 0 2_H enable channel 1 4_H enable channel 2 8_H enable channel 3 **others**, enable the corresponding combination of channels
MAVEN	21	rw	**MAV Enable Bit** Used to enable the interrupt associated to the corresponding bit in the MFLAGS register. 0_B Disabled 1_B Enabled
SRIEEN	22	rw	**SRI/SPB Bus Access Error Enable Bit** Used to enable the error interrupt associated to the corresponding bit in the MFLAGS register. 0_B Disabled 1_B Enabled
PIE1EN	23	rw	**PIE1 Error Enable Bit** Used to enable the error interrupt associated to the corresponding bit in the MFLAGS register. 0_B Disabled 1_B Enabled
PIE2EN	24	rw	**PIE2 Error Enable Bit** Used to enable the error interrupt associated to the corresponding bit in the MFLAGS register. 0_B Disabled 1_B Enabled
CRCEEN	25	rw	**CRC Error Enable Bit** Used to enable the error interrupt associated to the corresponding bit in the MFLAGS register. 0_B Disabled 1_B Enabled
TEIEN	29	rw	**TEI Enable Bit** Used to enable the interrupt associated to the corresponding bit in the MFLAGS register. 0_B Disabled 1_B Enabled
0	20:16, 28:26, 31:30	r	**Reserved**

Stream FIFOs Status Flags Register

SFSFLAGS

Stream FIFOs Status Flags Register (028$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RXFL	1:0	rh	**Stream RxFIFO Filling Level** Indicates the filling level of the FIFO with granularity of 32 bytes (one stream frame payload size). 00_B 0 01_B 1 10_B 2 11_B Reserved (not possible)
TXFL	3:2	rh	**Stream TxFIFO Filling Level** Indicates the filling level of the FIFO with granularity of 32 bytes (one stream frame payload size). 00_B 0 01_B 1 10_B 2 11_B Reserved (not possible)
EXFL	5:4	rh	**Stream Expect FIFO Filling Level** Indicates the filling level of the FIFO with granularity of 32 bytes (one stream frame payload size). 00_B 0 01_B 1 10_B 2 11_B Reserved (not possible)
ISF	15	rh	**Initiator Stream Frame Request** Indicates if stream TXFIFO request is pending. Set and cleared by the TXFIFO. 0_B TXFIFO fill request not pending 1_B TXFIFO fill request pending
0	14:6, 31:16	r	**Reserved**

35.4.3 Channel.Initiator Registers

This section describes the channel specific registers of the HSSL module, associated to both the initiator and the target functions. In order to support command queues, the addresses of the **IWDx (x=0-3)**, **ICONx (x=0-3)**, **IRWAx (x=0-3)**, and **IRDx (x=0-3)** registers are ordered in a sequence of IWD0, ICON0, IRWA0, IRD0, IWD1, ICON1, IRWA1, IRD1, IWD2, ICON2, IRWA2, IRD2, IWD3, ICON3, IRWA3, IRD3, see **DMA Operated Command Queues**. For the timeout handling description, see **Command Timeout Operation**.

Initiator Write Data Register x

The IWD register contains the data part of a write command.

IWDx (x=0-3)
Initiator Write Data Register x **(030_H+x*10_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															
rw															

Field	Bits	Type	Description
DATA	31:0	rw	**Data Part of the Payload of a Write Frame** For 8-bit and16-bit write command frames, the whole frame payload width of 32-bit is automatically filled with copies of the lower 8-bits or 16-bits of the register.

Initiator Control Data Register x

The ICON register contains the control and the configuration bits of a channel.

Bitfields TOREL and ITTAG are used for command frames, and also reused for streaming frames in channel 2. All other bit fields are used only for command frames.

ICONx (x=0-3)
Initiator Control Data Register x **(034_H+x*10_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOREL								ITTAG			BSY	RWT		DATLEN	
rw								rh			rh	rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOCV								CETT			LETT			TQ	IDQ
rh								rh			rh			w	w

Field	Bits	Type	Description
IDQ	0	w	**Read ID Request** This bit provides the only way to request a read ID frame. Reads always 0. Write of 1 commences a request. In case of parallel write of 1 to TQ and IDQ, the IDQ request has higher priority. 0_B No read ID request 1_B Read ID frame request
TQ	1	w	**Trigger Request** This bit provides an alternative way to request a trigger frame, without having to write to the channel address register. Reads always 0. Write of 1 commences a request. In case of parallel write of 1 to TQ and IDQ, the IDQ request has higher priority. 0_B No trigger request 1_B Trigger frame request
LETT	4:2	rh	**Last Error Transaction Tag**
CETT	7:5	rh	**Currently Expected Transaction Tag**
TOCV	15:8	rh	**Time Out Current Value**
DATLEN	17:16	rw	**Data Length** Defines the length of the data in bits of the write and read command. 00_B 8-bit 01_B 16-bit 10_B 32-bit 11_B Reserved (implemented as 32-bit)
RWT	19:18	rw	**Read Write Trigger Command Type** Defines if the write to the IRWA register will trigger read, write or trigger request. 00_B No action 01_B Read Frame 10_B Write Frame 11_B Trigger Frame
BSY	20	rh	**Channel Busy**
ITTAG	23:21	rh	**Initiator Transaction Tag** This bit displays the current value of the three bit counter generating the new transaction tag value.
TOREL	31:24	rw	**Time Out Reload Value** Defines the duration of the timeout in units of prescaled clock periods. This parameter is valid both in command and stream mode of the channel 2.

Initiator Read Write Address Register x

The IRWA register contains the address part of a write command. Writing the IRWA register triggers a transmit request for the appropriate channel.

IRWAx (x=0-3)

Initiator Read Write Address Register x **(038_H+x*10_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS															
rw															

Field	Bits	Type	Description
ADDRESS	31:0	rw	**Address Part of the Payload of a Write Frame** Writing to this registers triggers transmission of a Write Frame. The address must be aligned according to the data width: byte addresses for 8-bit data, half-word addresses for 16-bit data, word addresses for 32-bit data.

Initiator Read Data Register x

The IRD register contains the data read from the target, which has been fed back as a response to a read command.

IRDx (x=0-3)

Initiator Read Data Register x **($03C_H$+x*10_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA															
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA															
rh															

Field	Bits	Type	Description
DATA	31:0	rh	**Data Delivered by a Read Response Frame**

35.4.4 Channel.Target Registers

This section describes the channel specific registers of the HSSL module, associated to both the initiator and the target functions.

Target Current Data Register i

The TCD register contains the data part of a write command which is currently being processed by the channel on the target side, or the latest data which has been read by the SRI/SPB master in case of a read command.

TCDi (i=0-3)

Target Current Data Register i **(070_H+i*8)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D															
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D															
rh															

Field	Bits	Type	Description
D	31:0	rh	**Data Part of the Payload of a Write Command Frame or Read Data of a Read Command Frame**

Target Current Address Register i

The TCA register contains the address part of a read or write command which is currently being processed by the channel on the target side.

In case of Read ID frame, the address is copied from the register TIDADD.

TCAi (i=0-3)

Target Current Address Register i **(074_H+i*8)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
A															
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A															
rh															

Field	Bits	Type	Description
A	31:0	rh	**Address Part of the Payload of a Write Command Frame or a Read Command Frame or ID Frame**

Target Status Register

The TSTAT register contains the status bits of the common target functionality.

TSTAT

Target Status Register **(090_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LASTTT3			LASTCC3					LASTTT2			LASTCC2				
rh			rh					rh			rh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LASTTT1			LASTCC1					LASTTT0			LASTCC0				
rh			rh					rh			rh				

Field	Bits	Type	Description
LASTCCx (x=0-3)	8*x+4:8*x	rh	**Last Command Code** Indicates the latest command code from the header for the corresponding channel.
LASTTTx (x=0-3)	8*x+7:8*x+5	rh	**Last Transaction Tag** Indicates the transaction tag of the latest command or stream frame for the corresponding channel.

Target ID Address Register

The TIDADD register contains the address from which the target ID is fetched. After the Power-On Reset, System Reset, and Application Reset the start-up software writes this register with the address of CBS_JTAGID register. After Kernel Reset, the start-up software is not executed, and the reset value is not changed.

TIDADD

Target ID Address Register **(094_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
A															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A															
rw															

Field	Bits	Type	Description
A	31:0	rw	**Address Pointer** Address pointer containing the address of the memory location containing the unique ID data.

35.4.5 Initiator Stream Registers

(>> **Memory Block Transfer Modes of the Stream Channel**)

Initiator Stream Start Address Register

ISSAx (x=0-1)
Initiator Stream Start Address Register **($0A0_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
START															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
START											0				
rw											r				

Field	Bits	Type	Description
START	31:5	rw	**Start Address for the Memory Range** Aligned on 256-bit limit (stream frame payload size).
0	4:0	r	**Reserved**

Initiator Stream Current Address Register

ISCA
Initiator Stream Current Address Register **($0A8_H$)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURR															
rh															

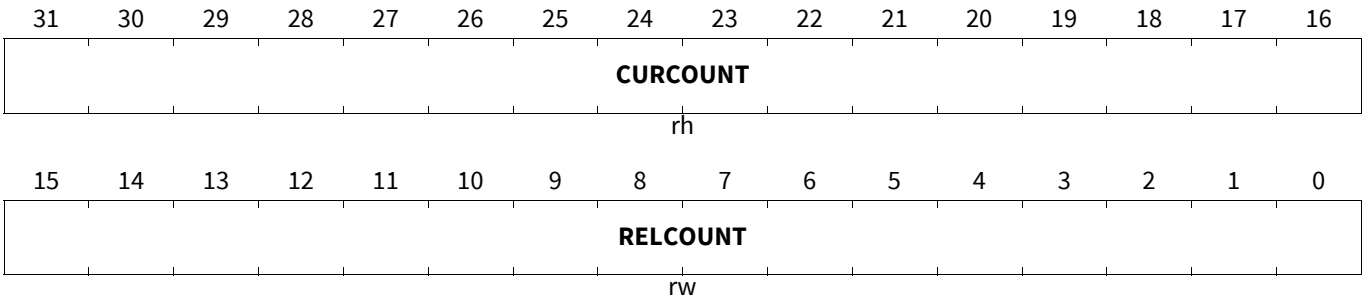
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURR											0				
rh											r				

Field	Bits	Type	Description
CURR	31:5	rh	**Address of the Memory Location for the Current Transfer** Aligned on 256-bit limit (stream frame payload size).
0	4:0	r	**Reserved**

Initiator Stream Frame Count Register

ISFC

Initiator Stream Frame Count Register (0AC$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RELCOUNT	15:0	rw	**Reload Count Number** Contains the number of frames to transfer per memory block. Bit field length depends on application requirements. 0000$_H$1 0001$_H$1 0002$_H$2 0003$_H$3 **others**, corresponding number of frames
CURCOUNT	31:16	rh	**Current Count Number** Displays the current count number, which is generated by down-counting from the RELCOUNT value. Bit field length depends on application requirements.

35.4.6 Target Stream Registers

(>> **Memory Block Transfer Modes of the Stream Channel**)

Target Stream Start Address Register x

TSSAx (x=0-1)

Target Stream Start Address Register x **($0B0_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR											0				
rw											r				

Field	Bits	Type	Description
ADDR	31:5	rw	**Start Address for the Memory Range** Aligned on 256-bit (or 32 byte) limit (stream frame payload size).
0	4:0	r	**Reserved**

Target Stream Current Address Register

TSCA

Target Stream Current Address Register **($0B8_H$)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURR															
rh															

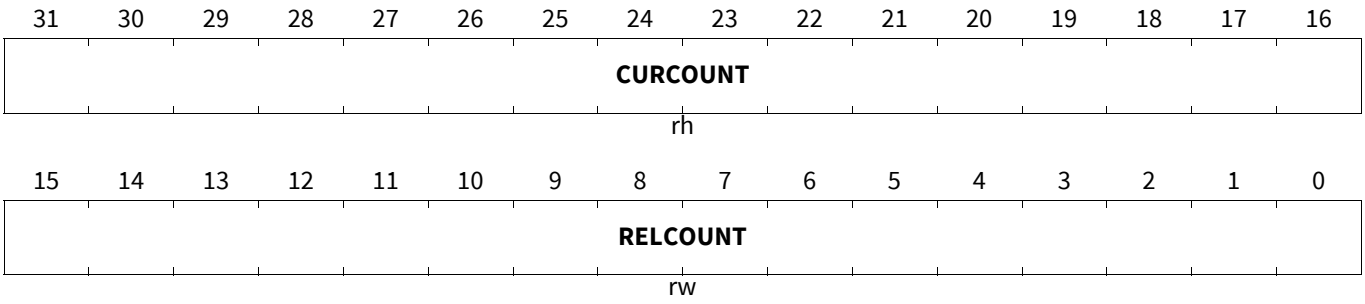
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURR											0				
rh											r				

Field	Bits	Type	Description
CURR	31:5	rh	**Address of the Memory Location for the Current Transfer** Aligned on 256-bit (or 32 byte) limit (stream frame payload size).
0	4:0	r	**Reserved**

Target Stream Frame Count Register

TSFC

Target Stream Frame Count Register (0BC$_H$) Application Reset Value: 0000 0000$_H$



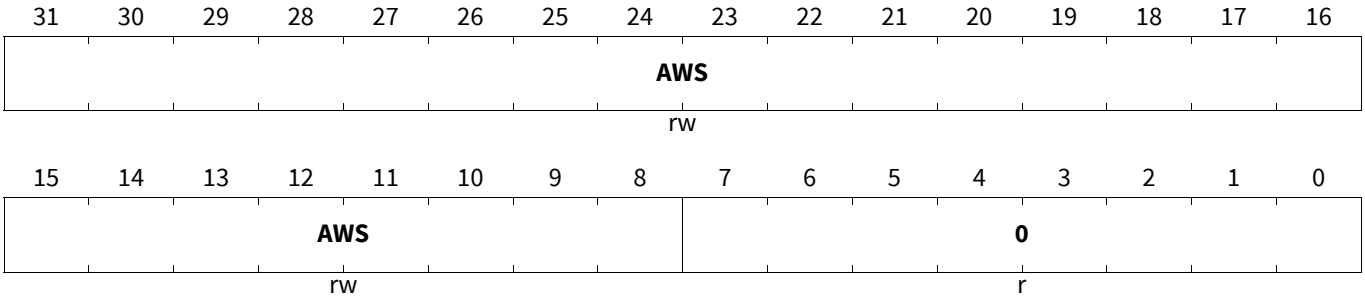
Field	Bits	Type	Description
RELCOUNT	15:0	rw	**Reload Count Number** Contains the number of frames to transfer per memory block. Bit field length depends on application requirements. 0000_H1 0001_H1 0002_H2 0003_H3 **others**, corresponding number of frames
CURCOUNT	31:16	rh	**Current Count Number** Displays the current count number, which is generated by down-counting from the RELCOUNT value. Bit field length depends on application requirements.

35.4.7 Access Protection Registers

This sub section describes the registers defining the access windows where the HSSL accesses are allowed. These registers are writable only in supervisor mode, which makes them not writable by the HSSL itself.

Access Window Start Register i

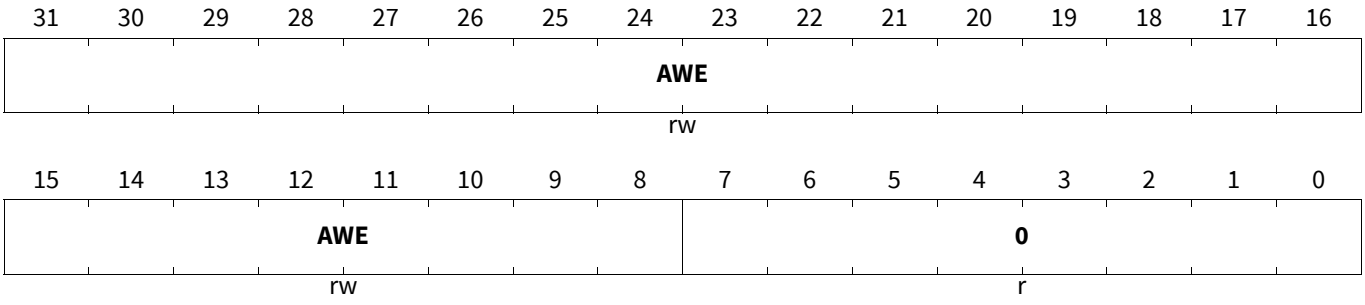
AWSTARTi (i=0-3)
Access Window Start Register i **($0C0_H$+i*8)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
AWS	31:8	rw	**Access Window Start Address** This bit field defines the upper 24 bits of the start address of the corresponding access window. This results in a granularity of 256 bytes for the start address.
0	7:0	r	**Reserved**

Access Window End Register i

AWENDi (i=0-3)
Access Window End Register i **($0C4_H$+i*8)** **Application Reset Value: 0000 0000_H**

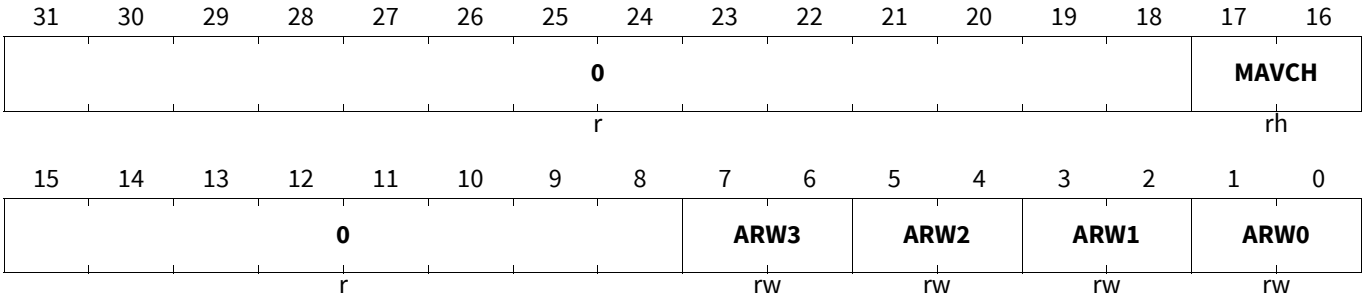


Field	Bits	Type	Description
AWE	31:8	rw	**Access Window End Address** This bit field defines the upper 24 bits of the end address of the corresponding access window. This results in a granularity of 256 bytes for the end address.
0	7:0	r	**Reserved**

Access Rules Register

AR

Access Rules Register **($0E0_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
ARWx (x=0-3)	2*x+1:2*x	rw	**Access Rule for Window x** 00_B No access (window disabled) 01_B Read access allowed 10_B Write access allowed 11_B Read and write access allowed
MAVCH	17:16	rh	**Memory Access Violation Channel** This bit field shows the number of the latest channel that attempted a not allowed access. 00_B Channel 0 01_B Channel 1 10_B Channel 2 11_B Channel 3
0	15:8, 31:18	r	**Reserved**

35.4.8 Security and Multi Slave Registers

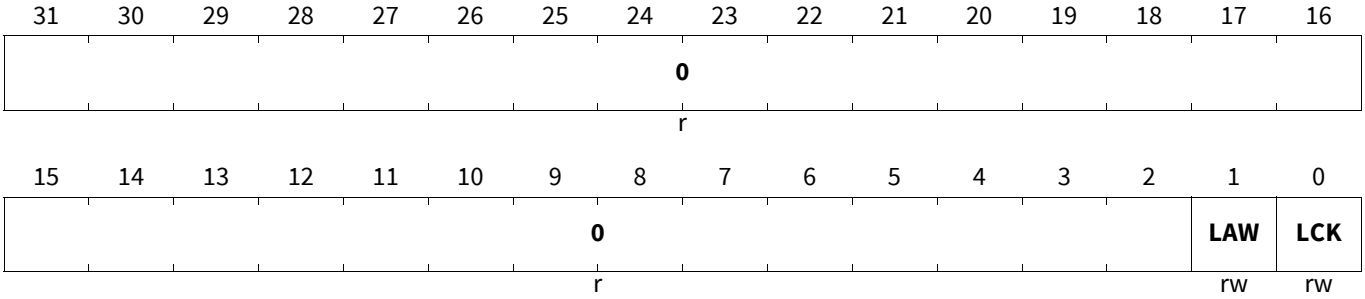
This sub section describes two registers: one handling the security aspects regarding the HSSL module operation and one configuring the multi slave mode of operation.

Security Control Register

This register can only be written by an access from the HSM master (TAG = 000011B). A write operation performed by any other master is ignored and the bit remains unchanged.

SEC

Security Control Register **(098_H)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
LCK	0	rw	**Lock the HSSL Module** Setting this bit field prevents the MFLAGS.INI bit to be cleared, that is prevents to leave the INIT state and go to RUN mode. If INIT bit has already been cleared, it is possible to set it, but not to clear it afterwards. This bit can only be written by an access from the HSM master (TAG = 000011B). A write operation performed by any other master is ignored and the bit remains unchanged. 0_B Unlocked 1_B Locked
LAW	1	rw	**Lock the Address Windows Registers** This bit field shows if the Address Window Registers AWSTARTx, AWENDx, AR and TIDADD are locked or not. If locked, the properties of the address windows cannot be changed any more. This bit can only be written by an access from the HSM master (TAG = 000011B). A write operation performed by any other master is ignored and the bit remains unchanged. 0_B Unlocked 1_B Locked
0	31:2	r	**Reserved**

Multi Slave Control Register

MSCR
Multi Slave Control Register **($09C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															ITXSTOP
r															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													SLAVETAG		EN
r													rw		rw

Field	Bits	Type	Description
EN	0	rw	**Multi Slave Mode Enable** This bit enables the multi slave mode of operation of the HSSL module. 0_B Disabled 1_B Enabled
SLAVETAG	2:1	rw	**Slave Tag** The master uses this bit field to define the current slave it sends to and receives from. The slave uses this bit field to pass through only the relevant incoming frames and to tag its own transmit frames. This tag is injected in the header by the Slave Tag Translator block and used for filtering in Slave Tag Filter Block. 00_B Reserved 01_B Slave 1 10_B Slave 2 11_B Slave 3
ITXSTOP	16	rw	**Initiator Transmission Stop** This bit is intended to be set or cleared by the master in a multi-slave operation by using 16-bit write command frames. MSCR.ITXSTOP functionality can be activated only if MSCR.EN = 1, otherwise it is ignored. Setting this bit stops the arbitration and transmission of new command and stream frames of the initiator but does not stop the ongoing frames. 0_B Initiator transmission active 1_B Initiator transmission stopped
0	15:3, 31:17	r	**Reserved**

35.4.9 BPI_FPI Module Registers

Figure 396 shows all registers associated with the BPI_FPI module, configured for one kernel. The Offset Address in the following BPI_FPI register table and in the BPI_FPI register descriptions are proposals. In a standard 32 bit peripheral the CLC is mapped to offset address 00h, module ID to 08h. The proposal for the new BPI_SPB register is to map them to the end address of the peripheral address range together with the peripherals SRNs.

BPI_FPI Registers Overview

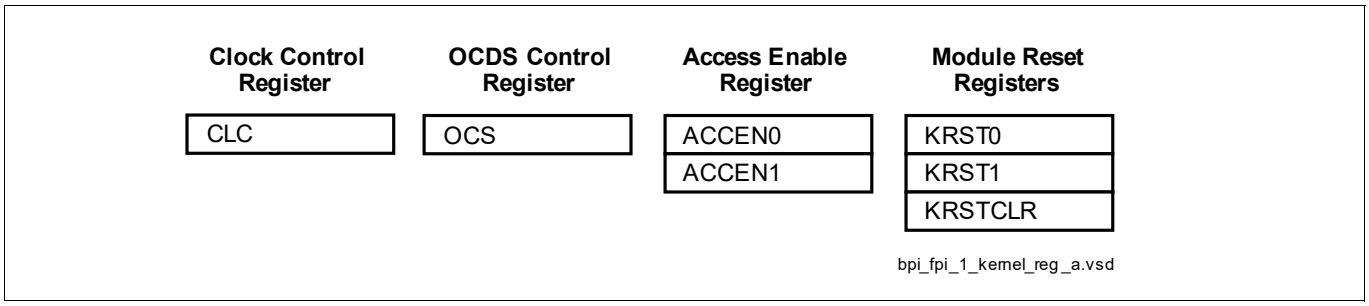


Figure 396 BPI_FPI Registers

The writes of the bus masters to the HSSL module is controlled by Access Protection registers ACCENx.

The HSSL implements two ACCENx registers, ACCEN0 and ACCEN1.

The ACCENx registers are protected by Safety Endinit mechanism.

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the $f_{<>}$ module clock signal, sleep mode and disable mode for the module.

CLC

Clock Control Register **(000_H)** **Application Reset Value: 0000 0003$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode.

Field	Bits	Type	Description
0	2, 31:4	r	**Reserved** Read as 0; should be written with 0.

OCDS Control and Status

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32 bit wide only and requires Supervisor Mode

Note: The bit SUSSTA is set only at entering the suspend state from the run state. It will not be set if the soft suspend request comes in any other state than run state, like init or disabled/sleep state.

OCS
OCDS Control and Status **($0E8_H$)** **Debug Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**SUSSTA**	**SUS_P**	**SUS**				**0**							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												**TG_P**	**TGB**	**TGS**	
r												w	rw	rw	

Field	Bits	Type	Description
TGS	1:0	rw	**Trigger Set for OTGB0/1** 00_B No Trigger Set output 01_B TS16_STR, Streaming Channel Trigger Set 10_B TS16_ERR, Errors Trigger Set 11_B Reserved
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1
TG_P	3	w	**TGS, TGB Write Protection** TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. 2_H Soft suspend **others**, reserved,
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.

Field	Bits	Type	Description
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:4, 31:30	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding. The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B,…, EN31 -> TAG ID 011111B.

ACCEN0
Access Enable Register 0 **($0FC_H$)** **Application Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENy (y=0-31)	y	rw	**Access Enable for Master TAG ID y** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID y 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

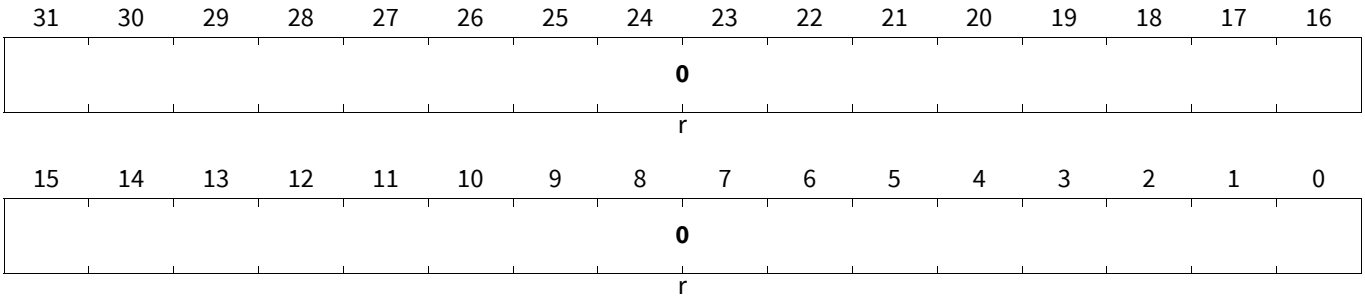
The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B,…, EN31 -> TAG ID 111111B.

1)

ACCEN1

Access Enable Register 1 **(0F8$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0

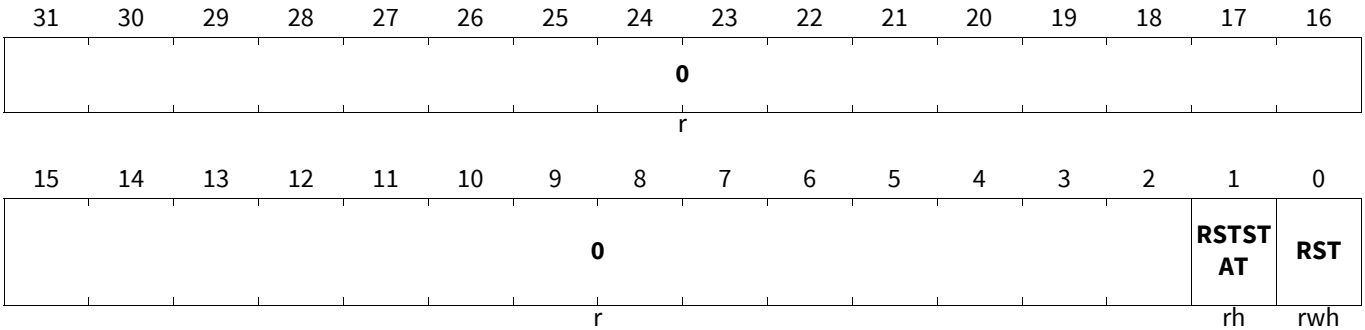
The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0

Kernel Reset Register 0 **(0F4$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested

Field	Bits	Type	Description
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($0F0_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

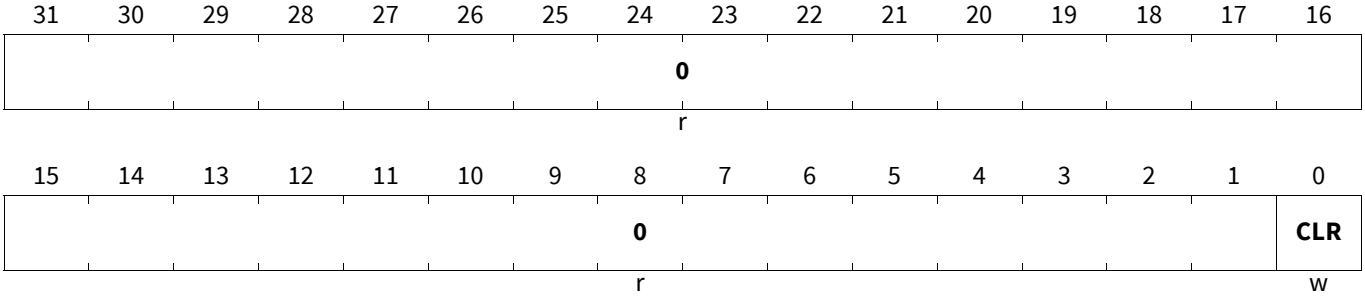
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**RST**
r															rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR
Kernel Reset Status Clear Register (0EC$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

35.5 IO Interfaces

Table 311 List of HSSL Interface Signals

Interface Signals	I/O	Description
COK_INT(3:0)	out	**Channel OK Service Request**
RDI_INT(3:0)	out	**Channel Read Data Service Request**
ERR_INT(3:0)	out	**Channel Error Service Request**
TRG_INT(3:0)	out	**Channel Trigger Interrupt Service Request m**
EXI_INT	out	**HSSL Exception Service Request**
sri_rph_err	out	**SRI read phase error**

35.6 Revision History

Table 312 Revision History

Reference	Change to Previous Version	Comment
V3.0.16		
Page 82	Previous versions removed from revision history.	
V3.0.17		
Page 3	Paragraph splitted, 2 questionmarks (“?”) removed.	
Page 82	Updated Interface Signal table.	
V3.0.18		
-	No functional changes.	

Table 312 Revision History (cont'd)

Reference	Change to Previous Version	Comment
V3.0.19		
Page 35	Typo fixed: "TSFC.TSE only when MFLAGS.CURCOUNT" replaced by "MFLAGS.TSE only when TSFC.CURCOUNT".	

35.7 High Speed Communication Tunnel (HSCT)

This chapter describes the High Speed Communication Tunnel interface for the Microcontroller high speed data communication including the physical layer and the control (link) layer.

35.7.1 Feature List

The HSCT interface features can be separated in common features, features supported by single Slave connection and features supported by multi-slave (max 3 slaves) connection to the microcontroller. The multi slave connection has restrictions, which are also listed in the multi-slave feature list.

HSCT common interface basic features:

- TX-link: Transmit Symbol information from the Master interface to the Slave interface
- RX-link: Receive Sample information from the Slave interface to the Master interface
- System Clock provided by the Master interface (crystal owner) to the Slave interface
- TX-link data transfer speed 5 MBaud and 320 MBaud.
- RX-link data transfer speed based on f_{SYSCLK} = 20 MHz:
  5 MBaud (low speed)
  20 MBaud (medium speed)
  320 MBaud (high speed)
- RX-link data transfer speed based on 10 MHz SysClk:
  5 MBaud (low speed)
  320 MBaud (high speed)
- High speed data transfer rate can be reduced by implemented dividers (2, 4, 8, 16)
- The interface is based on IEEE 1596.3 LVDS IO's and provides Driver swing amplitude configuration

Single Slave connection features

- Control information on TX link and RX link
- Clear To Send indication in both directions
- Unsolicited Status in both directions
- Regular data transfer in both directions based on data channels
- Sleep Mode

Multi Slave (max 3 Slave) connection features and restrictions:

- Control information on TX link only
- Multi-Drop configuration (Master is addressing all slaves connected to the trace line)
- Clear To Send indication not supported
- Unsolicited Status not supported
- Regular data transfer in both directions based on data channel only (derived by Higher layer protocol)
- Multiple sender active/driving the RX link the same time is prohibited (ensured by system Master SW)
- Slave communication channel to a selected slave by higher layer SW; Only one Slave can be addressed and communication must be finished before the next Slave can be addressed
- Slaves driving the LVDS trace line the simultaneously are prohibited
- Connection of 2 Slaves to the LVDS trace line limits the max Baud rate to 160 MBaud
- Connection of 3 Slaves to the LVDS trace line limits the max Baud rate to 80 MBaud

- Discrete termination resistor for differential receiver trace line impedance matching is required on PCB with 100 Ohm +/-5% (check PCB layout guidelines from IFX); internal resistor is not supported

35.7.2 Overview

The HSCT TC3xx employs an interface for inter chip communication between a Master interface and a Slave interface. The interface is based on LVDS IO technology and developed to function as a Master or a Slave interface. During initialization phase the role of the Interface (Master or Slave) must to be defined. It is not intended to change the system role after initialization phase.

The interface is a full-duplex interface for TX and RX data transfer. The data transfer speed can be configured to function in low or high speed and allows to configure in each transfer direction an independent transfer speed in order to reduce power and EMI, based on transfer speed requirements. A sleep mode for the LVDS IO's is implemented, which reduces power on the IO, if the interface is not in use (Note: only available in single slave mode, not available in multi slave mode).

The Master interface is defined to own the crystal and provides the clock (**HSCT SysClk**) to the Slave interface. The HSCT Master/Slave interface allows from system point of view one crystal only. In this scenario the HSCT SysClk input at Slave Side is used as reference clock for other PLL's like SysPLL. The interface reset is derived from System reset and supports common reset behavior, including module reset. The name of the SysClk signal driven by the Master is SYSCLK_OUT. The name of the SysClk signal received by the Slave is SYSCLK_IN.

In **Figure 397** a system level view of the interface and interface connection is shown.

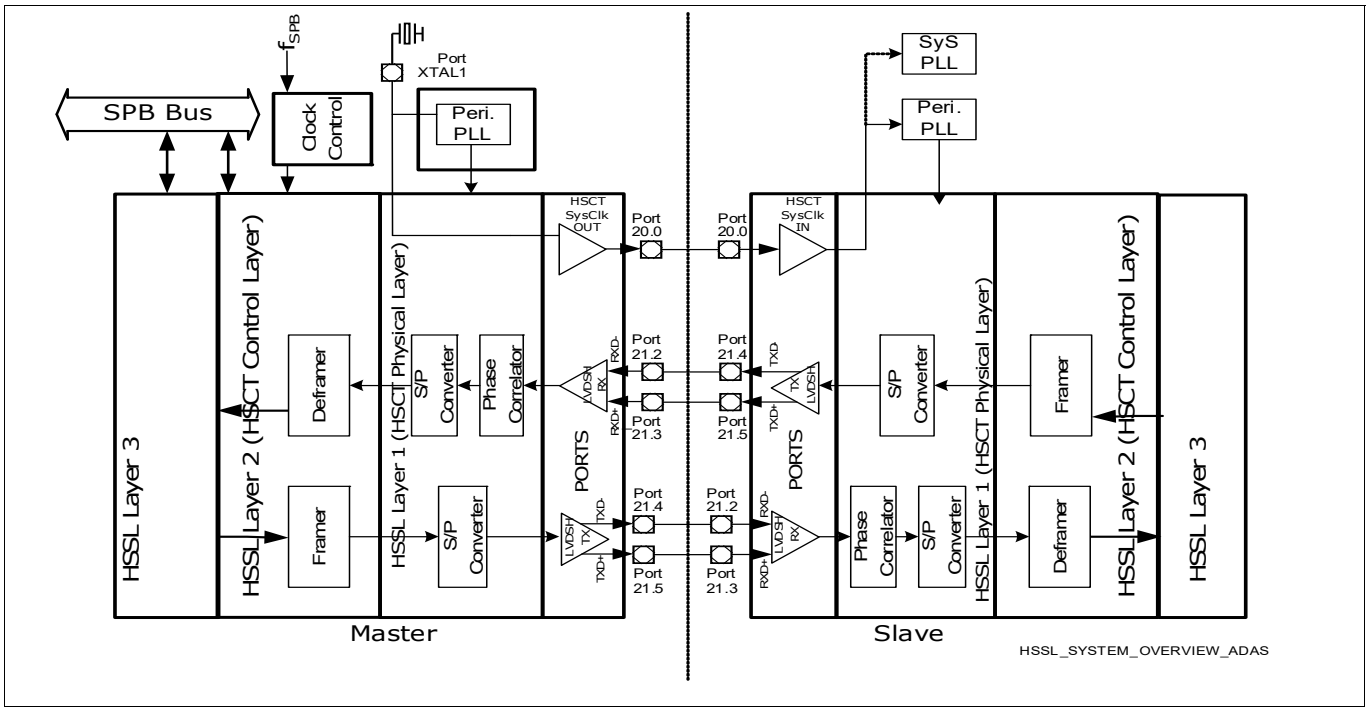


Figure 397 Overview of Interface Signals

35.7.3 Functional Description

35.7.3.1 Introduction

The initial HSCT communication is always established from Master interface. The LVDS IO's are based on standard IO technology following IEEE 1596.3 specification with minor limitations (see Datasheet). The HSCT LVDS IO common voltage is defined to be a 1.2 V. Compared to IEEE 1596.3 the differential voltage level can be reduced via configuration option to reduce EMI.

The intention is to implement the interface either as a Master interface or as a Slave interface. So the interface is able to handle both roles (Master or Slave). The following detailed description describes both roles, which requires the following definitions:

- TX line or TxData link (TX_DAT, TX_DATX) defines the data transfer from the Master interface to the Slave interface.
- RX line or RxData link (RX_DAT, RX_DATX) defines the data transfer from the Slave interface to the Master interface.
- Transmit direction, outbound data (or transmission) describes to send out data independent of the role the interface does have.
- Receive direction, inbound data (or receiving) describes to receive data independent of the role the interface does have.

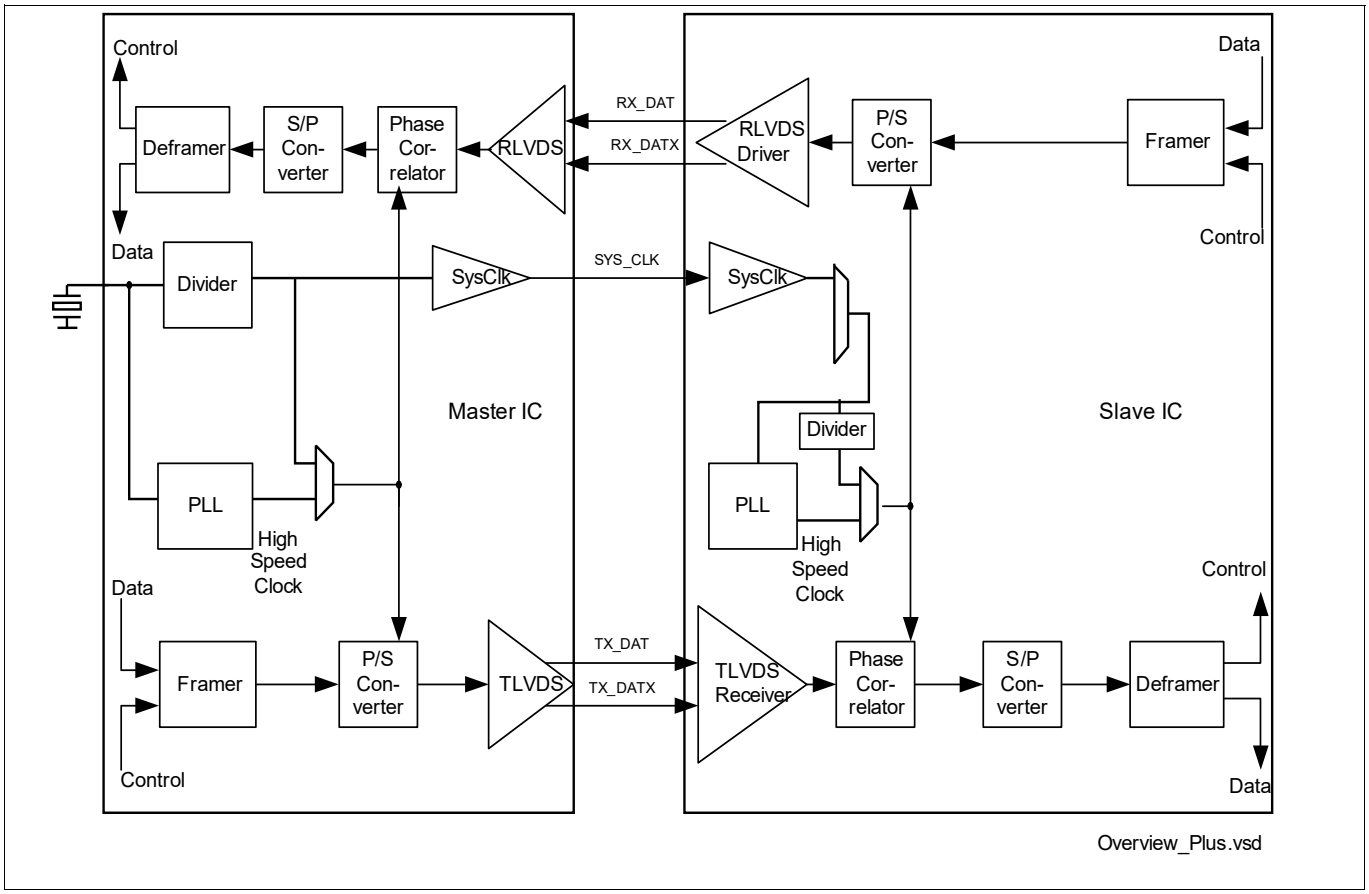


Figure 398 Overview internal block structure

35.7.3.2 Physical Layer

35.7.3.2.1 RX / TX Data communication

During initialization the associated TxData link starts in low-speed mode, with the high-speed clock not enabled or generators turned off and the RxData Link disabled. An interface control sequence, initiated by software, allows to switch the interface in high-speed mode.

The data interface can transmit with a max Baud rate of 320 Mbit/s in the RX and the TX direction (full-duplex). The data is recovered from RX data stream (RX_DAT, RX_DATX) in the Master interface and from TX (TX_DAT,TX_DATX) data stream in the Slave interface. **Table 313** shows the interconnects of the differential data lines for control and data transfer between the Master interface and the Slave interface and shows the limitations for multi Slave connections.

At TX link differential data with the configured baud rate is transmitted from the Master interface to the Slave interface. Proper phase selection is processed at receiver side via five different phases.

At RX link differential data is transferred from the Slave interface to the Master interface. A proper phase selection is processed for data recovery at the Master receiver path via five different phases derived from the clock generator (Peripheral PLL). The Slave interface reference clock must be the HSCTSysClk, driven by the interface Master. The Slave must receive the SysClk from Master interface to be enabled to perform reliable data transfer. A clock at Slave side derived from a crystal will lead to improper data communication, as frequency drift variation is uncontrollable and bigger then deriving it from a single clock source. The Slave interface max frequency (320 MHz data clock) is generated internally from the HSCT SysClk.

The HSCT SysClk at the Slave interface is also the reference clock for other PLLs on the chip.

Note: A local crystal at the Slave interface taken as reference clock for the interface PLL is not allowed as the data recovery based on clock phase shift works on a common reference clock source only!

Table 313 Physical Interconnects

Direction	Name	Description
Transmit (Single Slave)	**TX_DAT**, **TX_DATX**	Channel TX data and control information at a max rate of 320 Mbit/s from the Master interface to the Slave interface
Receive (Single Slave)	**RX_DAT**, **RX_DATX**	Channel RX data and control information at a max rate of 320 Mbit/s from the Slave interface to the Master interface
Transmit (Two Slave)	**TX_DAT**, **TX_DATX**	Channel TX data and control information at a max rate of 160 Mbit/s from the Master interface to the Slave interface
Receive (Two Slave)	**RX_DAT**, **RX_DATX**	Channel RX data and control information at a max rate of 160 Mbit/s from the Slave interface to the Master interface
Transmit (Three Slave)	**TX_DAT**, **TX_DATX**	Channel TX data and control information at a max rate of 80 Mbit/s from the Master interface to the Slave interface
Receive (Three Slave)	**RX_DAT**, **RX_DATX**	Channel RX data and control information at a max rate of 80 Mbit/s from the Slave interface to the Master interface

The RxDAT link (Slave TX) enters shutdown mode by Interface control logical channel command and exit shutdown mode by control command, again. The Slave transmission line is disabled by default and must be enabled by the Master. The Master Interface control commands are used to set the link into operation. TX or RX enable/disable shall not be changed during an active frame transmission or during sleep mode.

Payload content is transferred most significant bit first. For example, if a 32-bit payload contains a message derived from a 32-bit register at TX side, b31 of the register (the MSb) shall be the first bit of payload transmitted and b0 (LSb) shall be the last. Larger payloads should be built up according to the same principle.

35.7.3.2.2 Differential Signaling Principle Based on LVDS

The voltage at the termination resistor of the receiver is either positive or negative, which depends on the direction of the current flowing through it. This is determined by the state of the switches at the transmit stage. The voltage V_{OD} should be evaluated by a comparator with hysteresis.

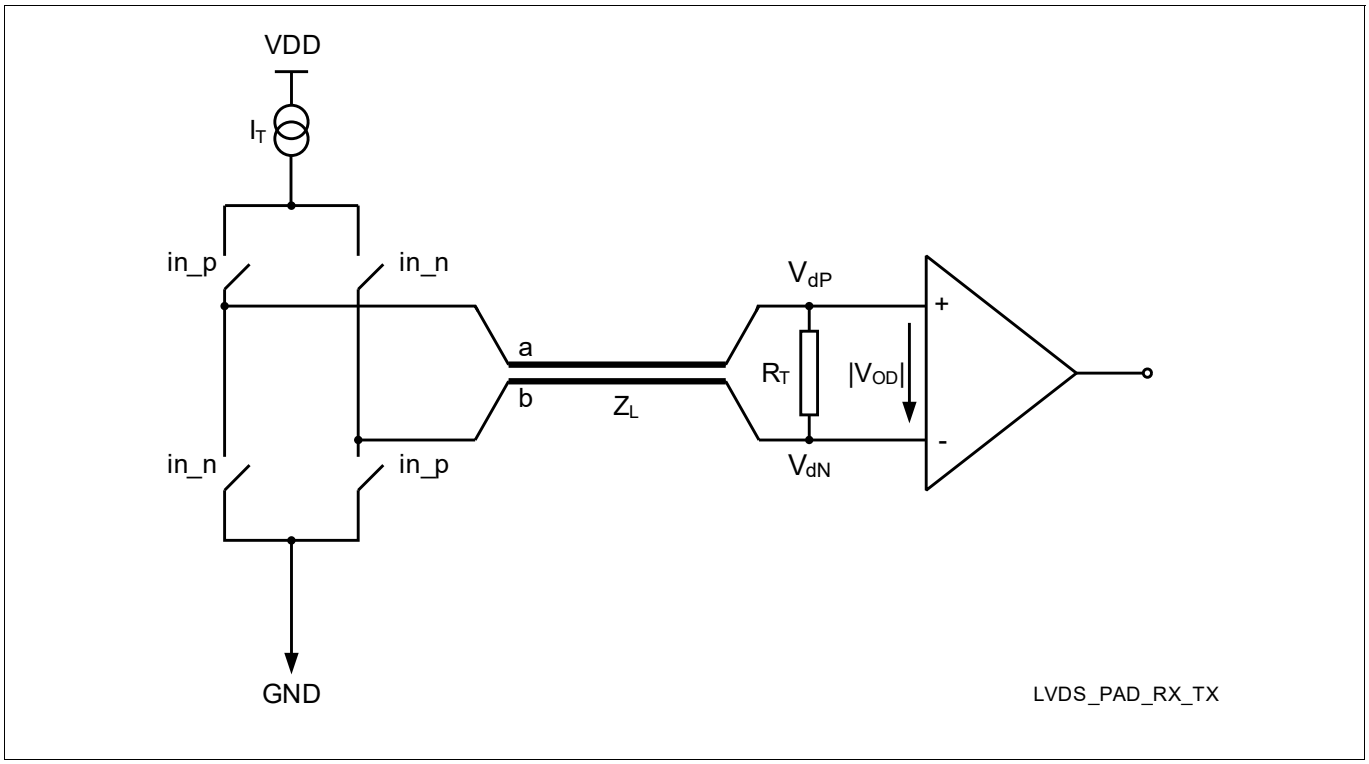


Figure 399 Pad Driver and Pad Receive

The HSCT interface defines no specific test for an error-free transmission, although the signal quality of course suffers from attenuation and noise and in particular it depends on proper wave propagation. Therefore the interconnect lines need to have a characteristic wave impedance Z_L equal to 100 Ω differential, corresponding to the termination resistor R_T at the receiver side.

Note: Due to the nature of wave propagation a termination other than with the characteristic line impedance leads to inevitable signal degradations. Therefore any operation mode other than 100 Ω termination is only supported from a functional point of view (i.e. the corresponding electrical configuration is possible) without any further guarantee of an error-free transmission.

Note: The interface receiver provides an input termination impedance of 100 Ω +/- 20% at the IO. If a more precise termination is required the design offers to disable the internal termination impedance and allow an external termination on the PCB. In multi Slave approach the internal termination impedance must be disabled and a resistor with 100 Ω +/- 5% must be placed on PCB instead (exact placement can be discovered in IFX PCB layout guidelines) as close as possible to the receiver IO.

The differential output voltage V_{OD} is defined as the difference of the voltages V_{dP} and V_{dN} at the receiver input pins.

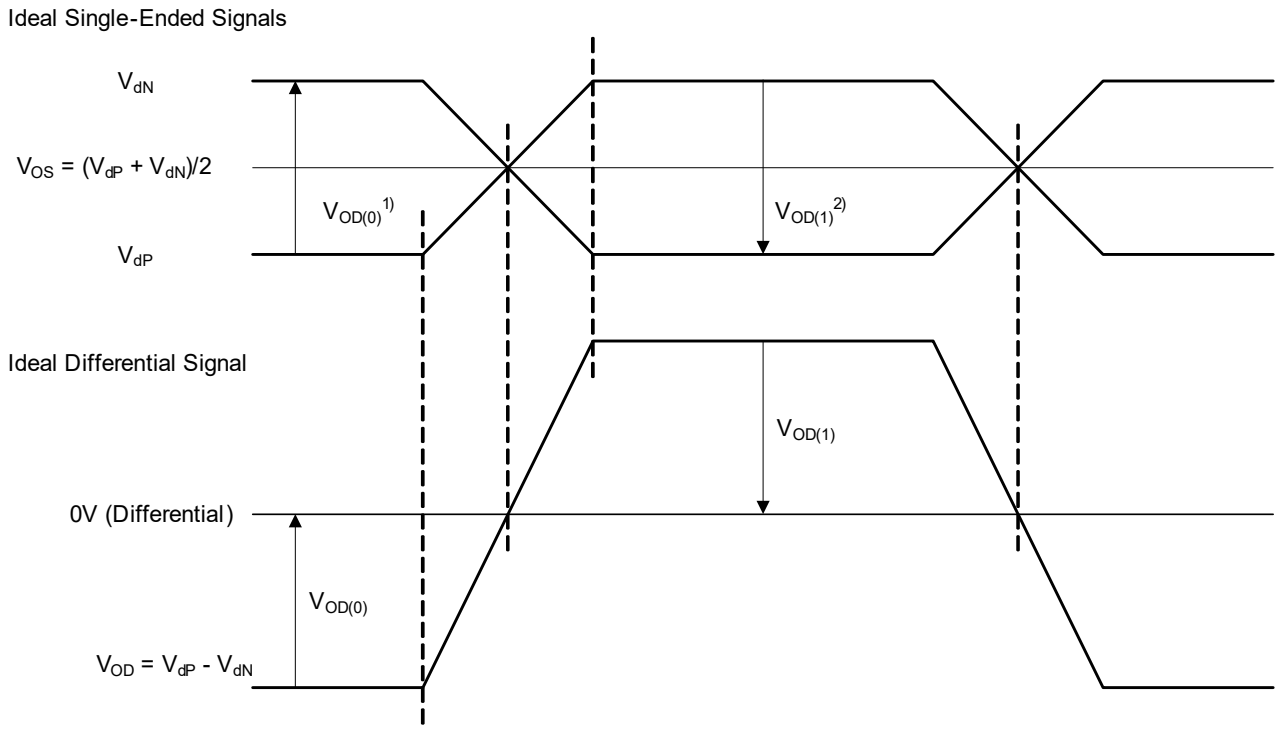
$$V_{OD} = V_{dP} - V_{dN} \quad (35.1)$$

The common-mode voltage V_{OS} is defined as the arithmetic mean value of the voltages at the receiver input pins:

(35.2)

$$V_{OS} = \frac{V_{dP} + V_{dN}}{2}$$

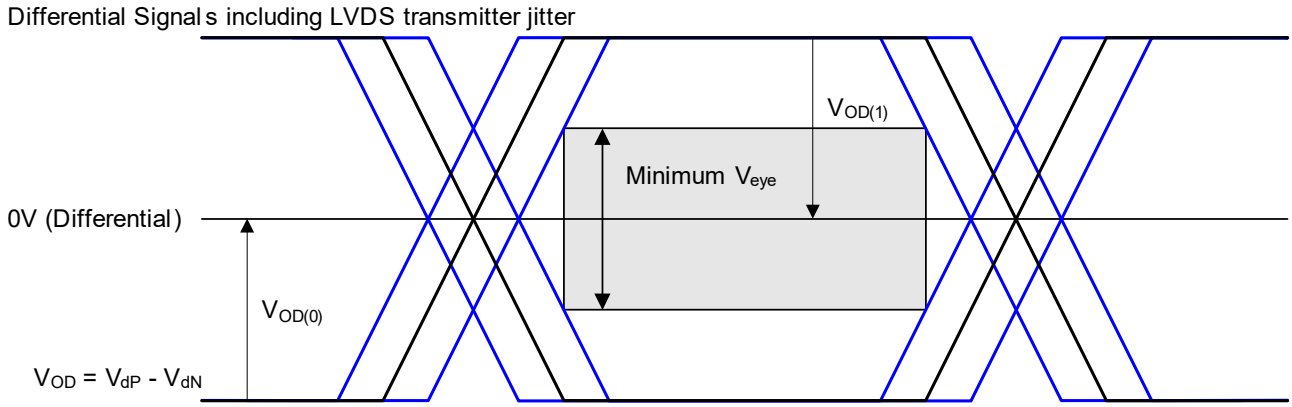
V_{OD}, V_{OS}, and the corresponding minimum differential eye opening V_{eye} considering LVDS transmitter jitter are exemplarily shown in **Figure 400** and **Figure 401** for ideal signals.



1) $V_{OD(0)}$ negative differential voltage representing a logic Zero

2) $V_{OD(1)}$ positive differential voltage representing a logic One

Figure 400 Ideal Single-Ended and Ideal Differential Signals



1) $V_{OD(0)}$ negative differential voltage representing a logic Zero

2) $V_{OD(1)}$ positive differential voltage representing a logic One

Figure 401 Differential Eye Opening

LVDS Pads in Multi Slave Mode

In multi slave mode, the procedure of changing the active slave consists of three steps:

- disabling the currently active slave LVDS transmitter,
- waiting for the LVDS pads to stabilize into Hi-Z state.
- enabling the LVDS transmitter of the next slave,
- waiting for the LVDS bus to stabilize into active state.

Afterwards the first HSSL command can be sent to the newly activated slave.

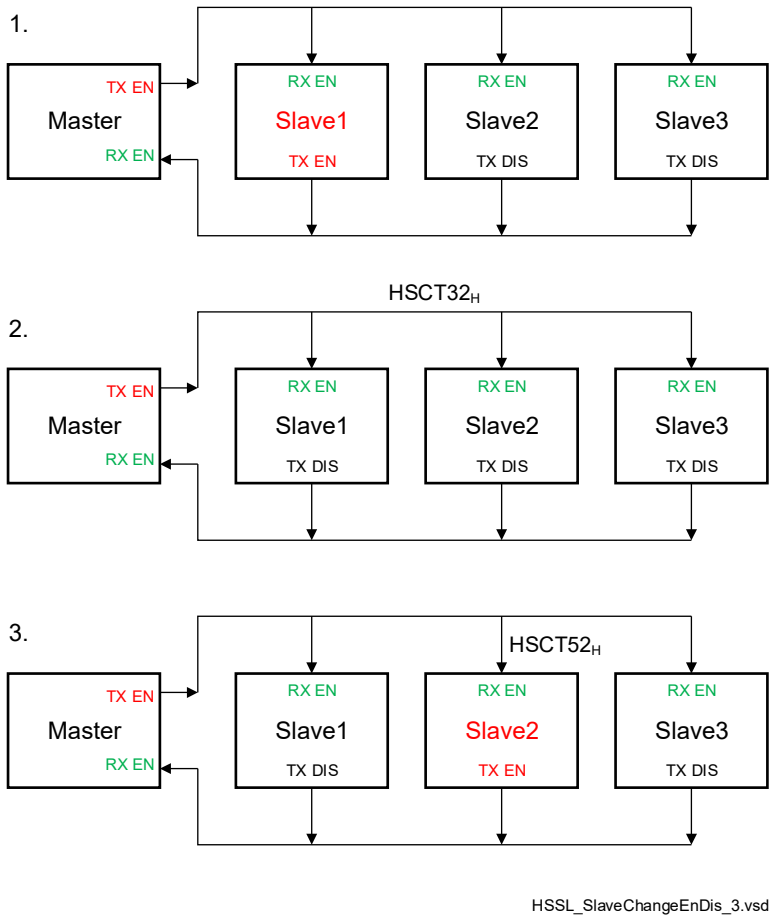


Figure 402 Changing the Active Slave

In the step 2, the output bus of the slaves is not driven by any slave, all LVDS Tx drivers are disabled. The differential voltage is equal to zero, and the offset voltage can take any value between the ground and power supply level.

The enable time and disable time of the LVDS pads shown in **Figure 403** are defined in the data sheet of the corresponding device as parameters t_{LVDSEN} and $t_{LVDSDIS}$.

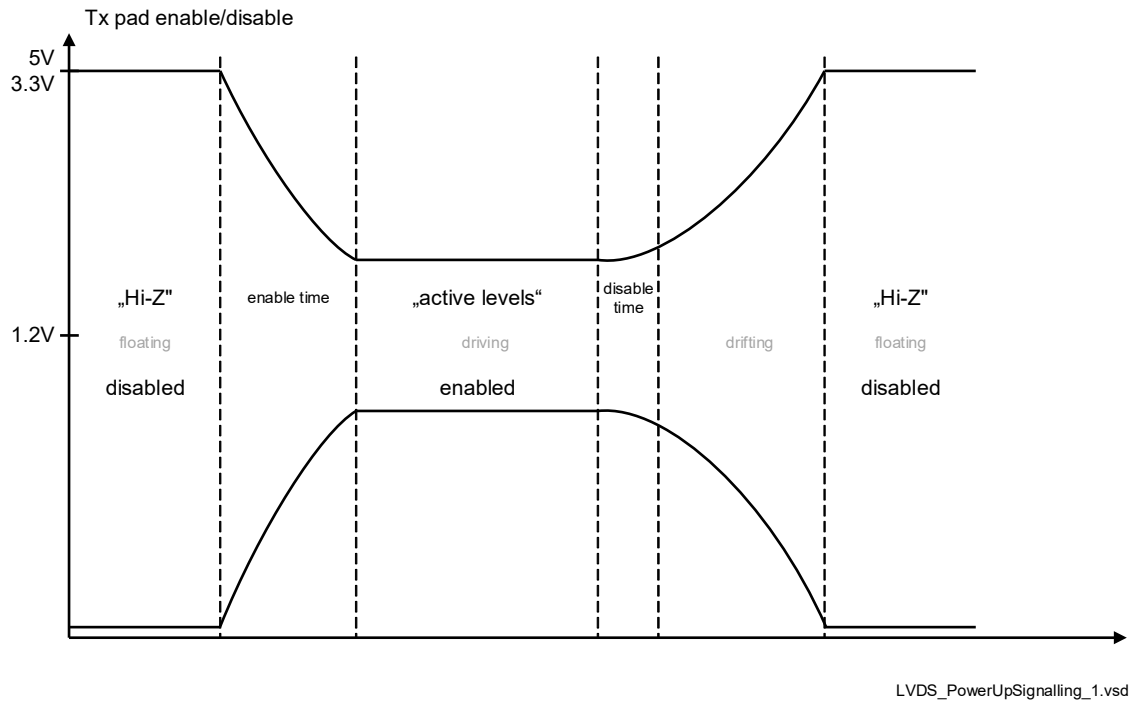


Figure 403 Voltage Levels on the Slave Driven Bus

35.7.3.2.3 Electrical Characteristics Based on LVDS for Reduced Link trace length

The characteristics are based on the LVDS standard IEEE P1596.3-1995 with some limitations, defined in the Datasheet. It is adapted for low power consumption and lower supply voltages for short ranges. It is defined that the interface supply voltages meet the common mode voltage for the differential signaling based on 1.2 V.

Operating Modes

The HSCT interface comprises the following operating modes:

- **Terminated Mode**: The data transmitter requires a 100 Ω external termination between the pins **TX_DAT** and **TX_DATX**. The termination is given by a 100 Ω resistor at the receiver input, of the opposite LVDS device, for matching the differential characteristic impedance of the line with the impedance of the receiver. PCB design has to follow guidelines on differential signal routing. The PCB placed termination resistor can be chosen very precise. Via configuration option an 100 Ω LVDS receiver internal resistor can be enabled with a +/- 20% tolerance and shall not be enabled for a multi Slave scenario (PCB placed and +/- 5% tolerance according to IFX PCB guidelines).
- **Power up**: The line receiver is always be enabled after power up and SysClk is active. The data decoder behind the receiver is not enabled at this point. It is enabled when the first transition on the data lines is detected by the line receiver (8 zero’s at configured Baud rate). The line driver must be enabled on the Master side via SW control. On the Slave side the line driver must be enabled by sending interface message control commands from the Master. After power up the link starts in low speed mode. After line driver enable some time is required to power up the LVDS transmitter line. The data link in RX and TX direction is ready to transmit and receive data after the defined setup time.
- **Sleep mode**: In terminated operation the differential voltage across the resistor will be reduced to nominally zero (-5mV to 20mV at 100 Ω +/- 20%) and hence, the line driver current requirement will reduce to internal bias currents. The common-mode voltage, however, should be maintained. The PLL is not switched off during sleep mode.

- **Power down**: When any interface driver supply voltage is removed, the driver shall provide a weak pull-down to 0 V to ensure that the signals are held in a well-defined state. It is a requirement to be free of cross- and bias current during power down. This can be configured in the LVDS port control.

Note: The line driver in the Master interface shall be turned on before the line driver in the Slave interface is activated, i.e. before sending the transmitter activation command to the Slave interface, in order to avoid receiving undefined data in the Slave interface.

Due to the strong dependency on link length and layout, as well as the susceptibility to noise and interference, it is recommended, to have the termination available on the PCB:

- Receiver internal termination mode not active (PCB termination): can be programmed inside LPCRx.TERM (LVDS Pad Control Register, see Ports chapter). The data transmitter is operated without a termination inside the differential receiver pair.
- Receiver internal termination mode active: allows to have a 100 Ω resistor internally with a +/-20% tolerance (prohibited use in multi Slave scenario).

Sleep Mode

Unless sleep mode is used the interface shall be held in the logic zero state, between frames, for a minimum of one bit period for the actual configured clock rate. This ensures to allow clock phase detection mechanism to initialize. The interface frame gap logic zero remains active until the next frame is send. The sleep mode is initiated by the interface line driver and the line receiver side, which reacts on a sleep request. This means the Master interface TX and the Slave interface TX can send out a sleep request independent of each other. The sleep request is sent out, based on missing data from higher layer TX datapath.

To enter sleep mode, the line driver shall assert a logic one for the bit period immediately following the last bit of the Frame, instead of the usual logic zero. Recognizing the sleep request at the end of the frame, the line driver shall enter a low power state in which the common mode voltage of the interface is maintained, but the differential voltage is reduced to between -5 mV and -20 mV (measured as TX_DAT relative to TX_DATX or RX_DAT relative to RX_DATX). Sleep mode shall be disabled by default (reset) on the line driver. Only after receiving a frame with sleep indication the sleep mode can be activated.

The primary objective in sleep mode is to reduce line driver current in the case where a terminating resistor is in use. By reducing the differential voltage across the resistor to nominally zero the line driver current requirement will reduce to internal bias currents only. The common-mode voltage should be maintained by the line driver, to avoid the risk of transients when full drive is restored.

The line driver and line receiver have implemented sleep mode in order to save power. In low-power state the common mode voltage of the interface is maintained.

The interface receive path may still be active and receive data during the time the transmit path is in sleep mode. During this situation it could reach an internal situation, which not allows to receive further data. With the Clear to send back pressure mechanism activated (Note: Only available in single slave mode) this situation must be reflected by sending out a frame with CTS bit cleared. As the transmitter is currently in sleep mode, it is required to wake up the transmit path, send out a CTS frame and enter sleep mode again. After the higher layer receive datapath can accept new data the HSCT control engine takes this information to send out a frame with CTS active, again. If the transmitter is still in sleep mode and new data can be accepted a wake-up mechanism is triggered and send out a CTS frame, again. After the frame is send out the interface enters sleep mode again, if still not requested to wake-up by higher layer data availability or a new HSCT frame sent.

During sleep mode there can also be the situation of sending a control command, ping answer or unsolicited status message. As this logical data type are initiated via the control interface, also in this situation leads to a wake-up of the interface and transmission of the frame. After the frame is send out the interface enters sleep mode again, if the higher layer HW still not requests to wake-up.

35.7.3.2.4 Data Rates

The Slave interface shall be ready to receive frames whenever the HSCT SysClk is running. The link shall initialize in low-speed mode with the high-speed clock generators turned off or high-speed clock from Peripheral PLL to HSCT disabled and the RxData link disabled. Subsequent speed and mode changes shall be commanded by the Master interface, except sleep mode on the RXData link, which is controlled by the Slave interface. Except when using sleep mode, both links shall default to the logic zero state, while dormant and between frames. This is required, to ensure that data will not be lost during a speed change.

Interface data rate change on both the TxData and the RxData link shall be possible without losing data. It is highly recommended to keep the link quiet during the time the interface transfer speed is changed. Changing transfer data rate on an active link does not guarantee stable data transfer.

The HSCT provides three different data speed modes

- Low speed mode
- Medium speed mode (only available on RX link)
- High speed mode

Additionally the HSCT module provides different data speed modes, derived from the crystal frequency. Taking the Automotive common crystal frequency of 20 MHz the following data rates can be produced based on the available multiplication factors. Future 40 MHz crystals are supported, too with a similar calculation procedure.

SysClk; RefClk = Crystal frequency = 20 MHz

- 5 MHz = Crystal frequency / 4 (Low Speed)
- 20 MHz = Crystal frequency (Medium Speed)
- 320 MHz = Crystal frequency * 16 (High Speed)

Low Speed mode and Medium Speed mode frequencies are generated directly out of the Crystal frequency. High Speed mode frequency requires a PLL, which takes the Crystal clock as reference clock. The following figure shows a Frequency Scheme based on 20 MHz crystal clock / HSCT SysClk as an example.

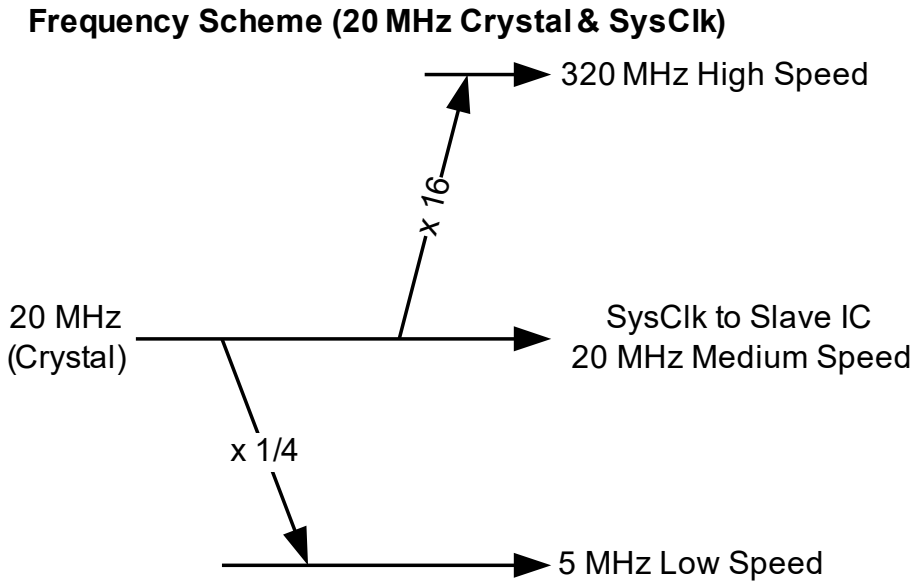


Figure 404 Frequency Scheme

Connecting a 20 MHz or 40 MHz Crystal to the system allows to generate the Automotive frequency scheme.

The following table (**Table 314**) shows the data transfer direction and the available Automotive transfer speed based on Crystal clock frequency.

Table 314 HSCT Data Rates

Direction	Data Rate	Operating Mode
Master interface to Slave interface	5 Mbit/s	“low speed mode” for startup / fallback
Master interface to Slave interface	320 Mbit/s	“high speed mode” (Single Slave only); Multi Slave divide
Slave interface to Master interface	5 Mbit/s	“low speed mode” for startup / fallback
Slave interface to Master interface	20 Mbit/s	“medium speed mode” *Note: only available, if reference clock; SysClk is at 20 MHz (Crystal)*
Slave interface to Master interface	320 Mbit/s	“high speed mode” (Single Slave only); Multi Slave divide

PLL configuration and Baud Rates

The PLL supports different speed modes and frequency modes, which will be described in this section. Each mode and transfer direction can be configured separately.

Starting from a reference clock selection in the CCU it is possible to configure a REFCLK divider (**INIT**.SRCF) for the internal clock switches and correlator circuit as well as configure the SYSCLK divider (**INIT**.SSCF), which generates the SysClk provided by the Master interface. The REFCLK DIV generated PHY_CLK is considered in Low Speed and Medium Speed mode. During configuration of the reference clock path the role of the interface (Master or Slave) needs to be considered.

The HSCT as Master Interface requires to enable the XTAL path. The preferred XTAL frequency for HSCT is 20 MHz. Based on the XTAL frequency a PHY_CLK of 20 MHz shall be configured using the REFCLK DIV. In addition the Master Interface provides the SysClk having f_{SYSCLK}, which is configured using SYSCLK DIV based on the Reference Clock frequency. Recommended SysClk out frequency is 20 MHz

The HSCT as Slave Interface requires to enable the SYSCLK PAD path. The preferred SysCLK frequency for HSCT is 20 MHz. Based on the SysClk frequency the PHY_CLK shall be configured using REFCLK DIV. Also in this case 20 MHz is recommended. A 10 MHz PHY_CLK leads to not availability of medium speed (20 MBaud).

The data path high speed transfer rate (Baud rate) can be configured independent per link direction using the **INIT**.TXHD and **INIT**.RXHD.

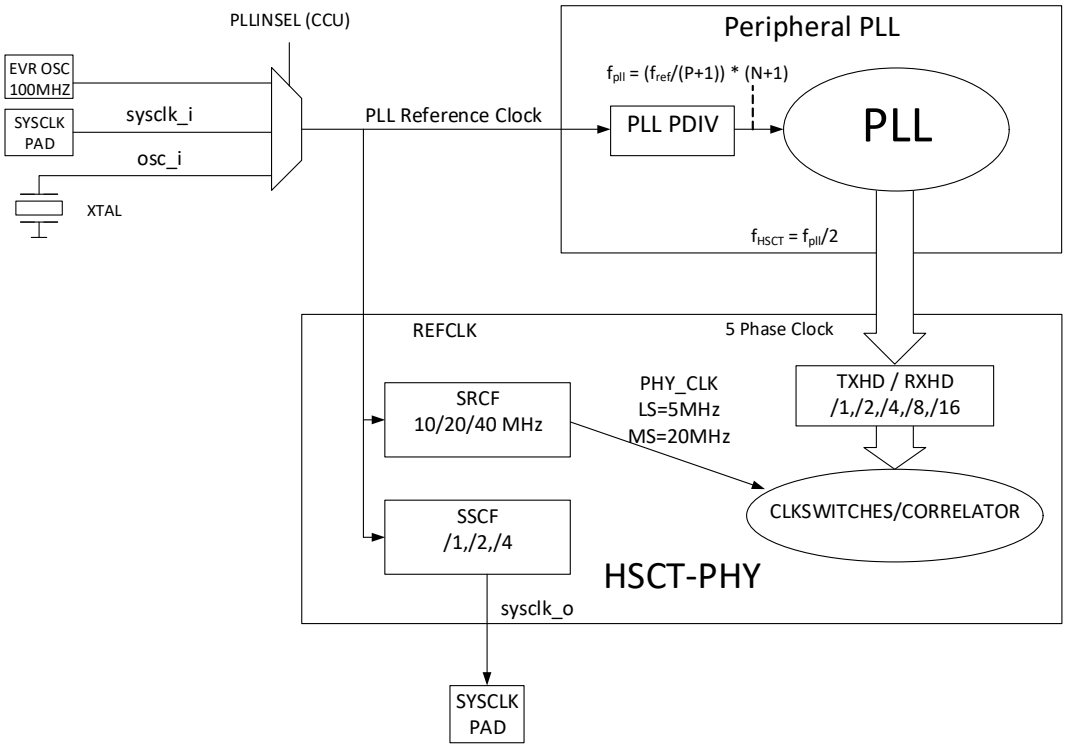


Figure 405 Frequency generation and clock divider

The following table gives more detailed information about the Baud rate configuration and the configuration bit's involved. The peripheral PLL reference frequency and their PLL frequency control word Multiplication factor (Peripheral PLL PDIV and NDIV) defines the PLL speed and are used to configure the output frequency of the PLL. The interface baud rate can be configured separately in TX direction (TXHD) and RX direction (RXHD). The table below (*** 'High Speed Mode selection' on page 96 ***) exemplarily shows the TX direction.

Table 315 High Speed Mode selection

PLL Speed fHSCT [MHz]	**320**	**320**	**320**
PLL reference frequency [MHz]	40	20	10
PLL PDIV (Peripheral PLL) [hex]	0x1	0x0	0x0
PLL NDIV (Peripheral PLL) [hex]	0x1F	0x1F	0x3F
TXHD 1/1 [MBaud] (max Baud single Slave)	320	320	320
TXHD 1/2 [MBaud] (max Baud two Slave)	160	160	160
TXHD 1/4 [MBaud] (max Baud three Slave)	80	80	80
TXHD 1/8 [MBaud]	40	40	40
TXHD 1/16 [MBaud]	20	20	20

35.7.3.2.5 Correlator

Data transfer from the Master interface to the Slave interface can either be in “high speed mode” or “low speed mode” (**for example Crystal**/4 at 20 MHz HSCT SysClk or SysClk/2 at 10 MHz SysClk). A correlator is used to select the optimum sample phase for the frame based data transfer.

Phase Selection

Every frame starts with a 16-bit sync field (see **Figure 406**) containing a synchronization pattern. Note that the sync pattern starts with 1, so a new frame always starts on the rising edge. In between frames, there is always at least one bit 0 in normal mode. By sending 1 immediately after the last bit in a frame, the Master interface controller demands sleep mode as described in Section Sleep Mode. Neither the Master interface nor the Slave interface may assume that the other side will maintain data phasing between frames. The receiving interface shall use the sync pattern to determine the start time of the frame relative to the local clock.

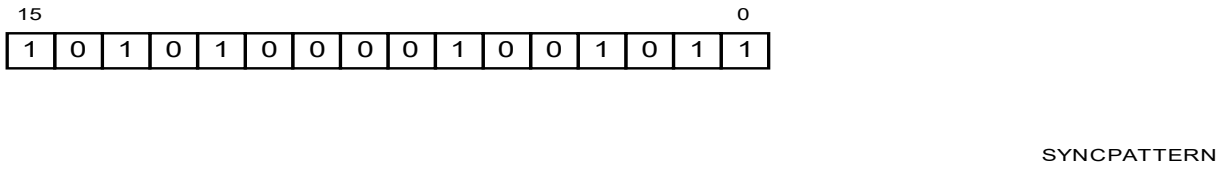


Figure 406 Synchronization Pattern

The synchronization pattern is only needed for selecting the optimum sample phase, so it is not feed through to the output register of the correlator.

Payload Size Decoding

Every frame consists of three parts: a 16-bit synchronization pattern, an 8-bit header and the payload field which can be 8, 32, 64, 96, 128, 256 or 288 bits. The header contains information about payload size and logical channel type. Although decoding the header is task of the deframer, the payload size shall also be evaluated inside the correlator for finding the end of a frame.

The header is sent most significant bit first. The first 3 bits (bit5, bit6 and bit7) contain the payload size of this frame (see **Table 316**).

Table 316 Payload Size Coding

b7:b5	Payload Size (bits)	Total Frame Size (bits)
000	8	32
001	32	56
010	64	88
011	96	120
100	128	152
101	256	280
110	512	536 (not supported)
111	288	312

Note: In a harsh Automotive environment a payload of 512 bit may cause increased error bit streams, due to the long distance from one synchronization pattern to the next. This is the reason why this data transfer length is not supported.

The correlator shall decode the payload size field and count the number of received bytes. After the last byte in a frame has been fed through to the correlator output register, the tuning of the multiplexer unit becomes invalid. In between frames at least one bit 0 shall be transmitted. The comparison procedure is started again as soon as there is a 1 at the MSB position in the shift registers.

The 8-bit header and the payload are used as input for the deframer block.

Note: The sync pattern and inter-frame bits are not passed to the deframer. An error in the sync pattern generates an error and is detected by the Physical layer.

Transmission initiated Sleep Mode - bringing receiver path into sleep mode

By sending a 1 immediately after the last bit in a frame, the interface indicates that its LVDS line driver is sent to sleep mode, and suggests sleep mode to the receiver LVDS.

To leave sleep mode, the Master interface shall send 8 bits of 0. This is performed in high speed mode and in low speed mode and allows the receiver to wake-up.

35.7.3.2.6 Jitter Budget

The system clock is distributed directly from the Master interface to the Slave interface (see **Figure 407**).

Subsequently, all jitter investigations considered in the following sections are referenced to the system clock f_{SYSCLK}.

Each system clock distribution has an associated filter function that comprehends the worst case combination of PLL bandwidth/peaking and equivalent jitter. The effective jitter seen at the LVDS receivers clock-data recovery inputs is a function of the difference in LVDS receiver and LVDS transmitter PLL bandwidth and peaking convolved with the jitter spectrum of the system clock.

System Clock variations contain jitter over a wide range of frequencies, some of which will be tracked by the LVDS receiver or otherwise removed by the combination of the LVDS transmitter and the LVDS receiver PLLs. Consequently, the nature of the filter functions is in part dependent on the way of how the system clock is distributed. In general, the system clock jitter is filtered by the PLL difference function which comprehends the “mismatch” between LVDS transmitter and LVDS receiver PLL and also accounts for the impact of the transport delay.

Direct System Clock Distribution

This architecture implies direct distribution of the crystal clock as system clock from the Master interface to the Slave interface. Hence, the system clock is directly supplied to both LVDS transmitter PLL and LVDS receiver PLL. Hence, a major part of the clock jitter is sourced equally through LVDS transmitter and LVDS receiver PLLs. **Figure 407** illustrates the direct system clock distribution, depicting noise sources, delay, and PLL transform sources. The amount of jitter appearing at the data recovery block is then defined by the transfer function difference of the LVDS transmitter and LVDS receiver PLLs, and by the sum of the clock jitter of the LVDS transmitter PLL, $X_{clkMaster(s)}$, and the LVDS receiver PLL, $X_{clkSlave(s)}$.

Based on the above clock architecture, a difference function may be defined that describes the “mismatch” between LVDS transmitter and LVDS receiver PLLs. As a first approximation second order transfer functions are assumed.

The jitter of the direct system clock distribution is described by the phase noise power spectral density function as the absolute square of $X_{d(s)}$, as follows.

$$|X_d(s)|^2 = |X(s)|^2|H_1(s)e^{-sT} - H_2(s)|^2 + |X_{clkMaster}(s)|^2 + |X_{clkSlave}(s)|^2 \quad (35.3)$$

where

$$H_1(s) = \frac{2s\ \zeta_1\omega_{n1} + \omega^2_{n1}}{s^2 + 2s\ \zeta_1\ \omega_{n1} + \omega^2_{n1}} \quad (35.4)$$

and

$$H_2(s) = \frac{2s\ \zeta_2\omega_{n2} + \omega^2_{n2}}{s^2 + 2s\ \zeta_2\ \omega_{n2} + \omega^2_{n2}} \quad (35.5)$$

H2

Conversion between the natural PLL frequency ω_n and the -3 dB point is given by the following expression.

$$\omega_{3dB} = \omega_n \sqrt{1 + 2\zeta^2 + \sqrt{(1 + 2\zeta^2)^2 + 1}} \quad (35.6)$$

Under the assumption that crystal clock jitter can be ignored, i.e.,

$$X(s) \approx 0 \quad (35.7)$$

and LVDS transmitter to LVDS receiver transport delay can be neglected, i.e.,

$$e^{-sT} \approx 1 \quad (35.8)$$

the phase noise power spectral density of the direct system clock distribution can be simplified, as follows.

$$|X_d(s)|^2 = |X_{clkMaster}(s)|^2 + |X_{clkSlave}(s)|^2 \quad (35.9)$$

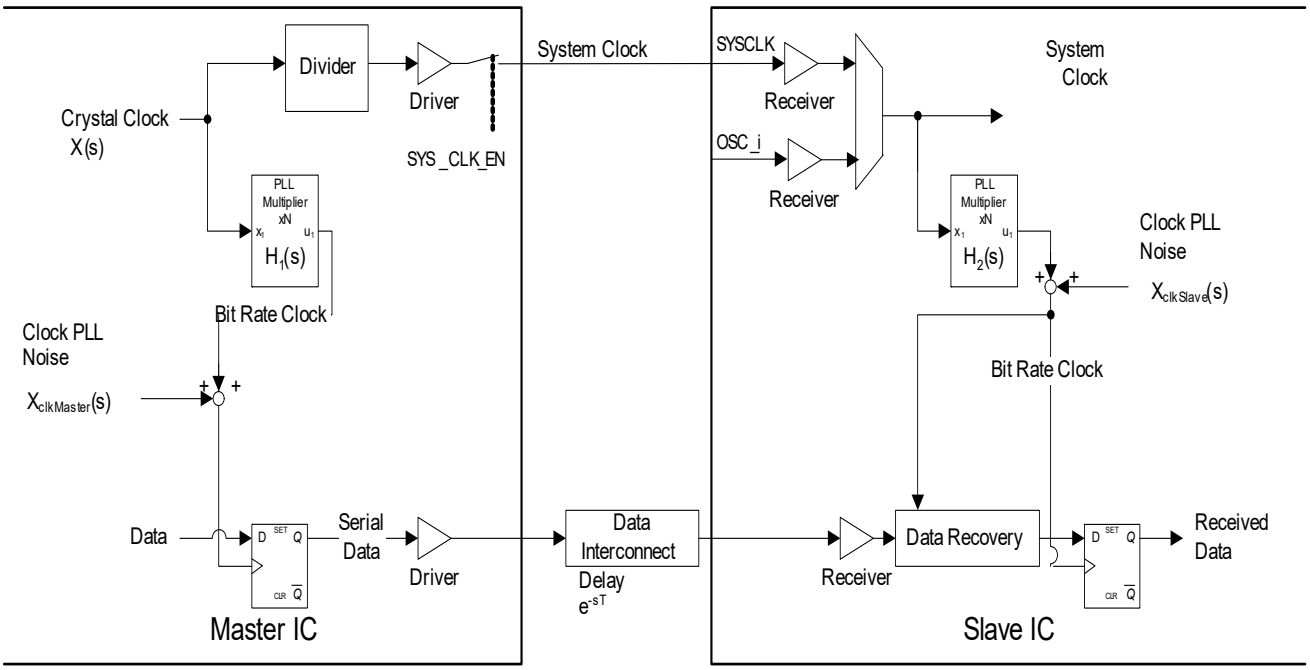


Figure 407 Direct System Clock Distribution

In the direct system clock distribution the overall jitter is given by the sum of both phase noise spectral densities from the LVDS transmitter PLL, $X_{clkMaster(s)}$, and the LVDS receiver PLL, $X_{clkSlave(s)}$.

Note: The Slave interface requires a HSCT SysClk from the Master always as reference clock. It is not supported by the interface to have a separate crystal as reference clock at the Slave interface. The Master interface generated HSCT SysClk needs to be enabled.

Jitter Budgeting - Direct System Clock Distribution

To estimate the total rms jitter of LVDS transmitter PLL and LVDS receiver PLL for data recovery the following procedure has to be followed.

- Piece wise linear integration of the additive SSB (Single Sideband) phase noise for direct clock distribution to obtain the total phase noise power of the LVDS transmitter PLL.
- Calculation of the rms jitter of the LVDS transmitter PLL as shown in **Figure 408**.
- Calculation of the total rms jitter *RJ* as the square root of the sum of the squares of the rms jitter from the LVDS transmitter PLL and the LVDS receiver PLL.
- Calculation of total jitter, i.e., random and deterministic, considering duty-cycle distortion and other effects to guarantee minimum eye opening at the interface.

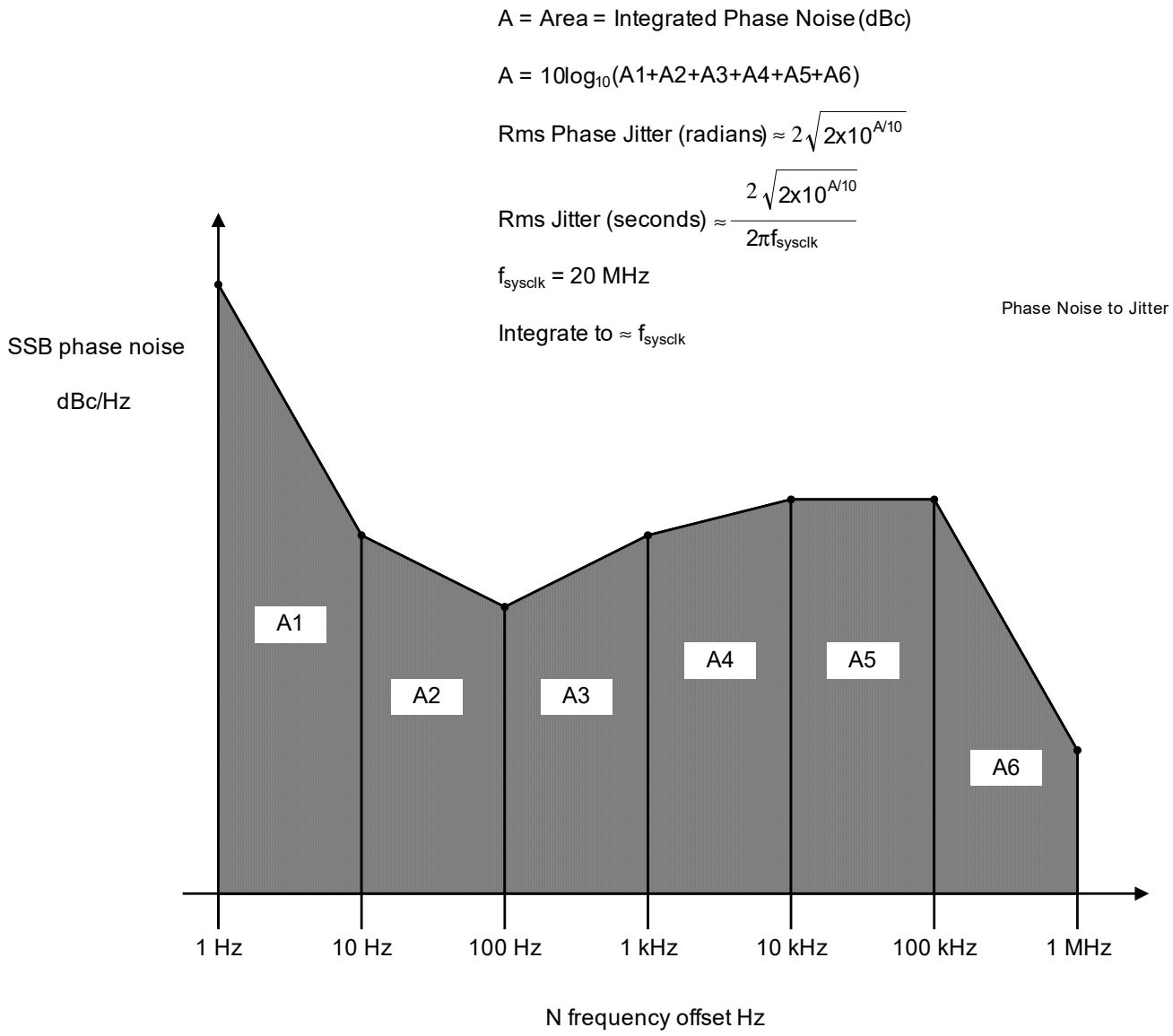


Figure 408 Calculation of Jitter from SSB Phase Noise

Calculation of Total Jitter Budget - Direct System Clock Distribution

To estimate the total jitter budget allowed for the random rms jitter *RJ* as described in the sections before, the following procedure for direct clock distribution has to be followed.

- Calculation of the jitter headroom available for the ideal jitter-free case given by the minimum eye opening, i.e., 55% of a 320 MHz period.
- Calculation of the total jitter headroom required under the assumption that the total long-term jitter *TJ* will not exceed the maximum peak-to-peak jitter specified.
- Subtraction of two phases of the oversampling frequency used for data recovery taking into account the data-recovery uncertainty in multiples of phases of the oversampling frequency. The data-recovery uncertainty is implementation specific and determined by the data-recovery oversampling factor DR_{OS}. **Figure 409** depicts the case where the selected sampling phase of the data recovery is almost one sample phase away from the

ideal sampling point. Thus, the budget left for the peak jitter is reduced by almost one sample phase and hence, the peak-to-peak jitter is reduced by almost two phases.

- The remaining sum represents the jitter budget which is the upper limit for the peak-to-peak jitter calculated from the random rms jitter distribution *RJ* described in the sections before. Given a required bit error rate BER = 10^{-12} for SysClk 20 MHz and BER = 10-9 for SysClk 10 MHz.

The jitter budget is given by the following equation.

$$TJ_{pp} = DJ_{PP} + RJ_{PP} = (DJ_{TX} + DJ_{RX}) + 2 * Q * \sqrt{J_{ABS}^2 + J_{ACC}^2}_{T} \quad (35.10)$$

Sampling Phase Selection

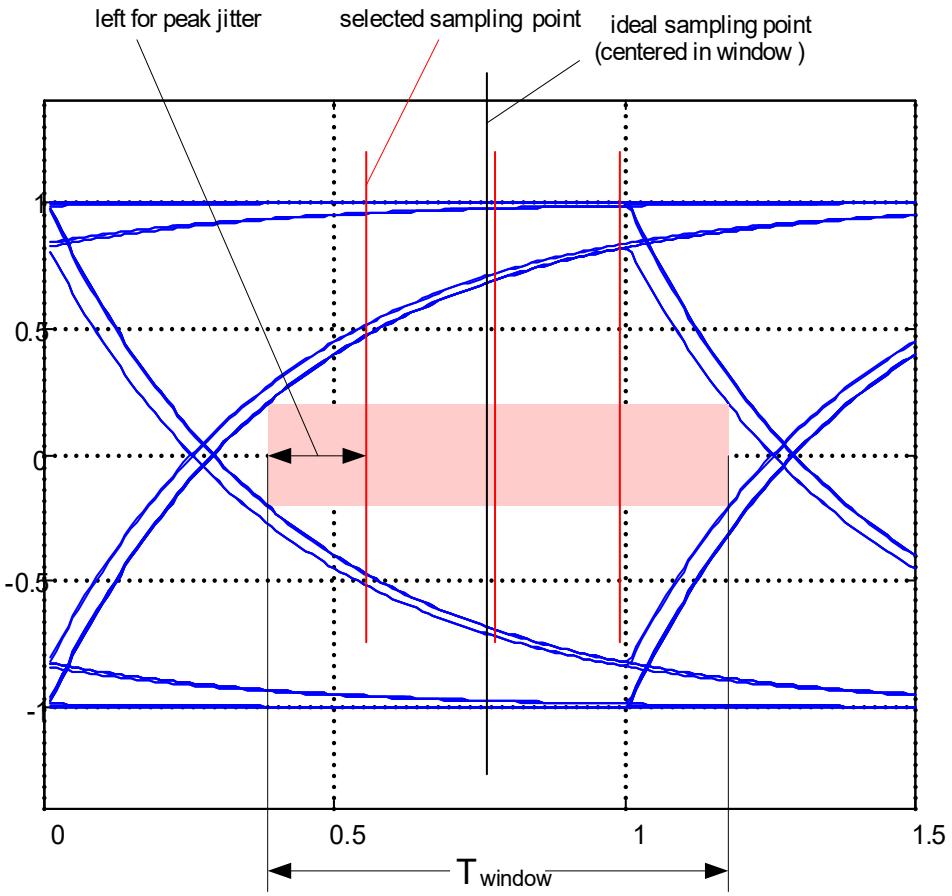


Figure 409 Sampling Point Selection

35.7.3.3 Protocol Layer

The protocol layer is between the higher layer protocol (HSSL) and the physical layer. The protocol Layer is performing the frame generation in transmit direction and extracts the payload out of a frame in receive direction, called deframing.

35.7.3.3.1 Deframer

The deframer's tasks are decoding the 8-bit header and separating the data stream according to the logical channel type.

During Microcontroller reset, the deframer shall ignore all data coming in from the correlator. This is a precaution measure to eliminate possible malfunctions due to floating HSCT lines.

Additionally the deframer checks the header for potential errors, e.g. size is matching received data or size is allowed and received type is allowed. An error in the header discards the received data.

Header Decoding

The first 8-bit in a frame, after the sync pattern has been filtered out by the correlator, is the header or frame type field. It contains information about the payload size and logical channel type coding (see **Figure 317**). Please note again that all transfers are most significant bit first.

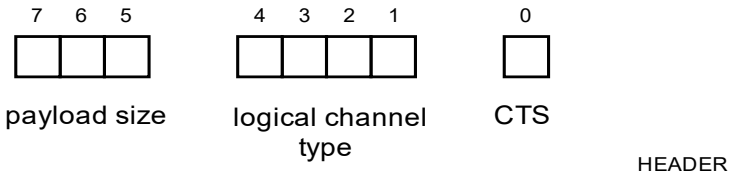


Figure 410 Header (Frame Type Field)

- The payload size coding can be seen in **Table 316** in the previous.
- Bits 1 to 4 define the logical channel type. See **Table 317** for details on the logical channel type coding. Note that some kinds of transfer are exclusively used from the Master interface to the Slave interface or from the Slave interface to the Master interface.
- Bit 0 is used as Clear To Send bit CTS for transfers from the Slave interface to the Master interface (Requires an enable via configuration). For transfers from the Master interface to the Slave interface also the Master interface can send CTS, which must also be enabled via configuration before. Also CTS from Master interface indicates the Master to the Slave, no more data can be accepted. This feature requires for the Master interface as well as for the Slave interface that higher layer indicates in HW early enough that no more data can be accepted with the next frame. This needs to be send out with the frame based CTS signaling. (Consider the transferred payload size for early enough CTS signaling via a frame.)

Table 317 Logical Channel Type Coding

b4:b1	Logical Channel Type Master interface to Slave interface (TX Link)	Logical Channel Type Slave interface to Master interface (RX Link)
0000	Interface Control; restrictions in multi Salve Mode	Interface Control (PING answer, 32-bit value; not supported in multi Slave mode)
0001	Master interface Unsolicited Status (32-bit only; not supported in multi Slave mode)	Slave interface Unsolicited Status (32-bit only; not supported in multi Slave mode)
0010	Reserved - receiver error interrupt	Reserved - receiver error interrupt
0011	CTS Transfers; not supported in multi Slave mode	CTS Transfer; not supported in multi Slave mode
0100	Data Channel A	Data Channel A
0101	Data Channel B	Data Channel B
0110	Data Channel C	Data Channel C

Table 317 Logical Channel Type Coding (cont'd)

b4:b1	Logical Channel Type Master interface to Slave interface (TX Link)	Logical Channel Type Slave interface to Master interface (RX Link)
0111	Data Channel D	Data Channel D
1000	Data Channel E (not supported, receiver generates an error interrupt)	Data Channel E (receiver error interrupt)
1001	Data Channel F (not supported, receiver error interrupt)	Data Channel F (receiver error interrupt)
1010	Data Channel G (not supported, receiver error interrupt)	Data Channel G (receiver error interrupt)
1011	Data Channel H (not supported, receiver generates an error interrupt)	Data Channel H (receiver error interrupt)
1100	Reserved - receiver error interrupt	Reserved - receiver error interrupt
1101	Reserved - receiver error interrupt	Reserved - receiver error interrupt
1110	Reserved - receiver error interrupt	Reserved - receiver error interrupt
1111	Reserved - receiver error interrupt	Reserved - receiver error interrupt

Note: A logical channel type TX transfer request of not supported or reserved types is not stopped or prevented to be send out. Receiver side is expected to react with an error indication.

*Note: The received header information can be checked by SW reading Header status register (**STAT**). An error in the header is a severe error in a running system. Application SW must handle the situation by interface link checks (PING; PING not supported in multi Slave scenario). Additionally it is suggested to restart a running stream, if data loss cannot be tolerated. Received header frames are discarded, the same as the CTS information inside this error frame.*

Payload Size for transmission Data

For a transmission data transfer it is recommended to use max 288-bit payload size. TC3xxHSCTTC3xx

Logical Channel Separation

Reflecting the logical channel type to the higher level interface, the deframer sends the frame payload. The payload, as all other parts in a frame, is sent most significant bit first. The HSCT in the multi Slave scenario is used as a data protocol tunnel engine. Any addressing of the connected Slaves is done at a higher layer in HW.

- All data received on and decoded as channel A to D data are sent via an 32-bit data interface to the higher layer HW level.
- Data TC3xx can be sent using the 32-bit, 64-bit, 96-bit or 288-bit payload size.

Note: Receiving a 8-bit header size on a logical channel results in an header error.

35.7.3.3.2 Framer

The framer combines data and control information from the higher layer protocol to the frame payload. Note that the payload again is sent most significant bit first. The framer codes the logical channel type and the payload size in the header according to the HSCT header format (see **Figure 410**). Then a complete frame consisting of 16-bit synchronization pattern (see **Figure 406**), 8-bit header and payload field is put together.

The following logical channels are used for transmissions between Master interface and Slave interface:

- Data from the transmit Unit: The frame will be sent to the receiving interface side as soon as the higher HW layer transmit path contains enough data for the required payload size. The appropriate values will be signaled with the payload size signal and in the logical channel type field in the header. For data transfer, only

the logical channel types Data Channel A to D are be used! Receiving a header type reflecting the type Data Channel E to H generates an error.

- Unsolicited status information can be send by Master interface and Slave interface based on a 32-bit payload size. An unsolicited status can be used to send an unexpected and urgent message to the other side. Receiving an unsolicited message results in an interrupt at the receiving side. The Unsolicited status information can only be send in single Slave scenario. In multi Slave Scenario this feature is not available.

Note: The Slave interface in single Slave scenario (RX link) can send out a control command (channel type encoding Hex 0), which is the predefined PING answer having a defined 32-bit payload. All other control commands are not available for the RX link to send. Receiving a header types not supported on RX link or TX link generates an error.

The handling of all frame parts is done by the framer. For determining the end of a frame, the framer shall count the number of payload bits sent to the receiver side. In between frames bits 0 shall be transmitted.

Clear to Send (CTS) - only available in single Slave scenario

The HSCT interface is able to use “data pull” in both directions (Master interface to Slave interface and Slave interface to Master interface) to perform flow control on the transmission data path. Therefore the CTS bit in the header is asserted, when the other side is allowed to send more data, and it is negated when the other side should stop transferring data.

A CTS deactivation can be send out in a “normal” transmit frame, indicated by the header bit. The following payload is not discarded at receiving interface side and processed normally. CTS deactivation can be send out in a CTS type transmit frame, if no actual transmit data is available. The clear to send information is indicated by the header CTS bit deactivation. Following payload is discarded at receiving side. A received clear to send deactivation requires to stop the transmission immediately after the actual frame being in transmission or prevent to send a new frame, if currently no transmission is active. Control frames can still be send.

The clear to send is independent of the channel having caused the back pressure. It simply stops the data channel transfer. As soon as the traffic jam is resolved the CTS bit in the next transmitted frame is activated again, or a separate CTS frame is send with the CTS bit activated.

Higher Layer software must analyze which channel caused the traffic jam and can send the information to the other side.

The usage of the CTS signaling is optional. CTS signaling can be disabled by disabling the CTS_FRAME **CTSCTRL**.

Per default the CTS bit in the header is set to 1. When the situation in the receive data path is reached, where no more data can be accepted , the CTS bit shall be reset to 0 in the header of the next frame sent. If there are currently no data in the higher layer module transmission data path to be sent in a new frame, a dedicated CTS transfer according to the logical channel type in **Table 317** is generated. This transfer always uses the 8-bit payload size containing dummy bits. The payload in a CTS transfer will be discarded by the receiving side. Generation of dedicated CTS frames can be disabled by resetting bit CTS_FRAME in register **CTSCTRL** to 0.

The CTS bit is set to 1 again when the receive data path at the higher layer HW protocol allows to accept new data. This is reflected by activating the CTS signal again.

The CTS signals are full duplex signals and are side information in the transferred data frame. The deframer filters the TXCTS information from the received frame header. The TXCTS is presented to the higher layer TX interface frame based. Changes within a frame is not intended. At the higher layer receiver side the higher layer does activate the RXCTS signal, if no more data can be received. The Framer sends the RXCTS information with the header of the next frame.

Sleep Mode for line driver path

TC3xx offers an optional transmission sleep mode in between frames. Whether the transition to the transmission sleep mode can be performed, depends on the actual symbol data rate, as well as the protocol payload size, because it takes the LVDS line driver some time to wake up from sleep mode, as defined in data sheet. In sleep mode, TC3xx suggests sleep mode to the other side by sending an 1 immediately after the last bit in a frame. Setting the SLPEN bit field to 1 enables Transmission sleep mode.

Transmission sleep mode shall be triggered after every frame in case there is no more data to be send from HSCT or higher layer HW. When Transmission sleep mode shall be triggered to the opposite receiving side, the physical layer inserts a 1 after the last bit in a frame, and activates the sleep to the transmission LVDS driver.

The sleep signal is reset as soon as a data frame transmission from higher layer HW requests a wake-up due to availability of data , or when any HSCT frame information shall be sent, . These events trigger the wake-up counter. The LVDS line driver becomes ready for operation within the time as defined in the data sheet. This wake-up time is converted to a counter value, derived from Register SLEEPCTRL.WKUPCNT and clocked by SRI clock frequency. After the wake-up time the physical layer starts sending eight bits 0 for signalling the start of a new frame to the receiving side. Subsequently the framer generates the new frame to be sent to the receiving side.

Note: If the sleep mode is used, the application software has to configure the counter SLEEPCTRL.WKUPCNT to a value corresponding to the LVDS pad wake-up time defined in the data sheet.

35.7.3.3.3 Interface Control

The Master interface is configured via a CPU based configuration. The Slave interface can be configured by sending an interface control command from the Master interface. The interface control command is used to configure the Slave interface into the defined mode. The receiving Slave interface shall take no action, if a reserved payload value is received. Instead it generates an error interrupt. HW reacts on the received control command.

After startup the interface is enabled to work at low speed (based on 20 MHz HSCT SysClk) and the PLL can/might be off (if not used by other peripherals), no matter of the mode it had been before. Therefore shutting down TC3xx causes the interfaceTC3xx to leave loopback-mode, clock test mode and sleep mode, if it was previously in any of these modes, and forces the receiver and transmitter to power down mode.

The interface transmitter path is disabled per default. It shall be enabled by using the appropriate interface control, which is a payload command for the Slave interface and a control via SW for the Master interface.

Note: No data can be transferred to the other side as long as the transmitter is disabled. In this case all data in TX direction are discarded!

The Interface Control Logical Channel transfer command always uses the 8-bit payload size and is send by the Master interface only. Payload is send MSb first. The interface control payload values are shown in **Table 318**. All other payload values are reserved for future use. Whenever a reserved payload value is received it shall be ignored and reported by an error. In addition the table is enhanced with a column describing the availability of the command in multi Slave scenario.

Table 318 Interface Control Payload Values

Value (hex)	Function	Multi Slave availability[1)]
00_H	"ping" (send by Master interface. Slave interface sends back a fixed 32-bit payload result.)	No
01_H	Reserved	No
02_H	Slave interface clock multiplier start (in preparation for high speed mode)[2)]	Yes
04_H	Slave interface clock multiplier stop[3)]	No
08_H	Select low speed mode for transfers from the Master interface to the Slave interface	Yes
10_H	Select high speed mode for transfers from the Master interface to the Slave interface	Yes
20_H	Select low speed mode for transfers from the Slave interface to the Master interface	Yes
40_H	Select medium speed mode for transfers from the Slave interface to the Master interface. *Note: Not allowed, if SysClk = Crystal/2 (10 MHz) at Slave interface.*	Yes
80_H	Select high speed mode for transfers from the Slave interface to the Master interface	Yes
31_H	Enable Slave interface transmitter[4)]	No
32_H	Disable Slave interface LVDS transmitter[4)]	Yes
34_H	Turn on clock test mode (Send 101010… on Rx line continuously using currently configured Rx line rate; cancelled by issuing the "test mode off" command. Internal transmit and receive path is disabled. Received data is discarded. The only exception to this is the "test mode off" function, which turns off the clock test mode, and re-initializes for normal operation, retaining whichever interface speed mode is currently configured.	No
38_H	Turn off test mode (cancel clock test mode and payload loopback)	No
51_H	Enable Slave interface LVDS transmitter of Slave 1 (only available in Multi Slave Mode)	Yes
52_H	Enable Slave interface LVDS transmitter of Slave 2 (only available in Multi Slave Mode)	Yes
53_H	Enable Slave interface LVDS transmitter of Slave 3 (only available in Multi Slave Mode)	Yes
FF_H	Turn on payload loopback (incoming frames at Slave interface are looped back until cancelled by a Frame containing "test mode off"). Internal transmit and receive path is disabled. Received data is discarded. The only exception to this is the "test mode off" function, which turns off the loop back mode, and re-initializes for normal operation, retaining whichever interface speed mode is currently configured. *Note: Requires same speed configuration for RX- and TX-link.*	No

1) In Multi Slave mode, it is not allowed for the user to use the commands marked as unavailable in Multi Slave mode.
2) The Slave PLL (Peripheral PLL) on a TC3x Product is also clock source for other modules on the SoC can only be switched on from Master side, using Higher layer Protocol commands or with initialization SW running on the Slave side. The command shows no effect on a TC3x product at Slave Interface side. For ASIC's and backward compatibility it is still supported.
3) The Slave PLL (Peripheral PLL) on a TC3x Product is also clock source for other modules on the SoC can only be switched off from Master side, using Higher layer Protocol commands or with initialization SW running on the Slave side. The command shows no effect on a TC3x product at Slave Interface side. For ASIC's and backward compatibility it is still supported.
4) For more details check **Figure** RX link State Machine

Note: Before changing the speed mode SW has to take care that data Transmission on both links is stopped and the interface remains quiet.

As a general principle, after a mode change the Master interface should request a data transfer from the Slave interface to stimulate a frame, which confirms the correct reception and the mode change was successful. The PING function, available in the interface control logical channel in single Slave scenario only shall be used to verify a successful speed change. The Master interface generated PING has to be send in system defined time intervals, to verify the link is still alive. Sending out a ping requires system SW to activate a system timer. Not receiving the PING answer in single Slave scenario within the system defined time, the link cannot be considered as stable and a new initialization is required. The Slave interface sends back in single Slave scenario only a fixed, known response based on a control command, at the currently configured link rate on RxData line.

Data Baud Rate configuration from Slow to Fast:

The slow to fast interface speed change can apply to only one of the interfaces data transfer directions or to both at the same time. At both sides Clock multipliers are needed, when either interface is to run at high speed. If the RXData link is not enabled, but needs to be, the Master interface sends “Enable Slave Interface transmitter”. If the Slave interface clock multiplier is not already running the Master interface sends “Slave interface clock multiplier start” to the Slave interface at low speed (on TC37x products via higher layer protocol commands the Peripheral PLL register needs to be directly configured and activate the PLL). The Master interface enables its own clock multiplier, if it is not already running. After the proper settling time, the Master interface sends “select high speed” to the Slave interface at the currently configured speed for the TXData link. The Slave interface switches the requested interface to high speed. The Master interface switches the requested interface to high speed. The Master interface sends a frame to the Slave interface requiring a response at the currently configured speed for the TxData Link. The Slave interface responds with a frame to confirm the mode change has worked. In order to select high speed for both RXData and TXData links the Master interface sends “select high speed” for the RxData link first and then for the TxData link. As described before it is suggested to send the PING for verifying a successful speed change. The PING command can only be send in single Slave scenario. For multi Slave scenario the validity of the speed transition shall be performed using higher layer protocol.

Fast to Slow:

The fast to slow interface speed change is the converse of the slow-to-fast transition. Both clock multipliers must be running for either interface to operate in high-speed mode.

Before configuring, enabling and entering high speed mode it shall be assured that the high speed clock is available. This can be achieved by reading back Peripheral PLL lock information from PLL status register via higher layer protocol register access mechanism. If a command for entering high speed mode is received and the PLL is not locked, the command is ignored and lost. Interface control command value 0x02 (PLL start) or 0x04 (PLL stop), are not supported by a TC3x products as the high speed reference clock comes from a common PLL also used by many other peripherals. Switching on/off via commands might lead to safety issues. PLL switch on/off must be performed by register configuration via higher layer protocol or via SW routines established in the application SW. Nevertheless the command is produced and send out if required for backward compatibility.

Figure 411 shows the TX transmission state machine, which controls transfers from the Master interface to Slave interfaceTC3xx. The interface control value int_ctrl represents the received interface control payload value according to **Table 318**.

Note: Application Software has to ensure, that no data communication is active on RX-link or TX-link during the time the interface speed is changed. Otherwise a save speed change without data loss can not be guaranteed interface.

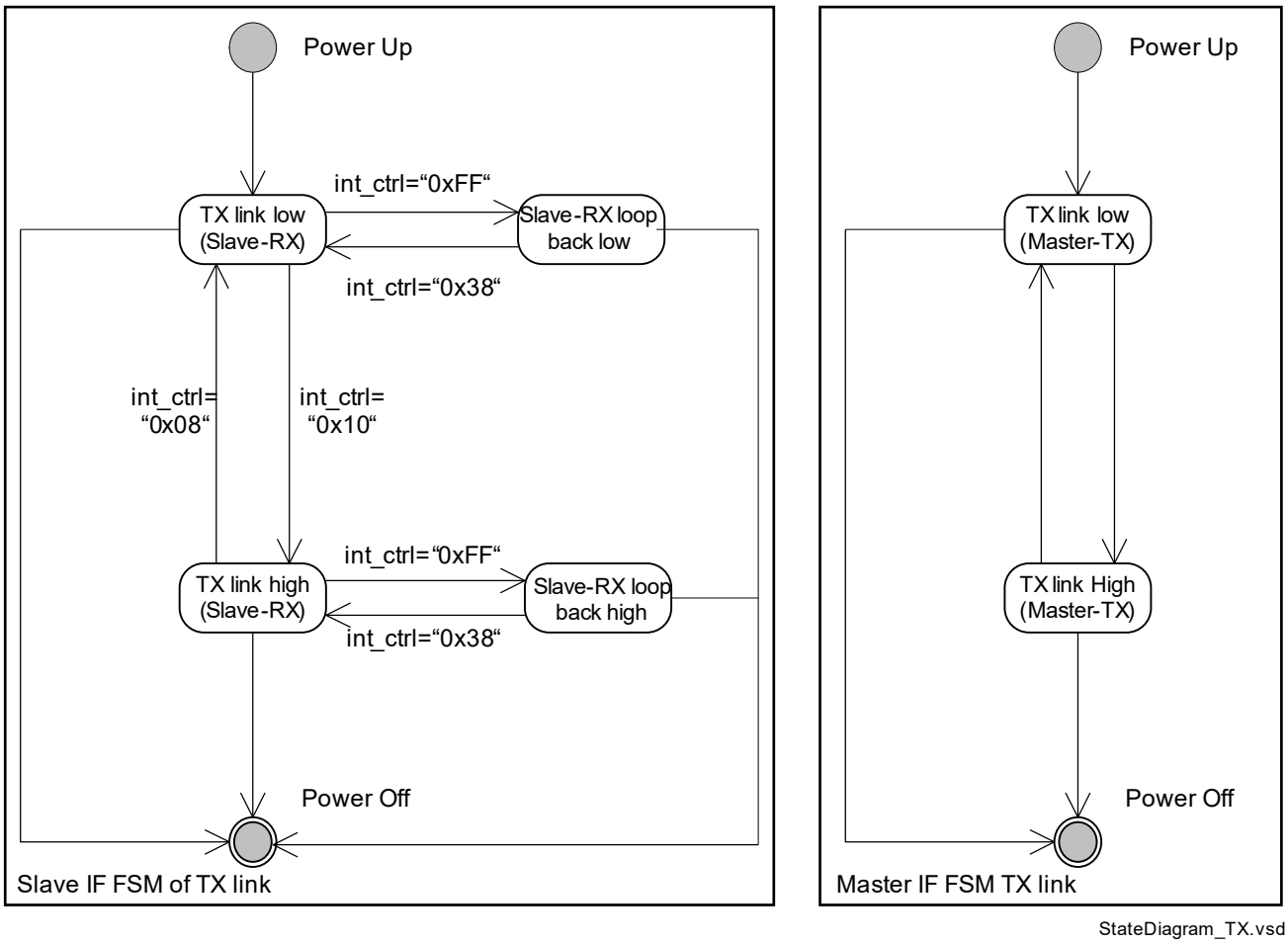


Figure 411 TX link State MachineTC3xx

The loopback mode (only in single Slave scenario) is used for test and verification to identify BER. When the Master interface demands loopback mode, the Slave interface internally loops back all received payload values in the following frames using the currently configured clock speed on the receive data link, without applying further processing internally. (Transmit and receive chains internally shall be disabled.) The header configuration is the same as in the incoming frame and the CTS bit in the header shall be asserted. Payload contents are ignored at Slave interface in loopback mode, except for the "turn off test mode" interface control value, which ends loopback mode. The "turn off test mode" frame shall not be looped back to the Master interface. After leaving loopback mode, the interface is prepared for normal operation at the previously used clock speed.

Note: The "loopback on" function has been intentionally coded at a large Hamming distance from the other codes (minimum of 5, usually 7, thus requiring at least five bit-errors in the Header field for another function to be corrupted into "loopback on".

The RX link state machine, which is responsible for controlling transfers from TC3xx to the Master interface, can be seen in **Figure 412**. The state machine is implemented on Slave interface at transmitting path. The interface control value int_ctrl represents the received interface control payload value according to **Table 318**.

HSCT

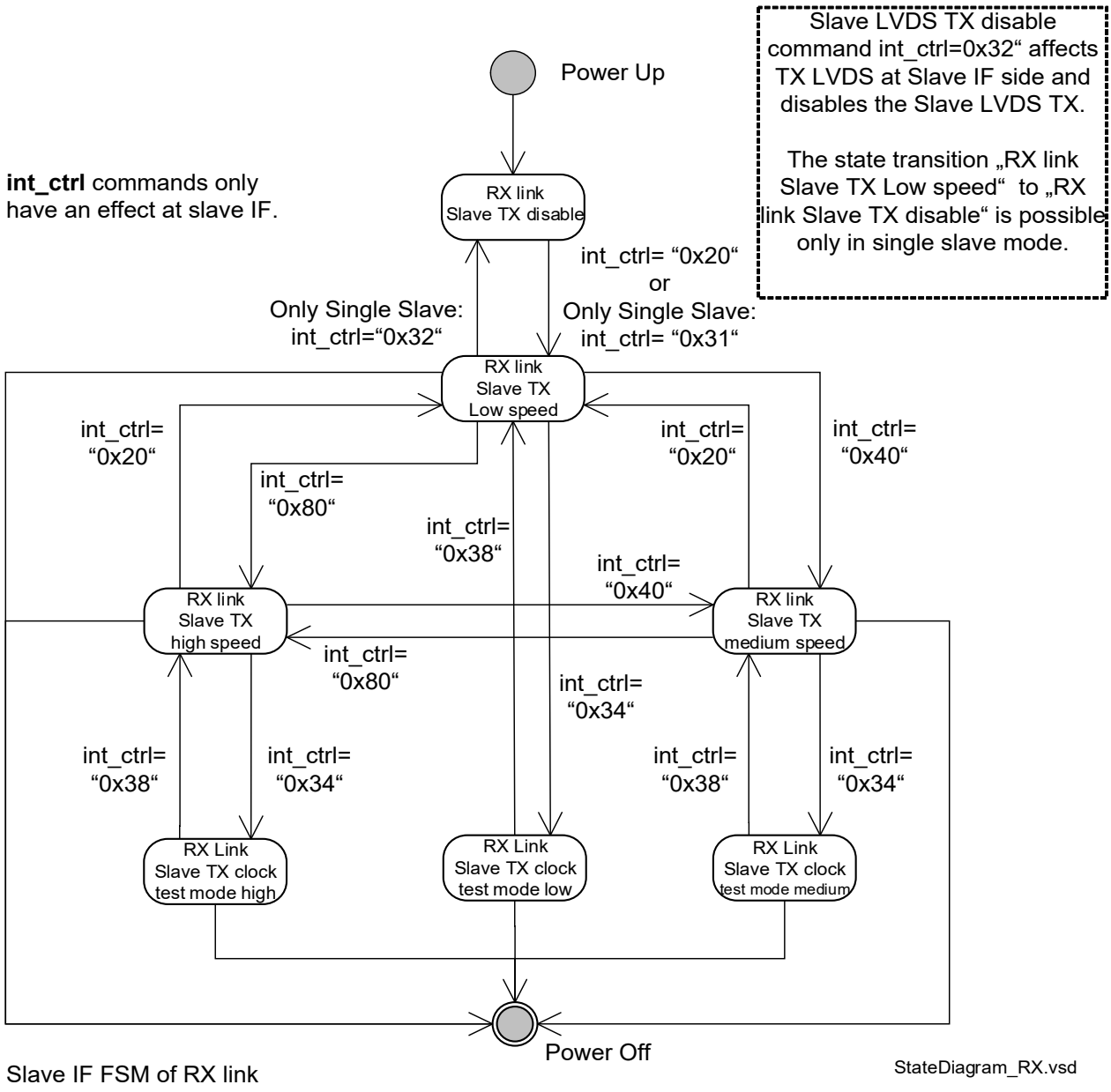


Figure 412 RX link State Machine

The Slave Interface TX, which is the RX link of the HSCT interfacing can be activated from disabled into Low speed mode using command “Select low speed mode for transfers from the Slave interface to the Master interface” (0x20) or in Single Slave mode also the command “Enable Slave interface LVDS transmitter” allows to change into “RX link Slave TX Low speed” mode. Receiving the command “Disable Slave interface LVDS transmitter” in any of the other states a disable of the LVDS TX at slave side is achieved, but no state transition. This means for example being in high speed mode and receiving the “Disable Slave interface LVDS transmitter” command only the TX LVDS is disabled, but the state remains in high speed.

Note: The Medium speed mode is only available, if the reference clock (SysClk) is running at 20 MHz.

In clock test mode (only in single Slave scenario) the transmitter sends ‘101010 …’ continuously using the currently configured clock speed. Clock test mode can be left by sending the interface control value “turn off test mode”.

The ping function in TC3xx always uses the 32-bit payload size sending back the value 0xABCDEF01 (see **Figure 413**) using the logical channel type Hex 0 (PING Status - on RX link).

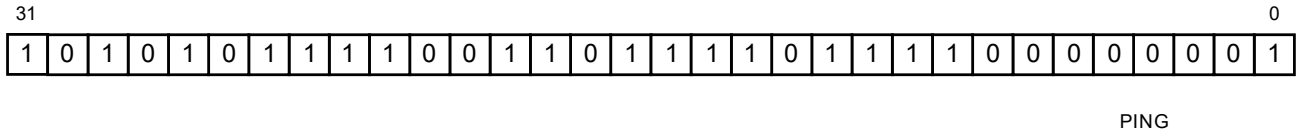


Figure 413 The Slave interface “ping” Response

The Interface control commands to configure the Slave interface have to be written into the Master interface register **IFCTRL**.IFCVS. For sending the control frame a logic 1 needs to be written to the **IFCTRL**.SIFCV register to trigger the frame transfer. The 8 register bits **IFCTRL**.IFCVS represent the same functionality as defined for the interface control payload values. Coding is as shown in **Table 318**. The register field **IFCTRL**.IFCVS shall be written with the appropriate interface control payload value for every state transition shown in **Figure 411** and **Figure 412**. The interface control in the Slave interface can alternatively be performed by programming the **IFCTRL**.IFCVS. This allows controllability of the Slave interface besides the frame based configuration. The Master interface is controlled by register configuration of physical layer only.

In **Table 319** the processing times for the time consuming interface control transitions (defined in **Table 318**) are listed.

Table 319 Processing Times for Interface Control

IFCVS (Hex)	Transition Description	Timing parameter reference (datasheet)
02_H	Slave interface clock multiplier off to Slave interface high speed clock avail.[1)]	n.a.
04_H	Slave interface clock multiplier on to Slave interface clock multiplier off.[2)]	n.a.
08_H	TX high speed mode to TX low speed mode	t_{HSLS}
10_H	TX low speed mode to TX high speed mode	t_{LSHS}
20_H	RX high or medium speed to RX low speed mode	t_{HMSLS}
40_H	RX high or low speed to RX medium speed mode	t_{HLSMS}
80_H	RX low or medium speed to RX high speed mode	t_{MLSHS}
31_H	TX disabled to TX low speed mode	t_{tx}

1) TC3xx Slave IF shows no action, if this command is received. PLL is automatically activated after power on.

2) TC3xx Slave IF shows no action, if this command is received. PLL switch off to be done using higher layer protocol command, if required.

35.7.3.3.4 Power up sequence

The power up sequence of the HSCT Physical Layer and the LVDS IO requires to have the stable bandgap and bias distributor voltages before they can be enabled. Bandgap enabling and bias distributor stabilization is automatically executed by HW after power on reset and takes t_{SET_LVDS} time. For HSCT master, LVDS TX/RX PADS can be enabled in two possible ways depending on the value of bit TEN_CTRL/REN_CTRL set in the LVDS PAD

Control register (LPCR). If TEN_CTRL/REN_CTRL =0 (port controlled condition), the LVDS TX/RX PADS are enabled in PORTS control by enabling the TX_EN/RX_EN bit in the LPCR register. If TEN_CTRL/REN_CTRL =1 (HSCT controlled condition), the LVDS TX/RX PADS are enabled via bit TX_DIS/RX_DIS in transmission disable register (DISABLE). The required time to power up LVDS TX PADS is about t_{tx} and LVDS RX PADS is t_{rxi}. For HSCT Slave interface mode, the slave LVDS TX PADS are enabled/disabled with interface commands from master and LVDS RX PADS should always be enabled. Therefore, TEN_CTRL and REN_CTRL bits in the LVDS PAD Control register needs to be enabled by startup software during initialization process. After the LVDS is ready the physical layer can be powered by enabling **Configuration Physical Layer Register**.PON. On enabling this bit the module is ready to transfer data after t_{PON}time.

Note: For actual values of the parameters please check the HSCT parameters section in the product specific data sheet.

35.7.3.4 Use Cases

The HSCT is as the name says a High Speed Communication Tunnel, which is used after initialization phase as high speed data transfer tunnel by higher layer protocol engines like HSSL. The interface is intended for chip to chip communication. The Micro Controller (MC) is from a system point of view intended to be the Master interface, if it owns the crystal. The connected Slave interface get's the clock (HSCT SysClk) provided by the master and can be operated without an additional reference clock for the internal PLL's. The reference clock at the Slave interface clocking the PLL is used to clock the data recovery path. The interface role definition (Master or Slave) is not allowed to change after the interface initialization phase.

There are two major UseCase scenarios in TC3xx the first one is the LVDS based point to point connection and the second is a point to multi point connection.

In both cases it is highly recommended to follow IFX PCB layout guidelines, which can be take as a reference to design a reliable PCB and meet the defined BER of 10-12.

35.7.3.4.1 HSCT point to point LVDS based communication

The HSCT point to point LVDS based communication is a full-duplex link in TX and RX direction, which allows independent data transfer per link direction. It allows max Baudrate of 320 MBaud with continuos data transfer in both directions. Subsequently some use cases are listed, which are an example of potential system scenarios.

MC to ASIC or FPGA

The MC (Master) is connected to the ASIC (Slave) and allows data streaming in both directions derived from higher layer HW protocol (e.g. HSSL). In addition the HSCT is used by the higher layer HW protocol to access Slave control register (read/write) and have therefore full controllability. This allows the MC to enhance it's system functions with the functions available in the connected ASIC or FPGA.

MC to MC

This use case allows to connect two MC's together, whereas one of them gets, from the Interface point of view, the Master function and the other the Slave function. Potential application would be register control or data streaming via higher layer HW protocol (e.g. HSSL).

35.7.3.4.2 HSCT point to multi point LVDS based communication

The HSCT is enhanced with the capability to have multiple Salves (max 3 Slaves) connected to one Master. Scenarios like MC connected to three ASICs or MC connected to two ASICs and one MC are supported. The initial Master configuration is fix and is not allowed to change during operation.

The connection of two ASIC's (or ASIC and 1 MC) via LVDS allows max Baud rate of 160 MBaud.

The connection of three ASIC's (or 2 ASIC's and 1 MC) via LVDS additional reduces the max Baud rate to 80 MBaud.

The communication with each of the connected Slave devices is time sequential, derived by the MC. Only one Slave can be addressed (selection by Higher layer HW protocol) and communication must be finished before the next Slave can be addressed. Rest of the Slaves stay quiet during the time one slave is addressed (selected). Higher Layer SW running on HSSL ensures this behavior and also addresses the respective slave device. Several transmitter driving the same LVDS link at once is prohibited. Electrical collisions are not allowed.

During Slave configuration phase HSCT commands are used and the Master is addressing all Slaves based on a Multi-Drop communication.

To achieve this scenario's careful high speed PCB design based on MicroStrip trace lines is required with well routed differential trace lines on PCB and good termination at the receiver input. In the multi Slave scenario the termination resistor needs to be a discrete 100 Ohm +/-5% or with less tolerance. An internal LVDS resistor is not supported in multiple Slave scenario. The location of the discrete resistor is suggested to be placed as close as possible to the Slave with a max geometrical distance from master defined (check IFX PCB layout guidelines for more details).

The LVDS IO (based on IEEE1596.3) offers several driver swing configurations, which allow application based adaptation.

35.7.3.5 Suspend, Sleep and Power-Off Behavior

Generally, the HSCT module transmits and receives data and control information via the differential signal interface.

A power down of the module can be done in any situation of the transmission. It is strongly recommended to have the transfers stopped from the higher layer protocol before sending the HSCT layer into power down state. In order to power down the module in a well defined way, the user software must take care that the power down request is issued after finishing the transfers or stop the transfers before. Issuing an power down within a transfer - all data is lost.

35.7.3.5.1 OCDS Suspend

OCDS suspend request, for debugging purposes, simply freezes the HSCT module in the current state. This corresponds to the hard suspend of the whole module, that is both the transmitter and the receiver. After the end of the suspend state, the receive data will be most probably corrupted. Therefore, the whole interface must be reset, re initialized and reactivated.

35.7.3.5.2 HSCT Protocol Sleep Mode

Going into HSCT protocol sleep mode requires that the HSCT module has finished all transfer activities and that the HSSL transmit path or HSCT has no more frames available for sending and the last data of the current frame are send out. The user enables or disables sleep mode (SLEEPCTRL.**SLPEN**) during module initialization phase.

If the sleep mode is used, the application software has to configure the counter SLEEPCTRL.WKUPCNT to a value corresponding to the LVDS pad wake-up time defined in the data sheet.

35.7.3.5.3 Disable Request (Power-Off)

Going into power off requires that the HSCT module has finished all activities, which needs to be ensured by SW before going into power off mode. It is also required that the higher level HSSL protocol has finished the transfers. It is the software responsibility to change to low speed mode and disable the transmitter and the receiver according to the application, by writing the appropriate registers, because the speed mode change and transmitter and receiver disabling do not occur automatically. The HSCT high speed dividers need to be disabled (**INIT**.RXHD = 0 and **INIT**.TXHD = 0), before the physical layer is set to Power-Off mode. Power-Off mode of Physical

Layer can be achieved by disabling the 5 phases provided by Peripheral PLL with **CONFIGPHY**.PON. The clocks of the other parts of the HSCT module can be switched off by the **CLC**.DISR request bit or by the chip system sleep mode control (see **CLC**.EDIS).

35.7.3.6 References

[2] IEEE Std 1596.3-1996 - IEEE Standard for Low-Voltage Differential Signals (LVDS) for Scalable Coherent Interface (SCI)

[3] C40FLA LVDS Development Spec from Infineon.

35.7.4 Registers

35.7.4.1 Registers Definition

This section describes the kernel registers of the HSCT module and the System related register in single kernel configuration. All HSCT kernel register will be referenced in other parts of the TC3xx User's Manual by the module name prefix "HSCT_".

HSCT Kernel Register Overview

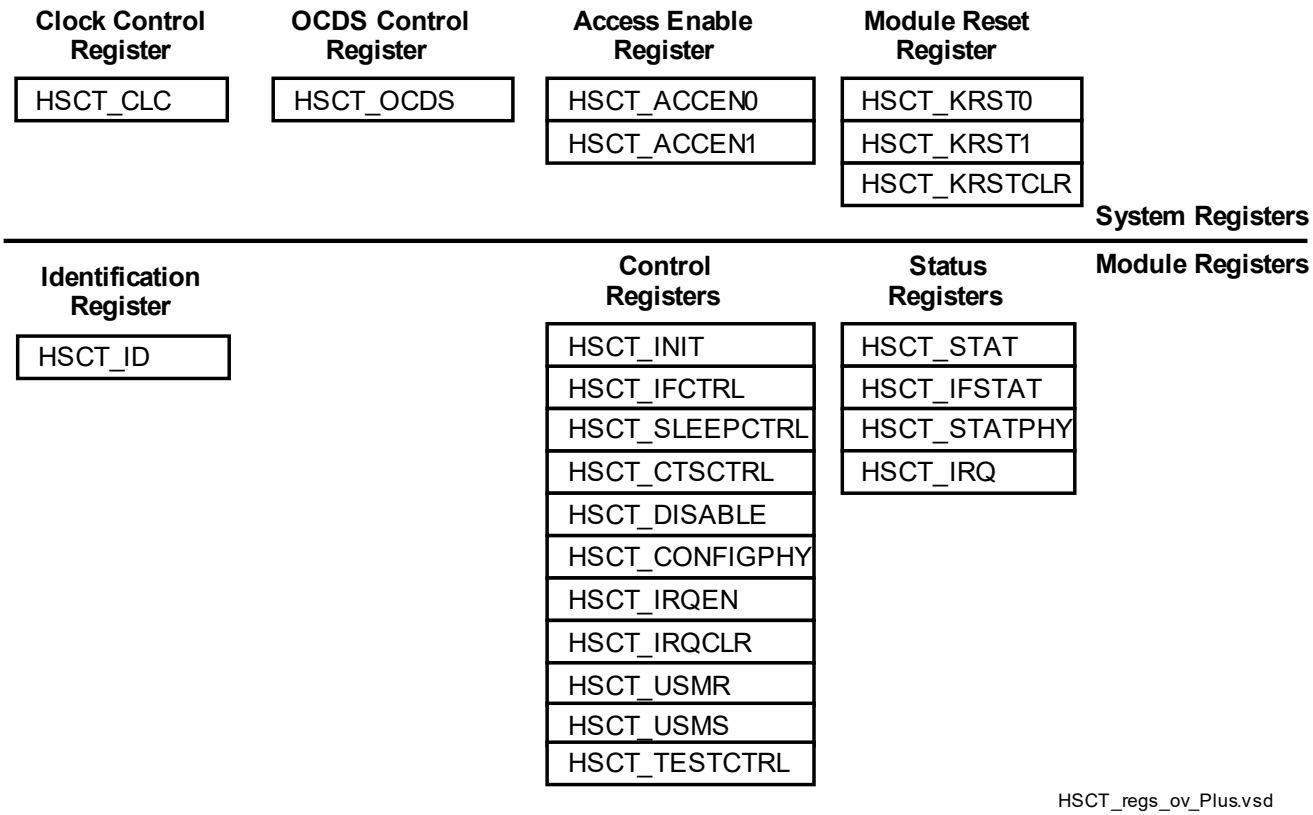


Figure 414 HSCT Register Overview

The complete and detailed address map of the HSCT modules is described in the next chapter.

35.7.4.1.1 HSCT Kernel Register Definitions

Table 320 Register Overview - HSCT (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	00000_H	U,SV	SV,E,P	Application Reset	134
ID	Module Identification Register	00008_H	U,SV	BE	Application Reset	116
INIT	Initialization Register	00010_H	U,SV	U,SV,P	Application Reset	117
IFCTRL	Interface Control Register	00014_H	U,SV	U,SV,P	Application Reset	118
SLEEPCTRL	Sleep Control Register	00018_H	U,SV	U,SV,P	Application Reset	120
CTSCTRL	Clear To Send Control Register	$0001C_H$	U,SV	U,SV,P	Application Reset	121
DISABLE	Transmission Disable Register	00020_H	U,SV	U,SV,P	Application Reset	122
STAT	Status Register	00024_H	U,SV	U,SV,P	Application Reset	123
IFSTAT	Interface Status Register	00028_H	U,SV	U,SV,P	Application Reset	124
CONFIGPHY	Configuration Physical Layer Register	00030_H	U,SV	U,SV,P	Application Reset	125
STATPHY	STATPHY	00034_H	U,SV	U,SV,P	Application Reset	125
IRQ	Interrupt register	00040_H	U,SV	U,SV,P	Application Reset	126
IRQEN	Interrupt Enable Register	00044_H	U,SV	U,SV,P	Application Reset	128
IRQCLR	Interrupt Clear Register	00048_H	U,SV	U,SV,P	Application Reset	130
USMR	Unsolicited Status Message Received	00050_H	U,SV	U,SV,P	Application Reset	130
USMS	Unsolicited Status Message Send	00054_H	U,SV	U,SV,P	Application Reset	131
TESTCTRL	Test Control Register	00060_H	U,SV	SV,P	Application Reset	131
OCS	OCDS Control and Status	$0FFE8_H$	U,SV	SV,P	Debug Reset	134
KRSTCLR	Reset Status Clear Register	$0FFEC_H$	U,SV	SV,E,P	Application Reset	139

Table 320 Register Overview - HSCT (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
KRST1	Reset Register 1	$0FFF0_H$	U,SV	SV,E,P	Application Reset	138
KRST0	Reset Register 0	$0FFF4_H$	U,SV	SV,E,P	Application Reset	137
ACCEN1	Access Enable Register 1	$0FFF8_H$	U,SV	SV,SE	Application Reset	137
ACCEN0	Access Enable Register 0	$0FFFC_H$	U,SV	SV,SE	Application Reset	136

Note: The absolute register address is calculated as follows: Module Base Address + Offset Address

A register is addressed word wise.

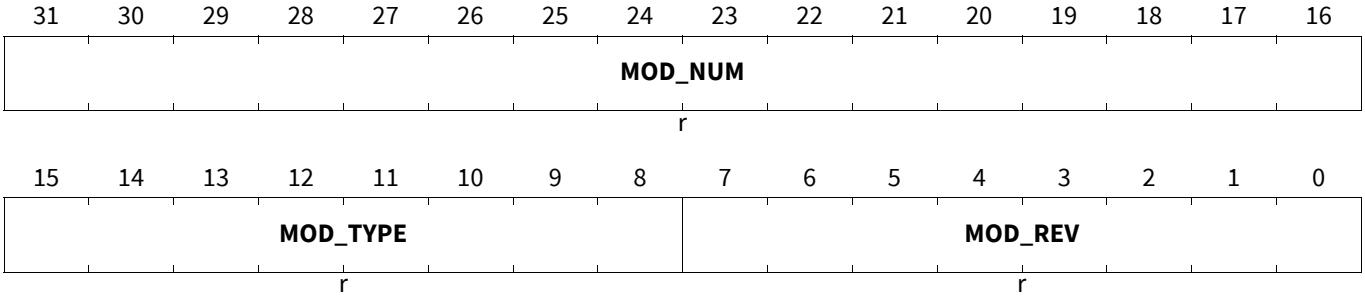
Note: Register bits marked reserved in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.

Module Identification Register

The Module Identification Register ID contains read-only information about the module version.

ID

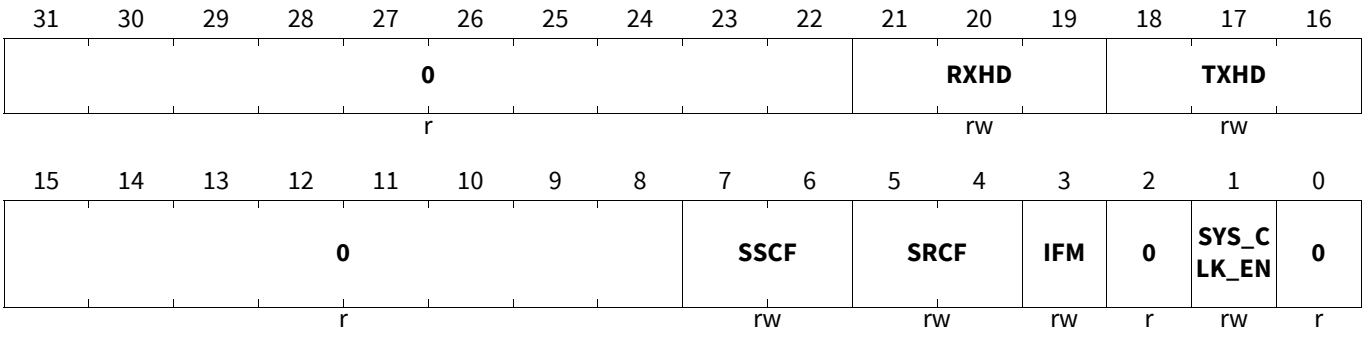
Module Identification Register (00008_H) Application Reset Value: $00B4\ C002_H$



Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision Number** 02_H The value of a module revision starts with 02_H (second revision).
MOD_TYPE	15:8	r	**Module Number Type** $C0_H$ 32 bit peripheral
MOD_NUM	31:16	r	**Module Number for module identification** $00B4_H$ Is the module identification number for the HSCT interface.

Initialization Register

INIT
Initialization Register **(00010_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SYS_CLK_EN	1	rw	**Enable HSCT SysClk in Master interface** SysClk enable activates the SysClk. This feature is only available in the Master interface. In Slave interface mode the register setting does not have an effect. 0_B Disable (default) 1_B Enabled
IFM	3	rw	**Select Interface Mode** Select the Interface Mode (Master IF or Slave IF). 0_B Master IF 1_B Slave IF
SRCF	5:4	rw	**Select Reference Clock Frequency Divider** Defines physical layer reference frequency (PHY_CLK) and respective input frequency divider. The configuration is valid for Low speed and Medium Speed mode. 00_B REFCLK 10MHz, LS 5MBaud, MS n.a. 01_B REFCLK 20MHz, LS 5MBaud, MS=20MBaud 10_B REFCLK 40MHz, LS 5MBaud, MS=20MBaud 11_B For future use
SSCF	7:6	rw	**Select SysClk Frequency Divider** For master interface defines SysClk pad output frequency. The allowed SysClk frequency is 10MHz or 20MHz. 00_B SYSCLK DIV 1/1 01_B SYSCLK DIV 1/2 10_B SYSCLK DIV 1/4 11_B For future use

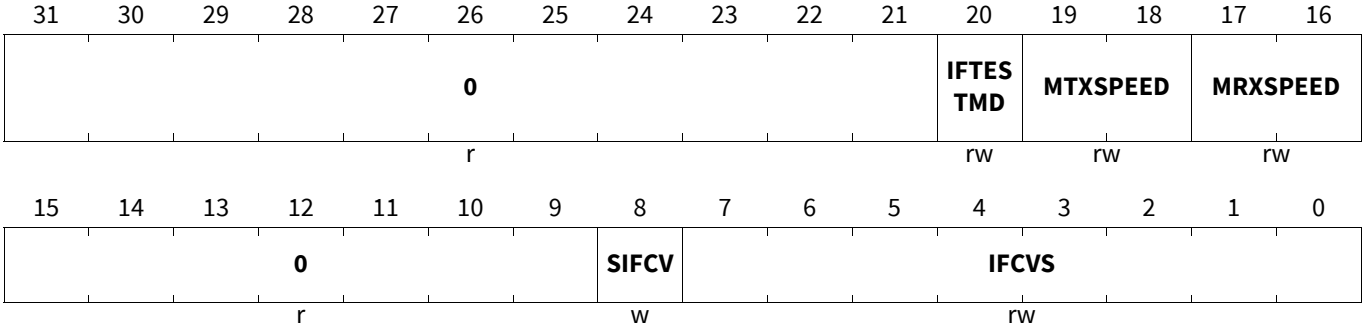
Field	Bits	Type	Description
TXHD	18:16	rw	**Transmit High Speed Divider.** The Transmit High Speed data rate can be reduced by dividing factors. The Transmit High Speed data rate is separated from the Receive High Speed data rate. 000_B Divider 1/1 001_B Divider 1/2 010_B Divider 1/4 011_B Divider 1/8 100_B Divider 1/16 101_B For future use. … 111_B For future use.
RXHD	21:19	rw	**Receive High Speed Divider.** The Receive High Speed data rate can be reduced by dividing factors. The Receive High Speed data rate is separated from the Transmit High Speed data rate. *Note: For future use configuration leads to no output clock delivered to the correlator.* 000_B Divider 1/1 001_B Divider 1/2 010_B Divider 1/4 011_B Divider 1/8 100_B Divider 1/16 101_B For future use. … 111_B For future use.
0	0, 2, 15:8, 31:22	r	**Reserved**

Interface Control Register

This register allows to configure the transfer Speed of the Master Interface and is also used to send the command frame to the Slave.

Note: The Master interface link speed configuration immediately takes place. Therefore the Master interface speed transition must not be changed with a Slave interface command send activity.

IFCTRL
Interface Control Register **(00014$_H$)** **Application Reset Value: 0000 0000$_H$**

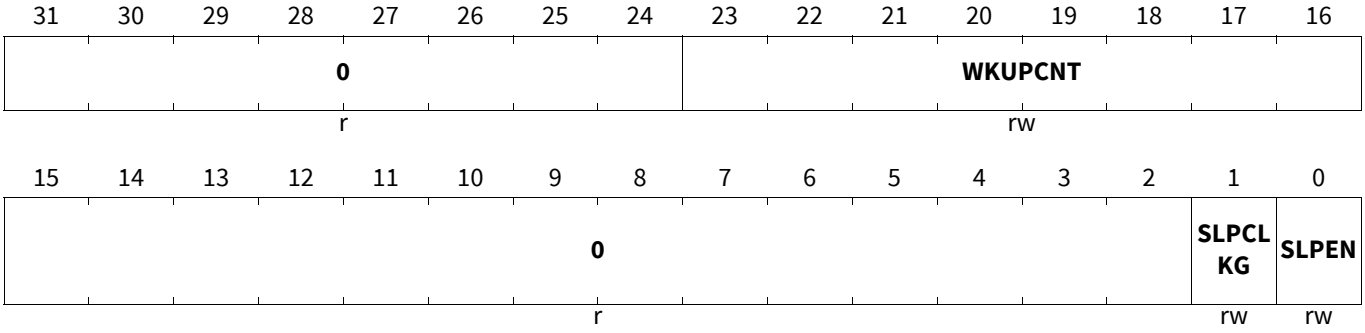


Field	Bits	Type	Description
IFCVS	7:0	rw	**Master Mode - Trigger for Interface Control Value to be send to Slave interface** See the table "Interface Control Payload Values". Master IF Mode: The value is taken as control frame value send as payload to the Slave IF. Slave IF Mode: The value is a new configuration of the Slave IF (not recommended flow!!!).
SIFCV	8	w	**Master Mode - Slave IF control frame trigger** Changing the interface configuration, software must guarantee not having transfers active on the interface. This register bit always reads back zero. 0_B The IFCVS field configured value has no effect (default). 1_B Writing a one to the register bit sends the control frame configured in register field IFCVS, if the interface is configured as Master. In Slave Mode the trigger has an effect and takes the IFCTRL.IFCVS value. In Slave mode this is not the recommended control flow. A frame based configuration shall be used instead. A control frame has higher priority then a register control. At a simultaneous occurrence of both configuration source the register configuration in Slave mode is lost.
MRXSPEED	17:16	rw	**Master Mode RX speed** *Note: Register setting only valid in interface Master mode.* 00_B Low Speed 01_B Medium Speed 10_B High Speed 11_B For future use

Field	Bits	Type	Description
MTXSPEED	19:18	rw	**Master Mode TX speed** *Note: Register setting only valid in interface Master mode.* 00_B Low Speed 01_B for future use 10_B High Speed 11_B For future use
IFTESTMD	20	rw	**Interface TX Test Mode** 0_B Test Mode disabled 1_B Test Mode enabled - send out 101010101… test pattern continuously.
0	15:9, 31:21	r	**Reserved**

Sleep Control Register

SLEEPCTRL
Sleep Control Register **(00018_H)** **Application Reset Value: $0020\ 0000_H$**



Field	Bits	Type	Description
SLPEN	0	rw	**Sleep mode enabled** Sleep mode is enabled and performed after receiving a 1 at the end of a received frame or in transmission direction, if no more data to be send. 0_B Sleep mode disabled. 1_B Sleep mode enabled.
SLPCLKG	1	rw	**Clock Gating in Sleep Mode** In sleep mode the clock for HSCT (framer, deframer) can be gated in order to minimize power consumption. *Note: Clock gating: Receiving path and transmitting path is separated.* 0_B Clock gating in sleep mode disabled. 1_B Clock gating in sleep mode enabled.

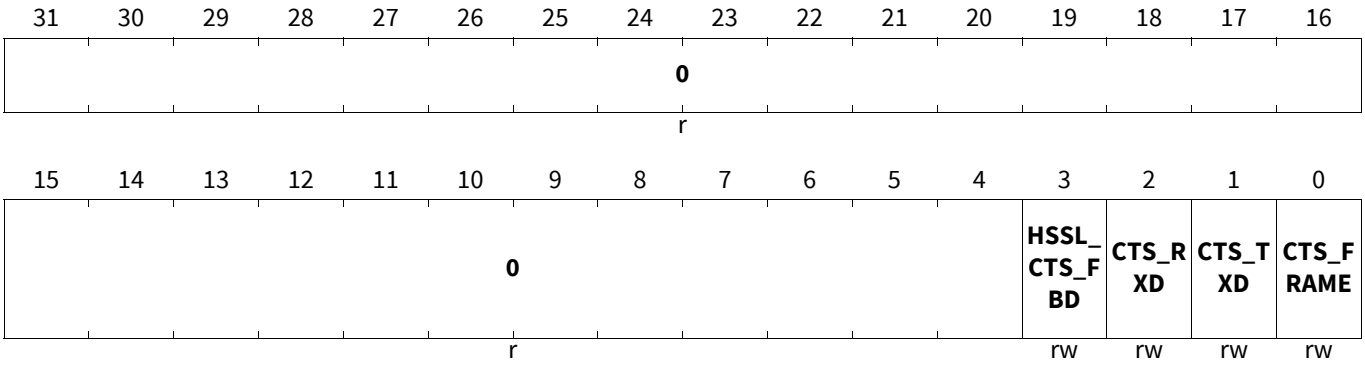
Field	Bits	Type	Description
WKUPCNT	23:16	rw	**Counter Value for Determining the Wake-up Time of the LVDS Line Driver** This counter value corresponds to wake-up time the LVDS requires from sleep to wake-up. Counter is clocked by SRI clock.
0	15:2, 31:24	r	**Reserved**

Clear To Send Control Register

The Clear To Send function is not available in multi Slave scenario. This function is an automatic HW controlled function and in multi Slave mode not easy to control. Therefore it is removed in this mode.

CTSCTRL

Clear To Send Control Register **($0001C_H$)** **Application Reset Value: $0000\ 0001_H$**



Field	Bits	Type	Description
CTS_FRAME	0	rw	**Transmit CTS Frame Generation** Dedicated CTS frames are generated after the receive data path is not able to accept more data. The situation is indicated by the CTS header bit in case there is currently no data to be transferred. 0_B Generation of dedicated CTS frames disabled. 1_B Generation of dedicated CTS frames enabled.
CTS_TXD	1	rw	**Disable TX CTS signaling** If this bit is set to 1, CTS signaling is not performed at the interface and the status remains at the clear to send for every frame send. 0_B Enable CTS signaling (default). 1_B Disable CTS signaling.
CTS_RXD	2	rw	**Disable RX CTS detection** If this bit is set to 1, CTS detection is not performed at the receiver and the status remains internally at clear to send for every frame received. 0_B Enable CTS detection (default). 1_B Disable CTS detection.
HSSL_CTS_FBD	3	rw	**Disable HSSL interface CTS Frame Blocking** If this bit is set to 1, CTS signaling is not performed at the interface and the status remains at the clear to send for every frame send. 0_B Enable CTS frame blocking (default). 1_B Disable CTS frame blocking.

Field	Bits	Type	Description
0	31:4	r	**Reserved**

Transmission Disable Register

DISABLE
Transmission Disable Register (00020$_H$) Application Reset Value: 0000 0003$_H$

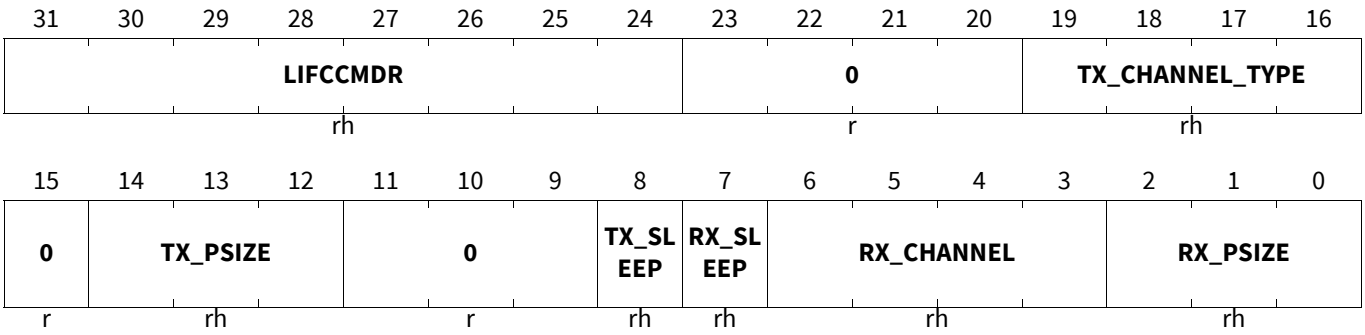
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													**RX_HEPD**	**RX_DIS**	**TX_DIS**
r													rw	rw	rw

Field	Bits	Type	Description
TX_DIS	0	rw	**Disable HSCT Transmit path in Master interface** Disable the transmit path of the HSCT interface. If this bit is set to 1 - no transfer can be initiated and the LVDS driver is disabled. 0_B Enable 1_B Disable
RX_DIS	1	rw	**Disable HSCT Receive path in Master interface** Disable the receive line path of the HSCT interface. If this bit is set to 1 - no transfer from the other side can be received and the Master RX path is in a low power state. This feature is only available in the Master interface. Slave interface RX path can not be disabled and a write to the register has no effect. 0_B Enable 1_B Disable
RX_HEPD	2	rw	**Disable RX Header Error Discard Payload data.** Instead of discarding the Payload data at a header error the payload data is passed to the higher layer (HSSL). Only channel data to HSSL is affected. This function is available in Master and in Slave mode. 0_B Header error received data is discarded 1_B Header error received data is passed to the higher hardware layer.
0	31:3	r	**Reserved**

Status Register

STAT
Status Register **(00024_H)** **Application Reset Value: $0000\ 0000_H$**

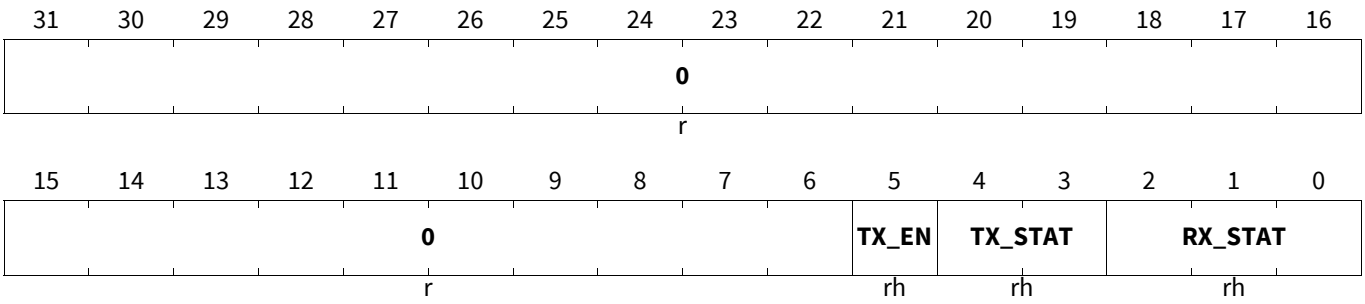


Field	Bits	Type	Description
RX_PSIZE	2:0	rh	**RX (Receiving) Payload Size** Contains the payload size of the last received frame.
RX_CHANNEL	6:3	rh	**RX (Receiving) Logical Channel Type** Contains the logical channel type of the last received frame. See Table "Logical Channel Type Coding".
RX_SLEEP	7	rh	**RX (Receiving) Sleep Mode Status** 0_B HSCT is in receive direction active 1_B HSCT is in receive direction in sleep mode
TX_SLEEP	8	rh	**TX (Transmission) Sleep Mode Status** 0_B HSCT is in transmit direction active 1_B HSCT is in transmit direction in sleep mode
TX_PSIZE	14:12	rh	**Transmission Payload Size** Coding of the logical channel type is according to the Table: Payload Size Coding. This value was used in the logical channel type field in the header for the actual transfer in transmit direction.
TX_CHANNEL_TYPE	19:16	rh	**Transmission Logical Channel Type** Coding of the logical channel type is according to the Table: Logical Channel Type Coding. This value was used in the logical channel type field in the header for the actual transfer in transmit direction.
LIFCCMDR	31:24	rh	**Last Interface Control Command Received** The bit value reflects the last control command received. The bit is active in Slave interface mode only. In Master mode it reflects logic 0 always. (See **Table 318**)
0	11:9, 15, 23:20	r	**Reserved**

Interface Status Register

Clock test mode and Loopback mode are only available in single Slave scenario. In multi Slave scenario are not available.

IFSTAT
Interface Status Register **(00028_H)** **Application Reset Value: $0000\ 0000_H$**

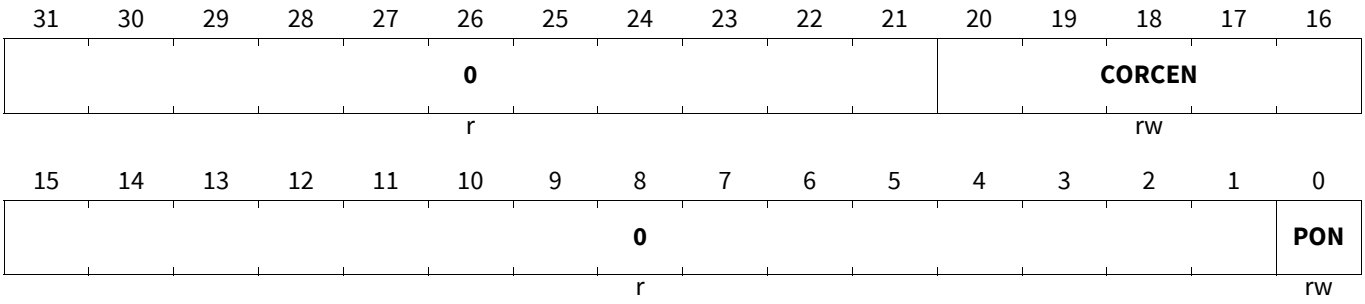


Field	Bits	Type	Description
RX_STAT	2:0	rh	**HSCT Slave interface Status for RX link** *Note: Slave interface transmitter only* 000_B Interface is disabled in RX link direction. 001_B Interface runs at low speed on RX link. 010_B Interface runs at medium speed on RX link. 011_B Interface runs at high speed on RX link direction. 100_B Reserved 101_B Clock test mode and low speed on RX ink 110_B Clock test mode and medium speed on RX link 111_B Clock test mode and high speed on RX link.
TX_STAT	4:3	rh	**HSCT Slave interface Status for TX link** *Note: Slave interface receiver only* 00_B Interface runs at low speed on TX link. 01_B Interface runs at high speed on TX link. 10_B Loopback mode low speed. 11_B Loopback mode high speed.
TX_EN	5	rh	**HSCT LVDS Slave interface TX enable** *Note: Slave interface only* 0_B LVDS TX disabled. 1_B LVDS TX enabled.
0	31:6	r	**Reserved**

Configuration Physical Layer Register

CONFIGPHY

Configuration Physical Layer Register (00030$_H$) Application Reset Value: 001F 0000$_H$

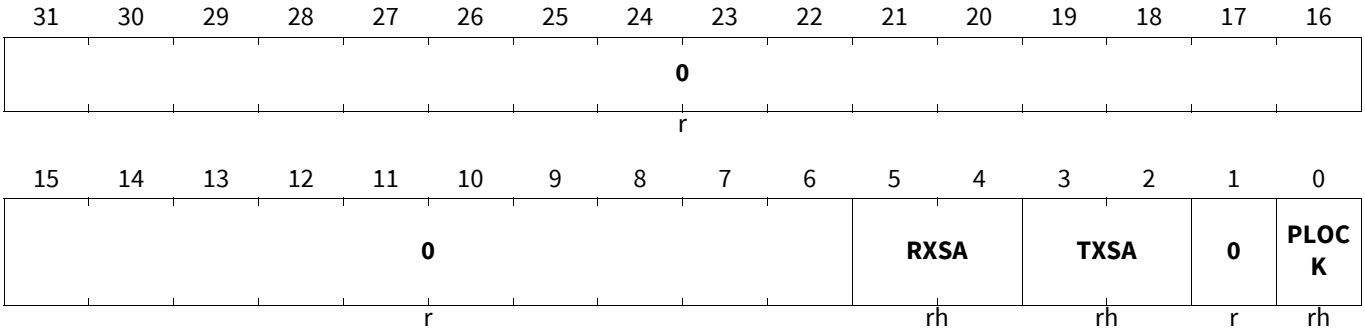


Field	Bits	Type	Description
PON	0	rw	**Physical Layer Power On.**
CORCEN	20:16	rw	**Correlator phase enable - allows to enable/disable each of the 5 Phase outputs separately.**
0	15:1, 31:21	r	**Reserved**

STATPHY

STATPHY

STATPHY (00034$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
PLOCK	0	rh	**PLL locked** 0_B Peripheral PLL out of lock (default) 1_B Peripheral PLL locked
TXSA	3:2	rh	**Transmitter speed** 00_B Transmitter in Low speed 01_B Transmitter in Medium speed 10_B Transmitter in High speed 11_B reserved

Field	Bits	Type	Description
RXSA	5:4	rh	**Receiver speed** 00_B Receiver in Low speed 01_B Receiver in Medium speed 10_B Receiver in High speed 11_B reserved
0	1, 31:6	r	**Reserved**

Interrupt register

Interrupt status register. Read only and updated by HW. The bit's remain active until cleared. The clear pulse is coming from the IRQCLR register. Only an interrupt source not being already active in the interrupt status register generates an interrupt pulse to the SoC interrupt controller.

IRQ

Interrupt register **(00040_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**MSCE**	**SFU**	**SFO**	**TXTE**	**PAR**	**USM**	**PLER**	**USMSF**	**SMER**	**IFCFS**	**CER**	**PYER**	**HER**	**0**
r		rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	r

Field	Bits	Type	Description
HER	1	rh	**Header error detected** Not supported size: • Received command at Slave interface 8-bit size only - other command sizes generate an error. • Received command ping answer at Master interface 32-bit size only - other command sizes generate an error • Unsolicited data 32-bit only • Logical data channel size 8-bit Not supported logical channel type: • 0b1xxx • 0010 (Slave interface control and Slave interface read) 0_B Header OK 1_B Header issue detected.
PYER	2	rh	**Payload error detected** Payload does not fit the header size 0_B Payload size OK 1_B Received payload size wrong

Field	Bits	Type	Description
CER	3	rh	**HSCT command error** Control command not valid or control payload size bigger than 8-bit. Single Slave specific commands received in Multi Slave Mode do not trigger CER error, but only MSCE error. 0_B No command error 1_B HSCT command error
IFCFS	4	rh	**HSCT interface control frame send** The scheduled interface control command is send. 0_B No interface control command send 1_B Interface control command send.
SMER	5	rh	**Speed Mode Switch Error (Master Mode only)** Speed mode change did not work. Received PING payload not valid. 0_B Interfaces runs at defined speed 1_B Ping payload error
USMSF	6	rh	**Unsolicited message frame send finished** Interrupt is indicated after the unsolicited message send is finished. 0_B No unsolicited message send. 1_B Unsolicited message send finished.
PLER	7	rh	**PLL lost lock error** After the PLL locked, the PLL may loose lock, which is reflected by the error 0_B PLL lock 1_B PLL lock loss
USM	8	rh	**Unsolicited Message Received** Unsolicited message received indication. Unsolicited message indicates a system event to the other interface side. 0_B no unsolicited message available 1_B unsolicited message available
PAR	9	rh	**PING Answer Received** The received message was identified as PING. 0_B No PING message available 1_B PING message received
TXTE	10	rh	**TX transfer error occurred on a disabled TX channel.** A disabled TX triggers an error interrupt, if: • TX disabled on a pending or active data transfer. • TX CTS configuration change on a active CTS frame. 0_B No error situation occurred 1_B Error situation occurred

Field	Bits	Type	Description
SFO	11	rh	**Synchronization FIFO overflow (in RX direction)** Physical layer to Controller data synchronization FIFO in RX transfer direction hit an overflow situation. *Note: This interrupt is an indication about a to slow SRI clock compared to Physical layer clock, which results in a overflow situation. (Minimum SRI frequency 40 MHz.)* 0_B RX synchronization is running well. 1_B RX FIFO overflow
SFU	12	rh	**Synchronization FIFO underflow (in TX direction)** Controller to Physical layer data synchronization FIFO in TX transfer direction hit an underflow situation. *Note: This interrupt is an indication about a to slow SRI clock compared to Physical layer clock, which results in a underflow situation. (Minimum SRI frequency 40 MHz.)* 0_B TX synchronization is running well. 1_B TX FIFO underflow
MSCE	13	rh	**Multi Slave scenario Command Error** This interrupt indicates a control command which is not allowed in multi Slave scenario. In Master and Slave mode a not allowed command results to an error. 0_B No multi Slave scenario command error. 1_B Multi Slave scenario command error (not allowed command used)
0	0, 31:14	r	**Reserved**

Interrupt Enable Register

Interrupt enable register. An enabled register generates a pulsed interrupt. On a disabled interrupt the interrupt pulse does not come through on the interrupt line.

IRQEN
Interrupt Enable Register **(00044_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		MSCEEN	SFUEN	SFOEN	TXTEEN	PAREN	USMEN	PLEREN	USMSFEN	SMEREN	IFCFSEN	CEREN	PYEREN	HEREN	0
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r

Field	Bits	Type	Description
HEREN	1	rw	**Header error detected interrupt enable** 0_B Interrupt disabled 1_B Interrupt enabled
PYEREN	2	rw	**Payload error detected interrupt enable** 0_B Interrupt disabled 1_B Interrupt enabled
CEREN	3	rw	**HSCT command error interrupt enable** 0_B Interrupt disabled 1_B Interrupt enabled
IFCFSEN	4	rw	**HSCT interface control command send enable** 0_B Interrupt disabled 1_B Interrupt enabled
SMEREN	5	rw	**Speed Mode Switch Error interrupt enable** 0_B Interrupt disabled 1_B Interrupt enabled
USMSFEN	6	rw	**Unsolicited message frame send finished** 0_B Interrupt disabled 1_B Interrupt enabled
PLEREN	7	rw	**PLL lost lock error interrupt enable** 0_B Interrupt disabled 1_B Interrupt enabled
USMEN	8	rw	**Unsolicited Message received enable** 0_B Interrupt disabled 1_B Interrupt enabled
PAREN	9	rw	**PING Answer Received enable** 0_B Interrupt disabled 1_B Interrupt enabled
TXTEEN	10	rw	**TX disable error interrupt enable** 0_B Interrupt disabled 1_B Interrupt enabled
SFOEN	11	rw	**Synchronization FIFO overflow (in RX direction) interrupt enable** 0_B Interrupt disabled 1_B Interrupt enabled
SFUEN	12	rw	**Synchronization FIFO underflow (in TX direction) interrupt enable** 0_B Interrupt disabled 1_B Interrupt enabled
MSCEEN	13	rw	**Multi Slave scenario Command Error interrupt enable** 0_B Interrupt disabled 1_B Interrupt enabled
0	0, 31:14	r	**Reserved**

Interrupt Clear Register

This register is the Interrupt clear register. By writing a logic 1 to the register the interrupt in the IRQ register is cleared by a clear pulse. This means the interrupt clear methodology is write one to clear. Reading from this register presents the logic value 0 to software.

IRQCLR
Interrupt Clear Register **(00048_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		MSCELR	SFUCLR	SFOCLR	TXTECLR	PARCLR	USMCLR	PLERCLR	USMSFCLR	SMERCLR	IFCFSCLR	CERCLR	PYERCLR	HERCLR	0
r		w	w	w	w	w	w	w	w	w	w	w	w	w	r

Field	Bits	Type	Description
HERCLR	1	w	**Header error detected interrupt clear**
PYERCLR	2	w	**Payload error detected interrupt clear**
CERCLR	3	w	**HSCT command error interrupt clear**
IFCFSCLR	4	w	**HSCT interface control command send interrupt clear**
SMERCLR	5	w	**Speed Mode Switch Error interrupt clear**
USMSFCLR	6	w	**Unsolicited message frame send finished interrupt clear**
PLERCLR	7	w	**PLL lost lock error interrupt clear**
USMCLR	8	w	**Unsolicited Message received clear**
PARCLR	9	w	**PING Answer received clear**
TXTECLR	10	w	**TX disable error interrupt clear**
SFOCLR	11	w	**Synchronization FIFO overflow (in RX direction) interrupt clear**
SFUCLR	12	w	**Synchronization FIFO underflow (in TX direction) interrupt clear**
MSCELR	13	w	**Multi Slave scenario Command Error interrupt clear**
0	0, 31:14	r	**Reserved**

Unsolicited Status Message Received

The Unsolicited Status message register is available to capture unsolicited messages received with the logical channel command encoding in the header of the frame (Logical Channel Type Hex 1). This function is not available in multi Slave scenario.

USMR

Unsolicited Status Message Received **(00050$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
USMR															
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USMR															
rh															

Field	Bits	Type	Description
USMR	31:0	rh	**Unsolicited status message received** The register contains the last received unsolicited status message.

Unsolicited Status Message Send

A write to the register activates the interface to send out an unsolicited status message to the other interface side. The Logical Channel Type reflected in the header is Hex 1 and the Payload is 32-bit always. An unsolicited frame has a higher priority then a Frame based on a data channel. In between frames the USMS gets higher priority then the data transfer requested by the HSSL. This function is not available in multi Slave scenario.

USMS

Unsolicited Status Message Send **(00054$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
USMS															
rw															

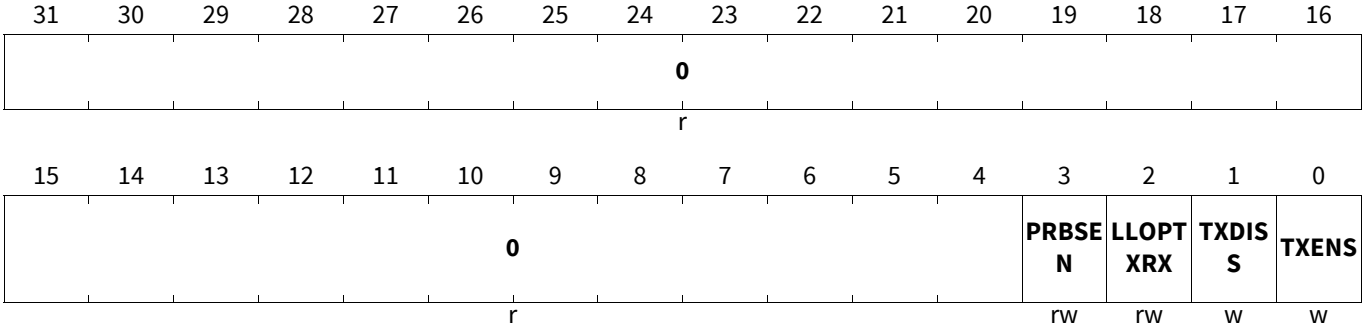
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USMS															
rw															

Field	Bits	Type	Description
USMS	31:0	rw	**Unsolicited status message send** Writing to the register triggers an unsolicited status message to be send to the other interface side.

Test Control Register

Note: The TESTCTRL.LLOPTXRX bit is not working with IFCTRL.IFTESTMD. Data is not transported to HSCT controller part (link layer).

TESTCTRL
Test Control Register **(00060$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TXENS	0	w	**Enable Slave TX path (Slave interface mode only)** *Note: This function should be only used during interface testing the mode and SW development. In functional mode the Slave interface should only be controlled via transfer commands received from Master interface. (This trigger register reads back 0 always.) If TXENS and TXDISS are written one, TXDISS has higher priority. The status is visible in IFSTAT.TX_EN.* 0_B Writing logic 0 has no effect 1_B Triggers the Slave interface TX enable.
TXDISS	1	w	**Disable Slave TX path (Slave Interface mode only)** *Note: This function should be only used during interface testing the mode and SW development. In functional mode the Slave interface should only be controlled via transfer commands received from Master interface. (This trigger register reads back 0 always.) If TXENS and TXDISS are written one, TXDISS has higher priority. The status is visible in IFSTAT.TX_EN.* 0_B Writing logic 0 has no effect 1_B Triggers the Slave interface TX disable.
LLOPTXRX	2	rw	**LVDS loop back TX to RX enable** Transmit data at LVDS is directly looped back to the LVDS RX. Data transfer speed is defined by the TX speed configuration. (The data path in the SoC is using the complete Transmit path through all functional layers and is looped back at LVDS from TX to RX. Also at RX data path all SoC data layers are active.) *Note: Requires same speed configuration for RX- and TX-link before enabled.* 0_B Disabled LVDS TX to RX data Loop back test mode 1_B Enabled TX to RX data loop back test mode.

Field	Bits	Type	Description
PRBSEN	3	rw	**PRBS Pattern enable** Enable of the PRBSEN bit allows a continuos PRBS stream with the configured transfer speed Baud rate. This feature is available to measure ISI during the time other SoC functions are running in an applicative mode. This feature is for measurement purpose only. 0_B Disabled PRBS pattern generation on TX 1_B Enabled PRBS pattern generation on TX.
0	31:4	r	**Reserved**

35.7.4.1.2 BPI_FPI Module Registers (Single Kernel Configuration)

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the HSCT module. Where a module kernel is connected to the CLC clock control interface, CLC register controls the module clock, SoC system sleep mode and disable mode for the module.

CLC
Clock Control Register **(00000_H)** **Application Reset Value: $0000\ 0003_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	r	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control (clock and local access) of the module. This bit switches the clock off immediately, therefore the software has to take care that the HSSL and HSCT communication has been terminated properly before setting this bit.
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the module and is set after the requested disable is active.
EDIS	3	rw	**Chip System Sleep Mode Control** The EDIS bit in the CLC register controls whether or not a module is stopped during Chip System initiated Sleep Mode. If EDIS is 0, a Sleep Mode request can be recognized by the module and, when received, its clock is immediately shut off. Therefore the software has to take care that the HSSL and HSCT communication has been terminated properly before activating the sleep mode request. If EDIS is set to 1, a Sleep Mode request is disregarded by the module and the module continues its operation. Note: This chip system sleep mode has nothing to do with the HSCT protocol sleep mode.
0	2, 31:4	r	**Reserved** Read as 0; should be written with 0.

OCDS Control and Status

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The OCS register can only be written when the OCDS is enabled.

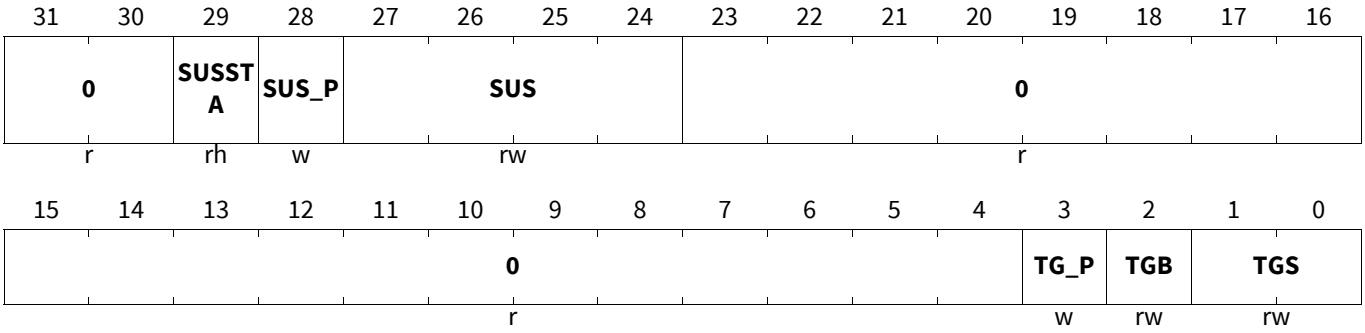
If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32-bit always.

The OCS register includes the module related control bits for the OCDS Trigger Bus (OTGB).

The register can only be written and the OCS control register bits are only effective while the OCDS is enabled (OCDS enable = ´1´). While OCDS is not enabled, OCS reset values/modes are effective.

OCS

OCDS Control and Status (0FFE8$_H$) Debug Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TGS	1:0	rw	**Trigger Set for OTGB0/1** Others reserved (no Trigger Set selected) 00_B No Trigger Set output 01_B TS16_HSCT
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1
TG_P	3	w	**TGS, TGB Write Protection** TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) Others reserved *Note: After a Hard suspend the HSCT and higher layer protocol module must be reset. A a new initialization sequence is required afterwards* 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately (read from registers still possible).
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.

Field	Bits	Type	Description
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:4, 31:30	r	**Reserved** Read as 0; must be written with 0.

The HSCT has one OCDS Trigger set available which gives a maximum of 16 triggers. The next table describes the available triggers.

Table 321 TS16_HSCT OCDS Trigger Set

Bits	Name	Description
0	HRXRQ	HSSL Interface RX transfer direction request
1	HRXCTS	HSSL Interface RX transfer direction CTS
2	HTXCTS	HSSL Interface TX transfer direction CTS
3	HTXRQ	HSSL Interface TX transfer direction request
4	TXF	First Byte transfer of transfer frame in TX transfer direction at Controller to Physical Layer interface
5	TXL	Last Byte transfer of transfer frame tin TX transfer direction at Controller to Physical Layer interface
6	TXW	Transmit direction wake-up at controller to physical layer interface
7	TXS	Transmit direction sleep indication at controller to physical layer interface
8	RXFS	Receive direction frame start at physical layer to controller interface.
9	DFSMI	Deframer state machine not in idle.
10	IFCT	Slave interface frame command trigger.
[15:11]		Reserved (value is 0)

Access Enable Register 0

The Access Enable Register 0 controls write access for transactions with the on chip bus Master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID Master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B,…,EN31 -> TAG ID 011111B.

ACCEN0
Access Enable Register 0 **($0FFFC_H$)** **Application Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables read/write access to the protected resources for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

The Access Enable Register 1 controls write access for transactions with the on chip bus Master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID Master peripheral mapping). The BPI_FBI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B,…, EN31 -> TAG ID 111111B.

ACCEN1

Access Enable Register 1 **(0FFF8$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

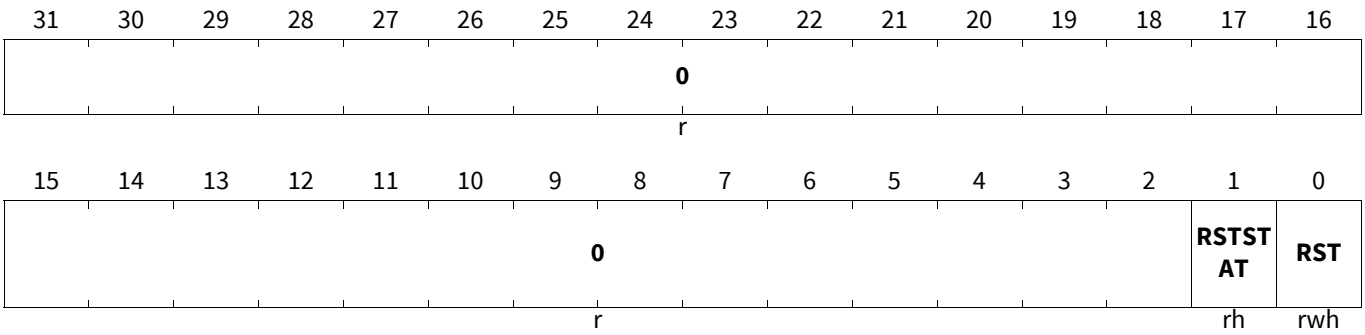
Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel and the HSCT physical layer. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset the module kernel and the physical layer it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLE.CLR register bit.

KRST0
Reset Register 0 **(0FFF4$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. *Note: It is strongly recommended to reset the HSCT and HSSL in sequence to avoid communication issues.* 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Reset Register 1 **(0FFF0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															RST
r															rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. *Note: It is strongly recommended to reset the HSCT and HSSL in sequence to avoid communication issues.* 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Reset Status Clear Register

The Kernel Reset Register Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR
Reset Status Clear Register **(0FFEC$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															CLR
r															w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT

Field	Bits	Type	Description
0	31:1	r	**Reserved** Read as 0; should be written with 0.

35.7.5 IO Interfaces

Table 322 List of HSCT Interface Signals

Interface Signals	I/O	Description
INT	out	**HSCT Service Request**
SYSCLK_IN	in	**Reference clock** Oscillator or System clock Directly connected to PLL_Wrapper
RXD	in	**Rx data** Received serial data stream from CMOS Rx pads
TXD	out	**Tx data** Serial data stream for transmitting data to LVDS Tx
SYSCLK_OUT	out	**sys clock output** Sysclk is used as a reference clock for the Slave device and transmitted via GPIO pad. Only active in Master mode
RXDN	in	**Rx data** Received serial data stream from LVDS Rx pads
RXDP	in	**Rx data** Received serial data stream from LVDS Rx pads
TXDP	out	**Tx data** Serial data stream for transmitting data to LVDS Tx
TXDN	out	**Tx data** Serial data stream for transmitting data to LVDS Tx
CLKA_REF_NXT	out	**Forward clka_ref_i to next HSCT PHY instance** Forward clka_ref_i to next HSCT PHY instance

35.7.6 Revision History

Table 323 Revision History

Reference	Change to Previous Version	Comment
V2.3.11		
Page 140	Previous versions removed from revision history.	none
V2.3.12		
Page 106	Table 13 “Processing Times for Interface Control” removed from section 1.3.3.4.	
Page 99	Typo fixed (ω instead of ?).	
Page 117	Formal change in bitfield description of Initialization Register (TXHD and RXHD), no functional change.	
V2.3.13		

Table 323 Revision History (cont'd)

Reference	Change to Previous Version	Comment
Page 125	Updated description for bit field PLOCK of register STATPHY.	
Page 113	Updated PLL in section Disable Request by 'Peripheral'.	
Page 93	Changed register reference for PCB termination not active from 'INIT' to 'LPCRx'.	
V2.3.14		
Page 111	Table and text added.	
Page 108	Changed text.	
V2.3.15		
Page 106	Added information regarding entering high speed mode.	

36 Asynchronous/Synchronous Interface (ASCLIN)

The main purpose of the ASCLIN module is to provide asynchronous serial communication with external devices using only data-in, data-out signals.

The focus of the module is set to fast and flexible communication: either fast point-to-point or master-to-many slaves communication using the LIN protocol.

Additionally, the module supports the synchronous SPI communication.

Figure 415 shows an overview of the ASCLIN module.

Note: The RX, TX, RTS, CTS, SCLKO, SLSO signal names are prefixed with "A" in order to achieve unique names on chip level, distinct from signal names of other communication modules.

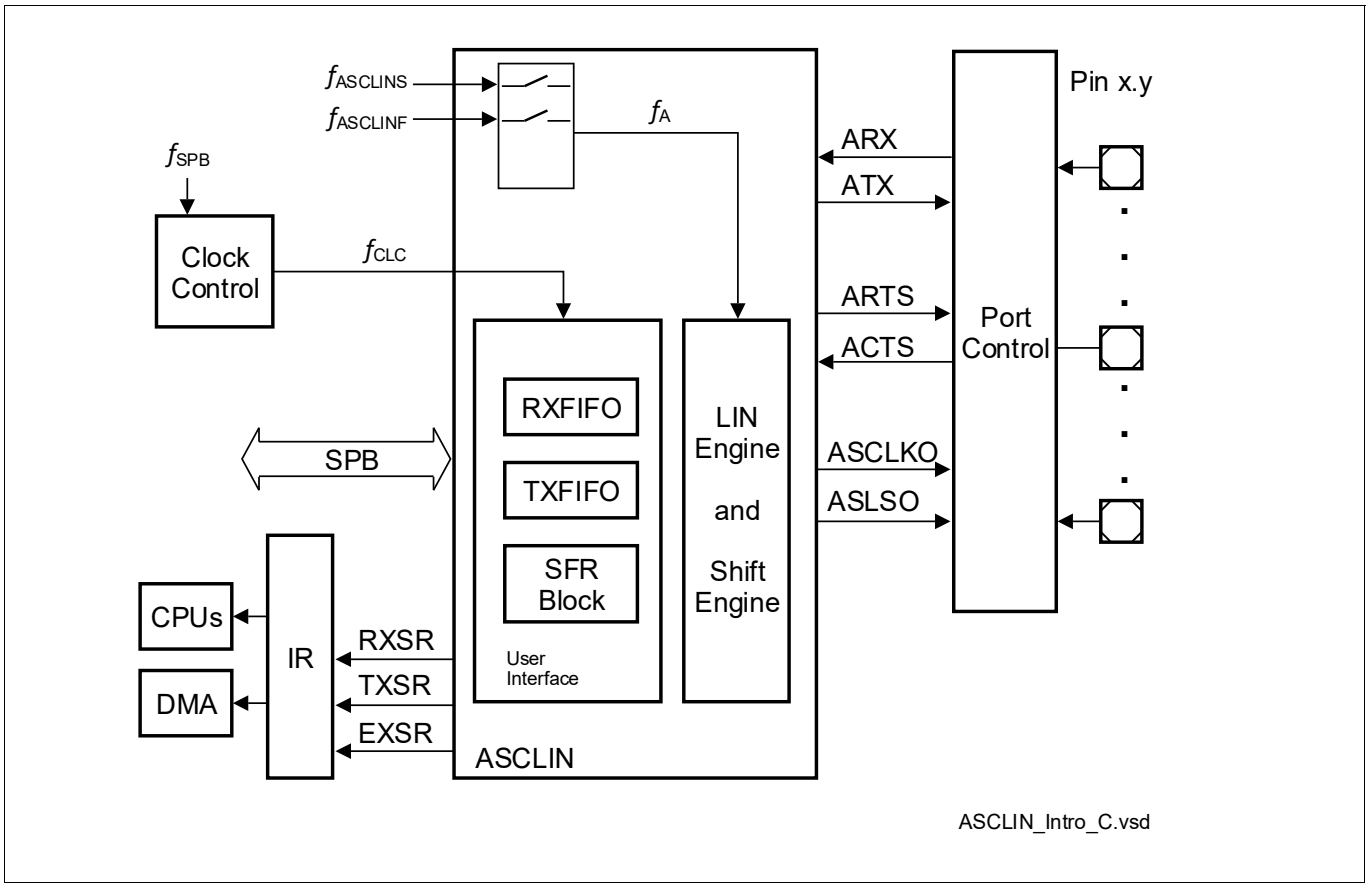


Figure 415 Block Diagram of the ASCLIN module.

36.1 Feature List

This section describes the features of the ASCLIN module.

General Features

- 16 bytes TxFIFO
- 16 bytes RxFIFO
- Pack / unpack capabilities of the Tx and Rx FIFO
- Interrupt generation
  - On a configurable transmit FIFO level
  - On a configurable receive FIFO level
  - On an error condition (frame, parity, overrun error)
  - On various module internal events (end of ASC/SPI frame, LIN events)
- Interrupt signals capable of triggering either a CPU or a DMA
- Programmable oversampling of 4 to 16 times per bit
- Programmable sampling point position
- Programmable digital glitch filter and median filter for incoming bit stream
- Shift direction LSB first for ASC and LIN mode, and programmable LSB/MSB first for SPI mode
- Internal loop-back mode

Standard ASC Features

- Full-duplex asynchronous operating mode
  - 7-bit, 8-bit or 9-bit (or up to 16-bit) data frames, LSB first
  - Parity-bit generation/checking
  - One or two stop bits
  - Max baud rate f_A / 16 (6.25 MBaud @ 100 MHz f_A module clock)
  - Min. baud rate f_A / 268 435 456 (0.37 Baud @ 100 MHz f_A module clock)
- Optional RTS / CTS handshaking

Extended ASC Features

- Programmable oversampling of 4 to 16 times per bit
  - module capability of up to 4 times higher baud rates (f_A / 4) then the standard ASC
  - system considerations like pad type, incoming signal quality and accumulated PLL jitter can lead to constraints regarding the usable oversampling ratios (for example f_A / 8 = 200MHz / 8 = 25 MBaud)
- Programmable sampling point position

LIN Features

- Support of
  - LIN version 1.3
  - LIN version 2.0
  - LIN version 2.1
  - LIN version 2.2 and
  - J2602

- Break detection
- Break injection
- Sync field generation
- Auto baud detection based on Sync Field measurement
- Optional Collision detection, required for LIN version 2.1
- LIN Watchdogs
  - Header time-out
  - Frame or Response time-out
- Stuck at zero/one monitoring
- Bus idle time monitoring
- Wake-Up
- Minimum CPU load in master mode
  - Single interrupt indicating the end of the frame
- Minimum CPU load in slave mode
  - Single interrupt at the end of the header reception
  - Single interrupt at the end of the response or end of frame
- Standard operation with one interrupt per transmitted or received byte supported

SPI Features

- SPI master modes (slave mode not supported):
  - Four-wire or three-wire (with / without slave select output signal)
- Up to 16-bit data width
- Full-duplex and half-duplex
  - Min. baud rate f_A / 268 435 456 MBaud (= 0.37 Baud @ 100 MHz f_A module clock)
  - Max. baud rate f_A / 4 MBaud (= 25 MBaud @ 100 MHz f_A module clock)
- Programmable leading and trailing delays

36.2 Overview

This section shows the essential sub blocks of the module when used as standard ASC, as LIN or as SPI master.

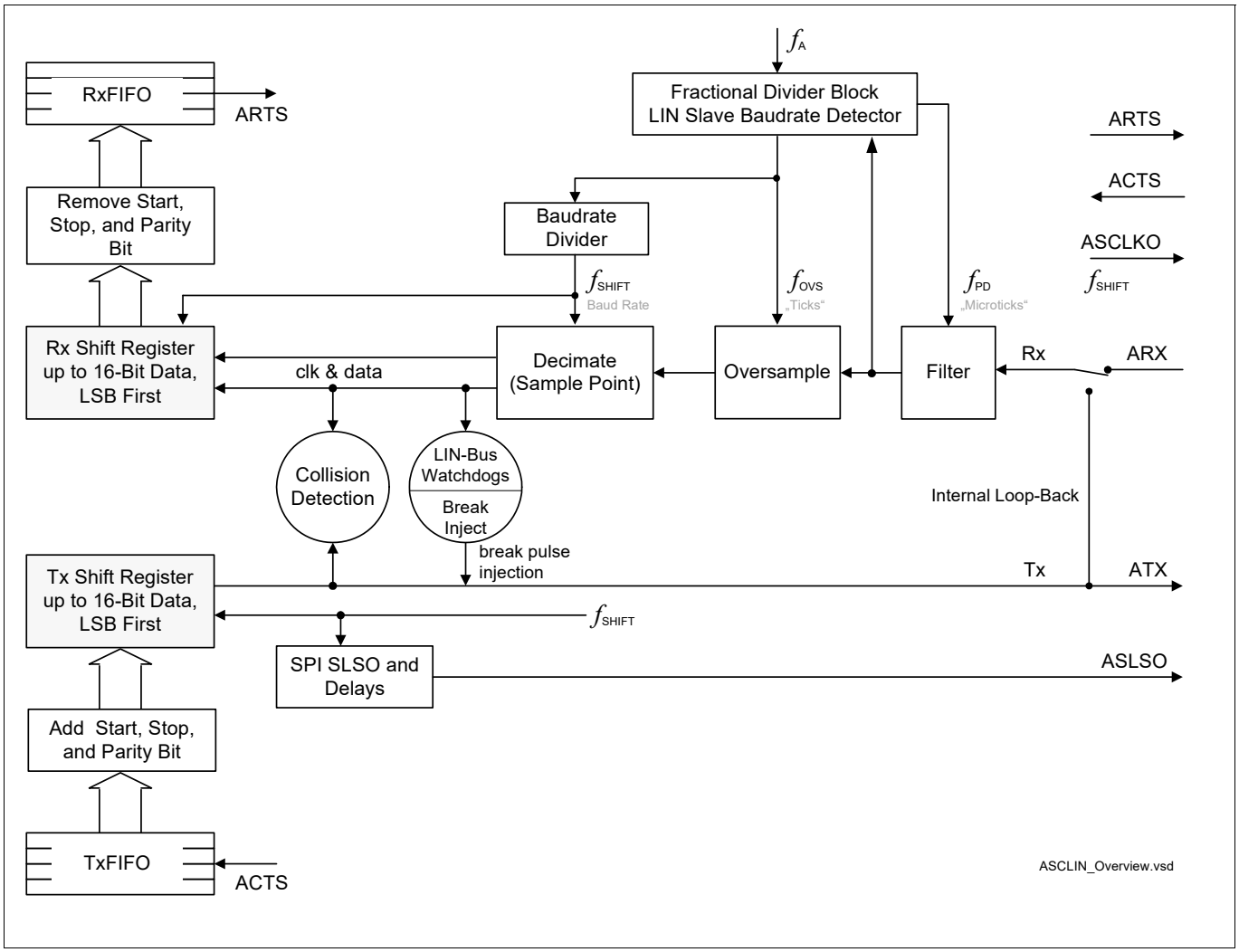


Figure 416 Architecture Overview

Notes

1. *Collision detection is mandatory only in the LIN specification V2.1.*
2. *In LIN mode, both ARX and ATX signal must be connected to the LIN bus in order ASCLIN module to work properly.*

36.3 Functional Description

36.3.1 External Signals

The ASCLIN module provides the following external signals:

- Serial clock output ASCLK
- Receive data input ARX (Master Receive MR input in SPI mode)
- Transmit data output ATX (Master Transmit MT output in SPI mode)
- Slave select signal output ASLSO
- Request to send handshake output ARTS
- Clear to send handshake input ACTS

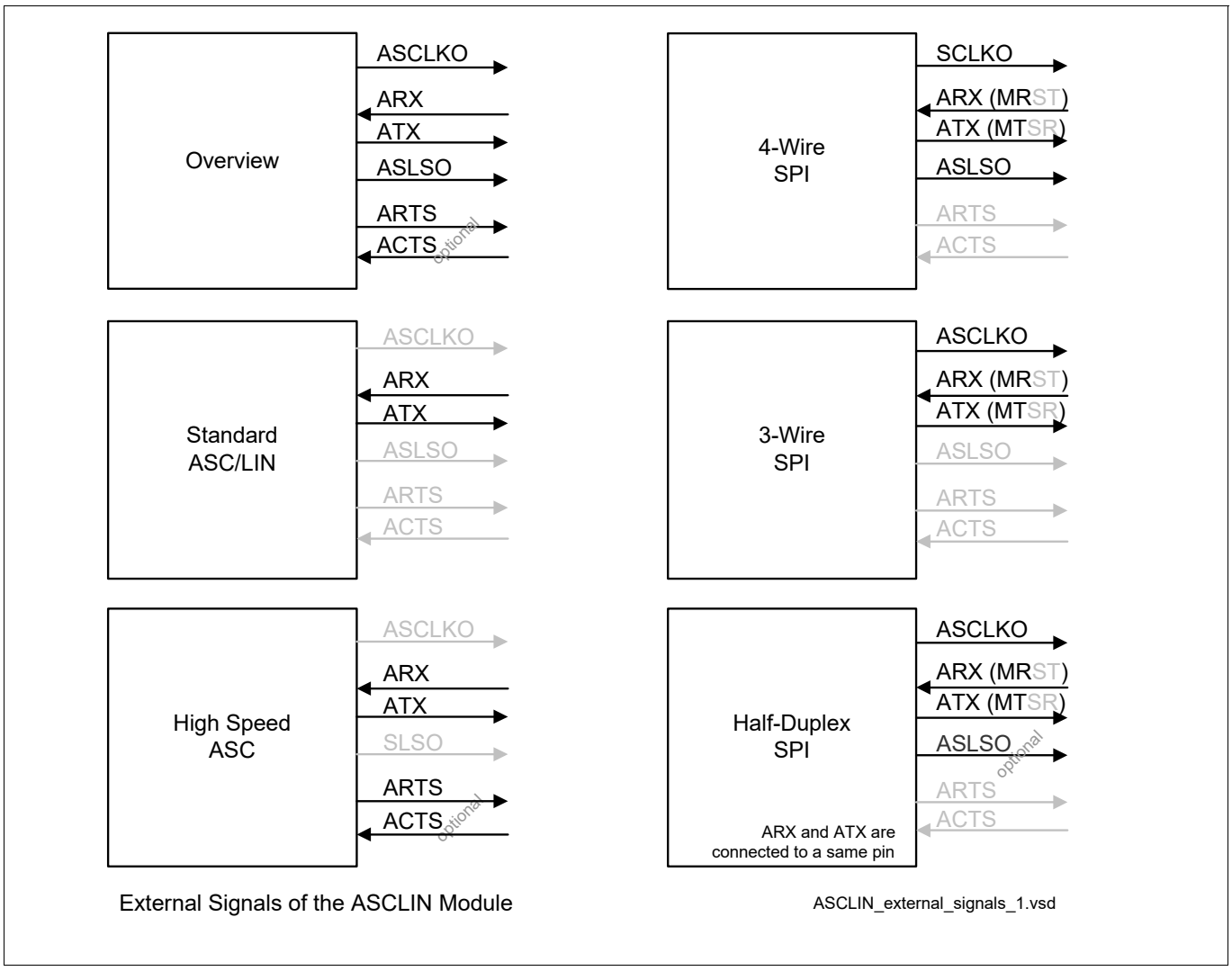


Figure 417 External Signals of the ASCLIN Module in Different Modes

36.3.2 User Interface

The user interface contains a Tx FIFO and a Rx FIFO.

They provide the following services: takes data packets with width optimal for the FPI[1)] bus and packs them to serial frames, buffers the FPI bus data, manages handshaking. These features are designed to optimize the use of the module in LIN, high-speed ASC and SPI mode. They also optimize the use of the module in standard ASC mode.

36.3.2.1 TxFIFO Overview

The Tx FIFO has the ability to pack 16-bit writes from the FPI bus to two up-to-8-bit frames or one up-to-16-bit frame. It also has the ability to pack 32-bit writes from the FPI bus to four up-to-8-bit or two up-to-16-bit frames.

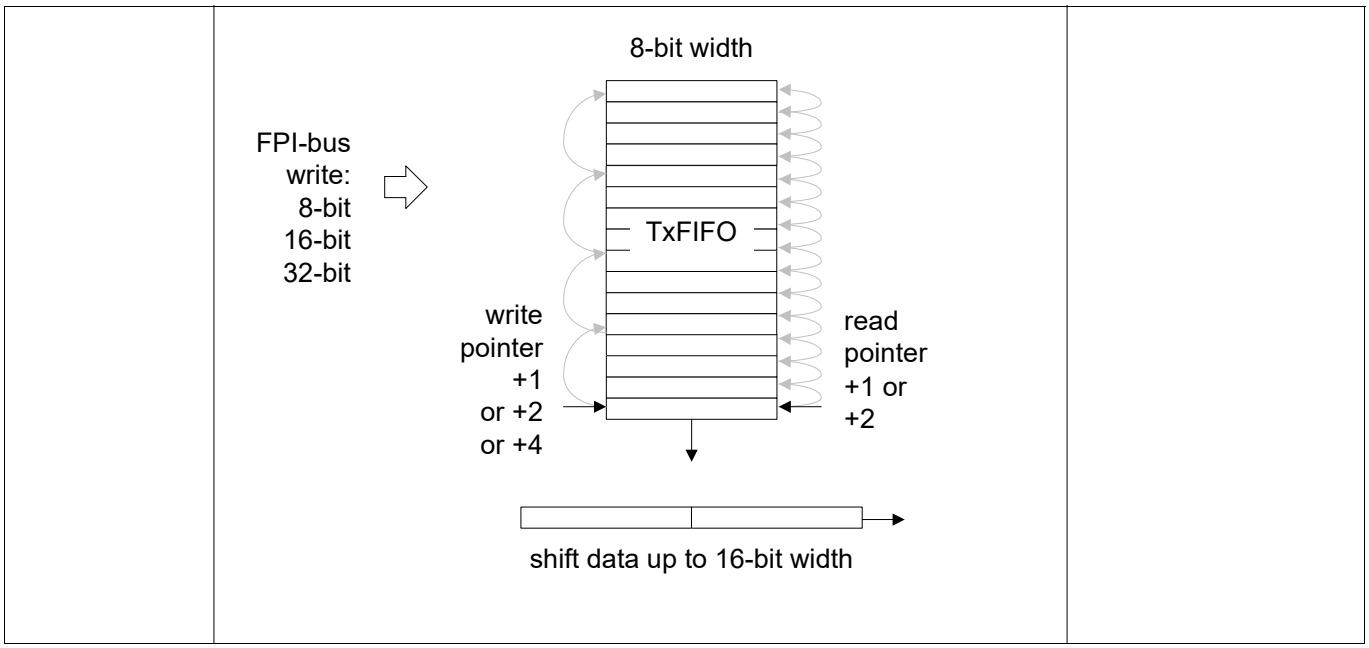


Figure 418 TXFIFO Overview

The number of bytes written to the TxFIFO is always defined with the TXFIFOCON.INW (**TX FIFO Configuration Register**), does not depend on the SPB write width, and should be equal to it.

If the SPB write width is shorter than INW, then the missing part is padded with zeros.

If the SPB write width is longer then INW, then the excessive part is lost.

On the shift register side, the TxFIFO is always emptied by a number of bytes as needed for the data width field DATCON.DATLEN (**Data Configuration Register**).

The table **Inlet Width versus SPB Write Width** describes all the possible write (fill) combinations. The bytes which will be filled into the TXFIFO are labeled as A, B, C, D and 0.

1) FPI is the protocol used on the System Peripheral Bus (SPB).

Table 324 Inlet Width versus SPB Write Width

TxFIFO Contents		SPB Bus Write Width		
		8-Bit A	**16-Bit BA**	**32-Bit DCBA**
Inlet Width INW	**8-Bit**	A	A	A
	16-Bit	0A	BA	BA
	32-Bit	000A	00BA	DCBA

36.3.2.2 Using the TxFIFO

The Tx FIFO has the ability to pack 16-bit writes from the FPI bus to two up-to-8-bit frames or one up-to-16-bit frame.

Note: Some (but not all) use-cases are described in the sections below. The software can always access the TXFIFO 8, 16 or 32 bit wide, according to its requirements.

36.3.2.2.1 Standard ASC Mode

The most standard usage of the ASC module is to transmit 7 or 8-bit values, and to fill the FIFO with 8-bit wide FPI-bus accesses. The FPI bus access is 8-bit wide, and there is only BS0 (Byte Select 0) signal active.

The transmit data is written to the address TXDATA (**Transmit Data Register**). The Tx_Inlet_Width is 8-bit, and the Tx_Outlet_Width is 8-bit.

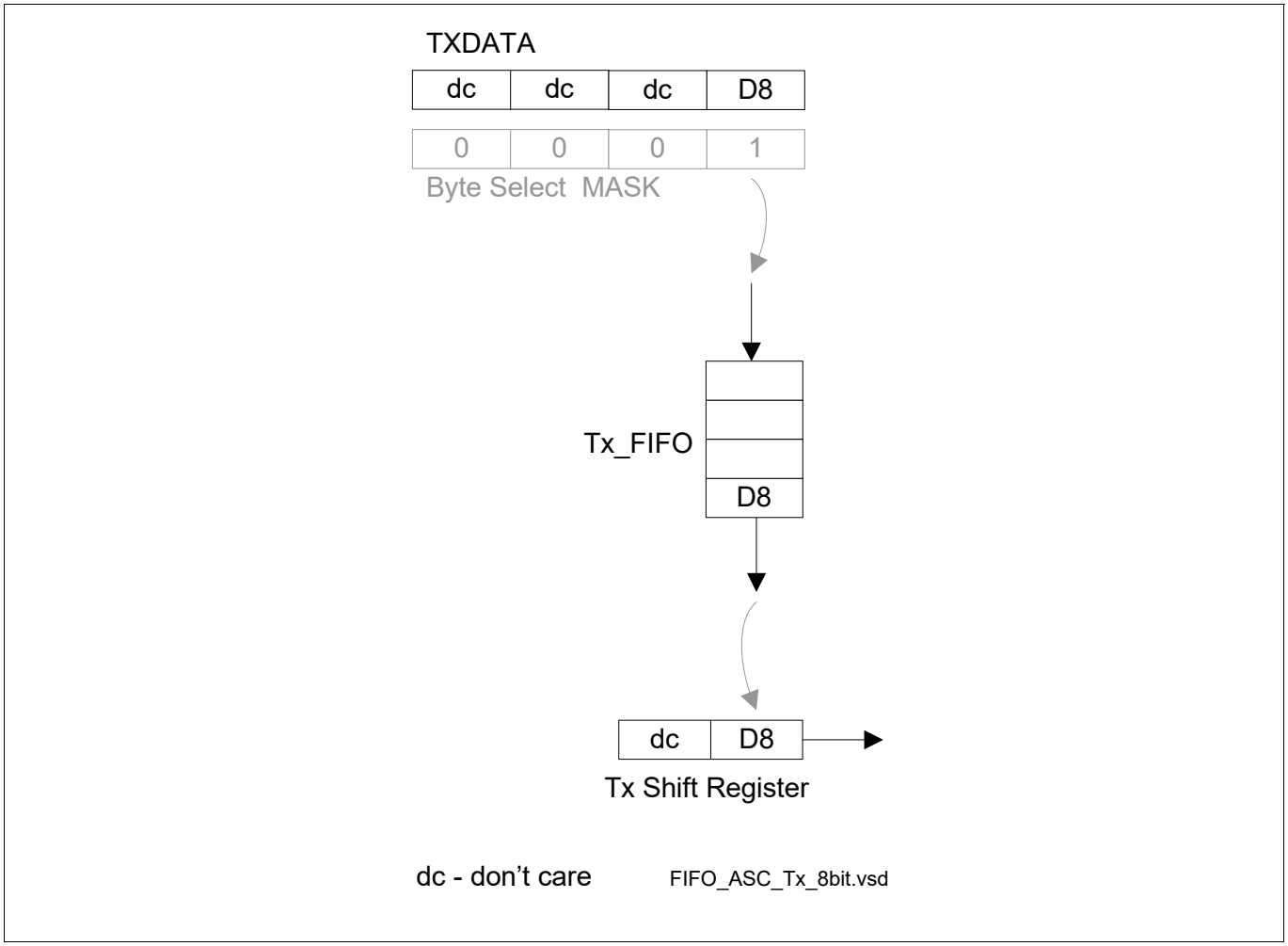


Figure 419 FIFO Operation in 7- or 8-Bit ASC Mode, with or without Parity

A less common mode of operation of the ASC module is to transmit 9-bit values, containing either 9-bit data or 8-bit data. The FPI bus access should be 16-bit wide, and there are BS0 and BS1 (Byte Select 0 and 1) signals active. The shift register is filled with two 8-bit values at the location read pointer and read pointer + 1.

The transmit data is written to the address TXDATA (**Transmit Data Register**).The Tx_Inlet_Width is 16-bit, and the Tx_Outlet_Width is 16-bit.

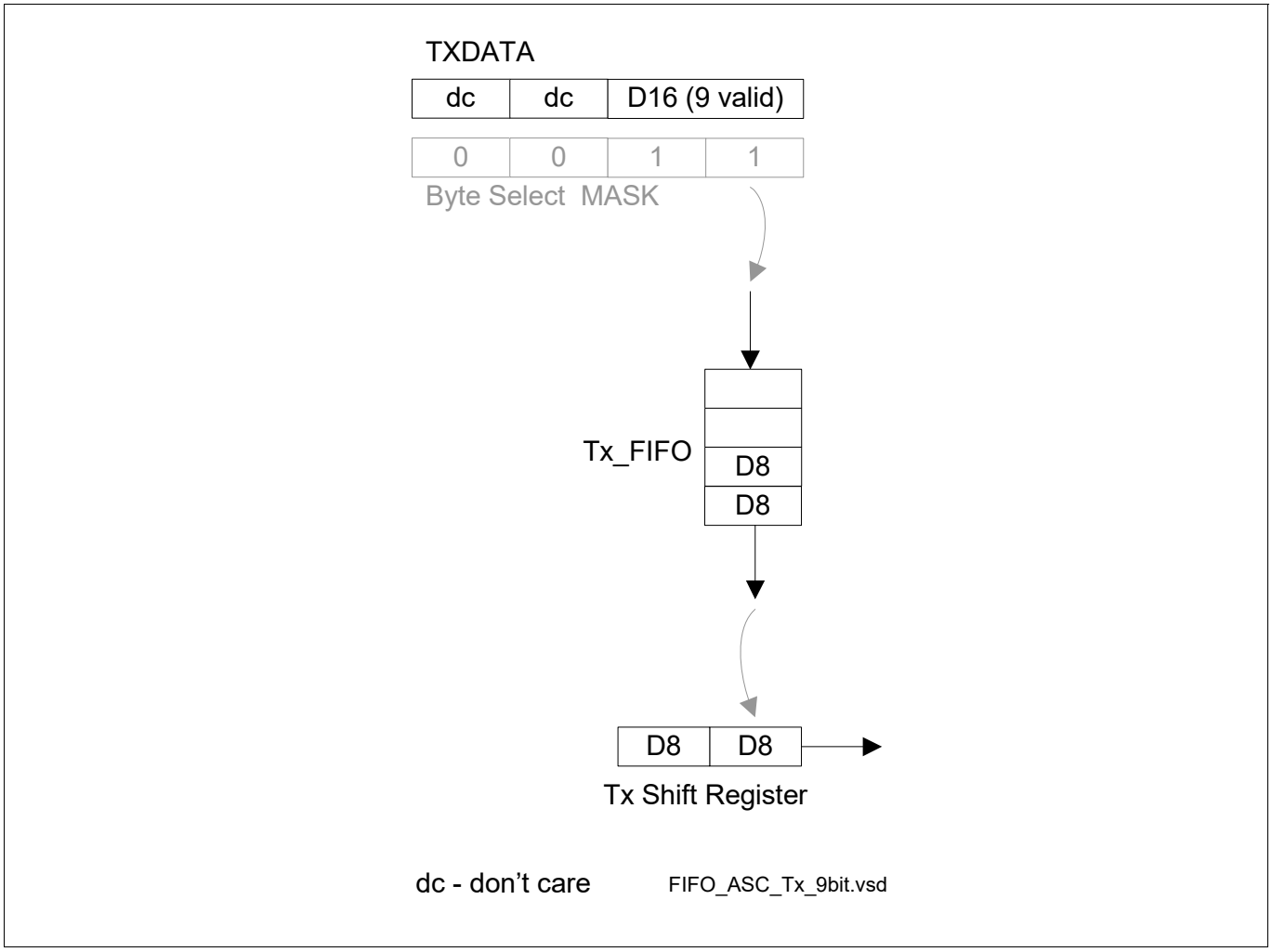


Figure 420 FIFO Operation in 9-Bit ASC Mode

36.3.2.2.2 High Speed ASC Mode

In order to reduce the FPI bus load by not using a 32-bit move for 7 or 8-bit values, one option is to fill the FIFO with 32-bit wide accesses containing four 8-bit wide values. All Byte Select signals BS[3:0] are active, and all four bytes are written in the TX FIFO in one cycle. At the same time, the write pointer of the TXFIFO jumps by the amount of four.

The transmit data is written to the address TXDATA (**Transmit Data Register**). The Tx_Inlet_Width is 32-bit, and the Tx_Outlet_Width is 8-bit.

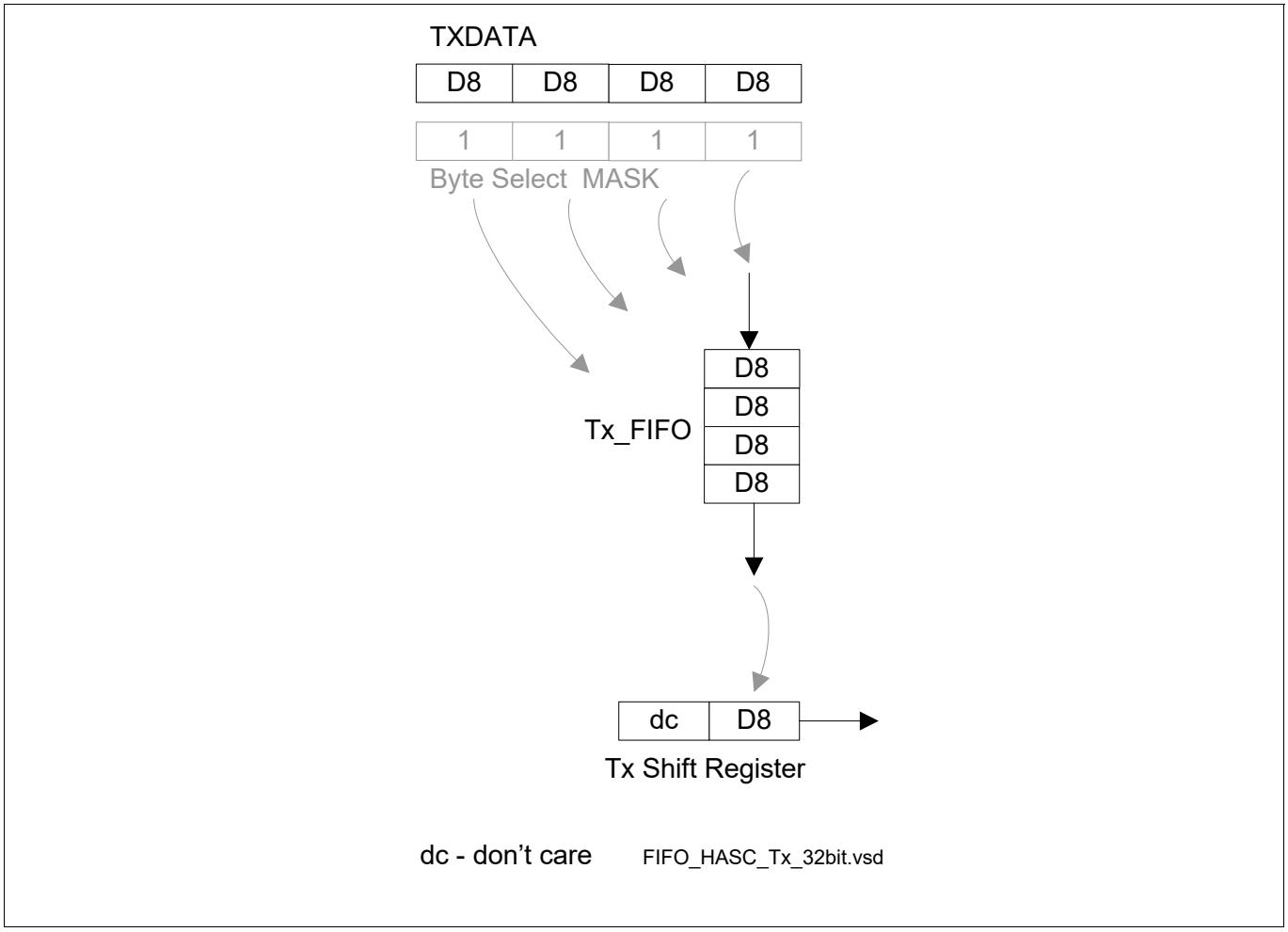


Figure 421 High-Speed Communication Scenario

36.3.2.2.3 LIN Mode

In LIN mode, several 8-bit frames are preceded by a low break pulse.

The generation of the break pulse is programmable with an 6 bit timer in units of bits, where the wide range of break pulses can be generated, for example in a range between 13 and 26 bits and beyond (up to 64 bits, but these lengths would violate the maximum LIN header length). The detection of the break pulse is programmable with an 8 bit timer in units of bits. Among others, the standard thresholds of 10 and 11 bit times can be set.

The pulse is generated by an module internal timer. The transmit sequence consists of filling the stopped TXFIFO (**Transmit Data Register**) with the appropriate bytes, and then starting the break pulse, which activates the TxFIFO.

The transmit data is written to the address TXDATA (**Transmit Data Register**). The Tx_Inlet_Width can be 8-bit (or 16-bit, or 32-bit), and the Tx_Outlet_Width is 8-bit.

36.3.2.2.4 SPI Mode

SPI mode is used most often to send or receive data of 8 or 16-bits length, or some length in between. Therefore the reading out of the Tx FIFO must be 8 or 16-bit wide, and write could be 16-bit wide, or even 32-bit wide if two frames are packed in one FPI bus access.

The transmit data is written to the address TXDATA (**Transmit Data Register**). The Tx_Inlet_Width is up to 32-bit, and the Tx_Outlet_Width is up to 16-bit.

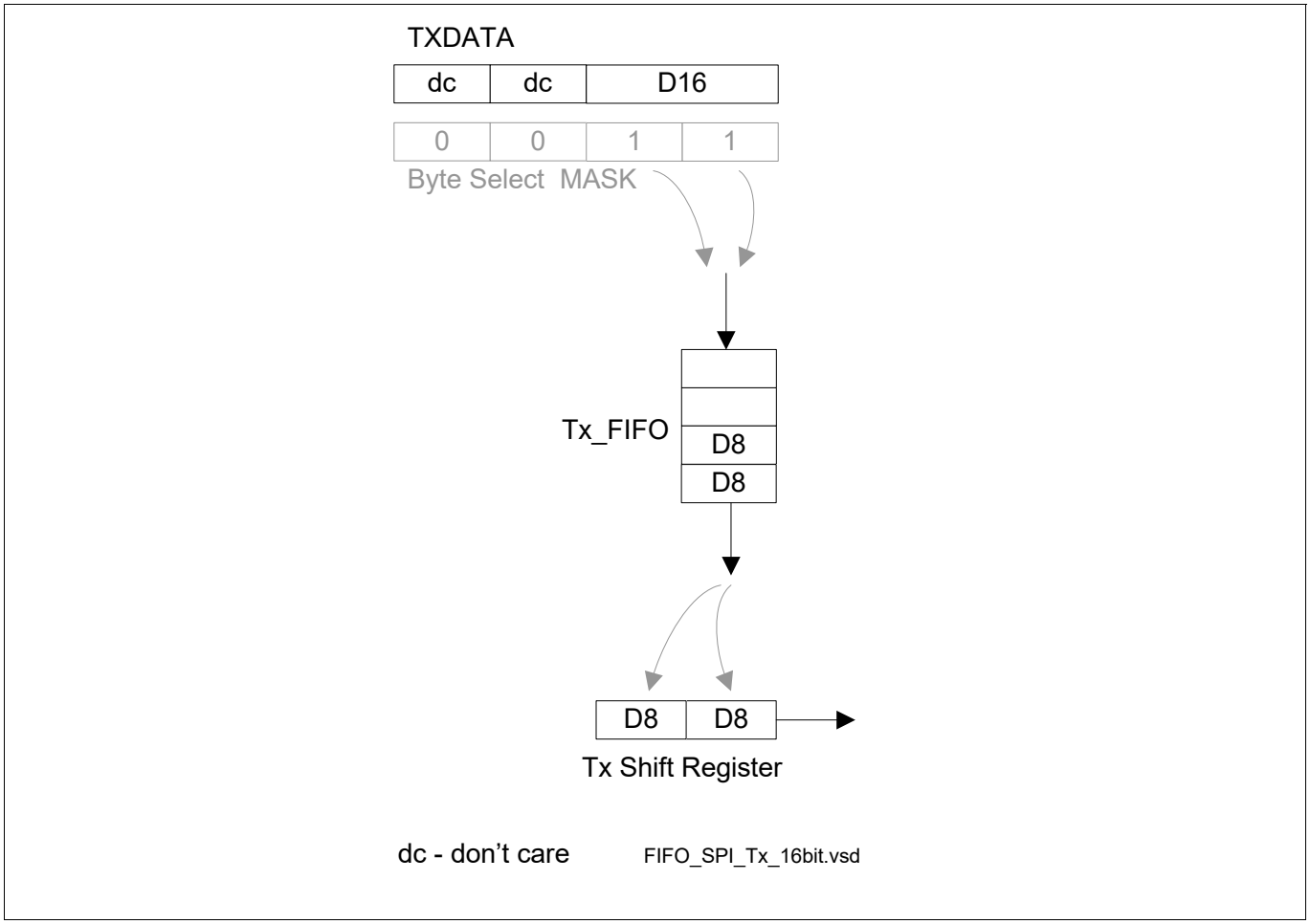


Figure 422 TXFIFO Operation in SPI Mode

36.3.2.3 TXFIFO Interrupt Generation

The TXFIFO provides three interrupt generation modes, selected by the bit-field TXFIFOCON.FM (**TX FIFO Configuration Register**):

- Single Move Mode
- Batch Move Mode
- Combined Mode

The RXFIFO and the TXFIFO can use the interrupt generation modes independently of each other.

Single Move Mode

The purpose of the Single Move Mode is to keep the TXFIFO as full as possible, refilling the TXFIFO by writing to it as soon as there is a free element.

The single move mode supports primarily a DMA operation using single move per TXFIFO interrupt. In this mode the DMA keeps the TXFIFO full. A DMA request is triggered each time a write to the TXFIFO is performed, and there is at least one empty element available.

If the write:

- makes the TXFIFO full (no more empty elements), or
- the next write would make the TXFIFO overflow taking TXFIFOCON.INW (**TX FIFO Configuration Register**) into account

an interrupt is not generated in order to prevent overflow. It is generated later when the shift register reads one element from the TXFIFO, making it sufficiently not full to accept one INW wide element.

The **Figure 423**, shows the filling levels of the TXFIFO, and the events associated with each filling level triggering a TXFIFO refill interrupt.

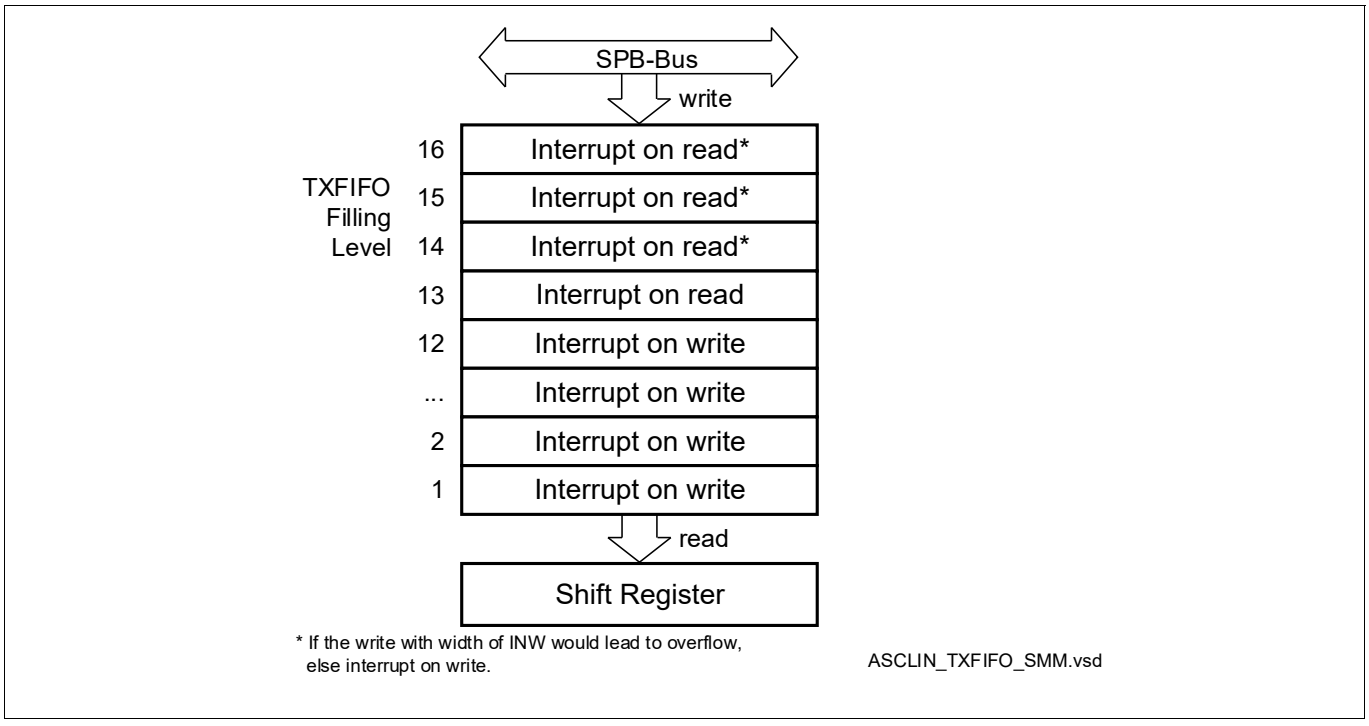


Figure 423 Interrupt Generation in the Single Move Mode

In order to initiate the very first interrupt in the chain of the refill interrupts after power-on reset, it is necessary that the software either performs one write to the TXFIFO or by setting the interrupt flag in the TXFIFO interrupt node. Afterwards, the (DMA) interrupt-chain is self sustaining until the whole transaction is over.

Attention: ***In Single Move Mode multiple software writes or block DMA moves would lead to multiple interrupts and (false) transaction lost events. Therefore they should be avoided - only single moves should be used. In Single Move Mode, when the TXFIFO is almost full (the next write operation causes the TXFIFO full) or when the TXFIFO is full, a flush on TXFIFO will trigger an interrupt (FLAGS.TFL)***

Batch Move Mode

Batch Move Mode supports the following use case:

- CPU servicing the TXFIFO

The purpose of the Batch Move Mode is to reduce the number of interrupts by triggering an interrupt when more then one TXFIFO elements are empty. For example, a CPU can be interrupted less frequently and it can perform more than one moves per interrupt.

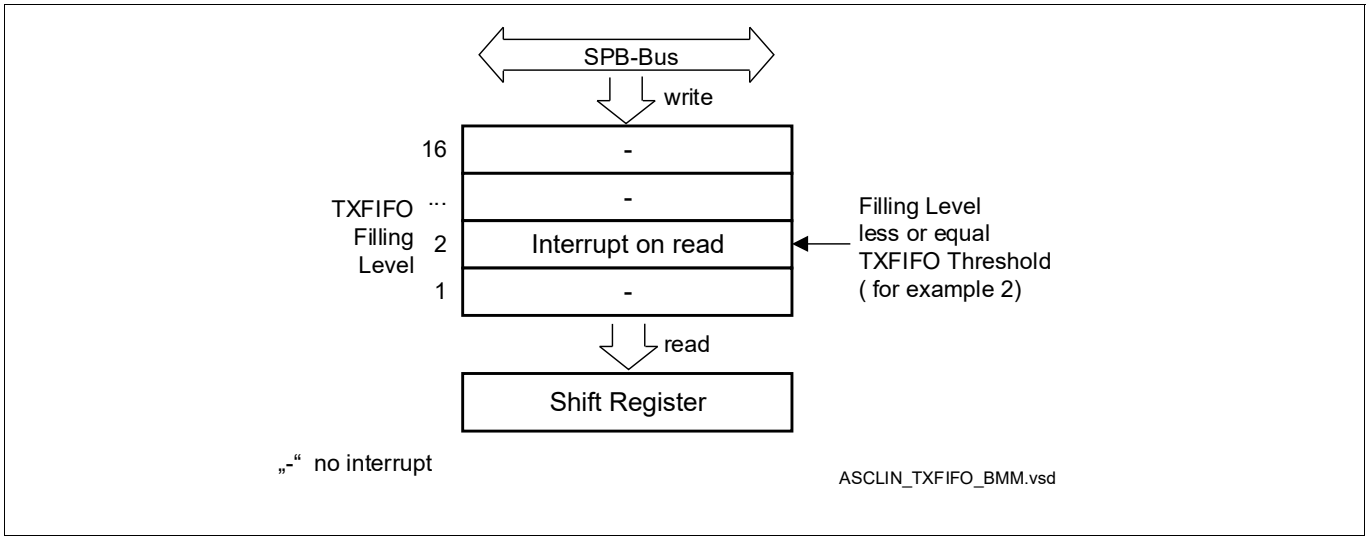


Figure 424 TXFIFO - Interrupt Generation in the Batch Move Mode

In Batch Move Mode, an interrupt is generated only at one point in the TXFIFO, when the filling level falls to or below the programmed threshold, as described in TXFIFOCON.INTLEVEL, implying that there are at least the predefined number of empty TXFIFO elements available.

Attention: ***The TXFIFO must be refilled until the filling level has risen above the interrupt threshold (ensured by polling the filling level), in order to guarantee that the next interrupt will occur.***

At high baud rates and short frames, it could be possible that a couple of frames have been transmitted until the moment the TXFIFO is refilled. For example, if the threshold is set to three full elements, then the interrupt will be set when the filling level falls below three, making two elements free. If the refill moment of the TXFIFO is delayed so long that two frames have been transmitted, the TXFIFO will become empty, and refill sequence of two will not reach the level of three. So, the next interrupt would never come unless the filling level is being polled and the CPU keeps refilling in a loop until the threshold has been passed (ideally the TXFIFO would be filled completely with each batch of moves).

Combined (Compatibility) Mode

An interrupt is generated when the filling level falls to INTLEVEL or below, each time when a data byte is taken out of the FIFO, as defined in the TXFIFOCON.INTLEVEL (**TX FIFO Configuration Register**). This is the mode which is compatible with the previous versions of the ASCLIN module and described in the TXFIFOCON.INTLEVEL bit field.

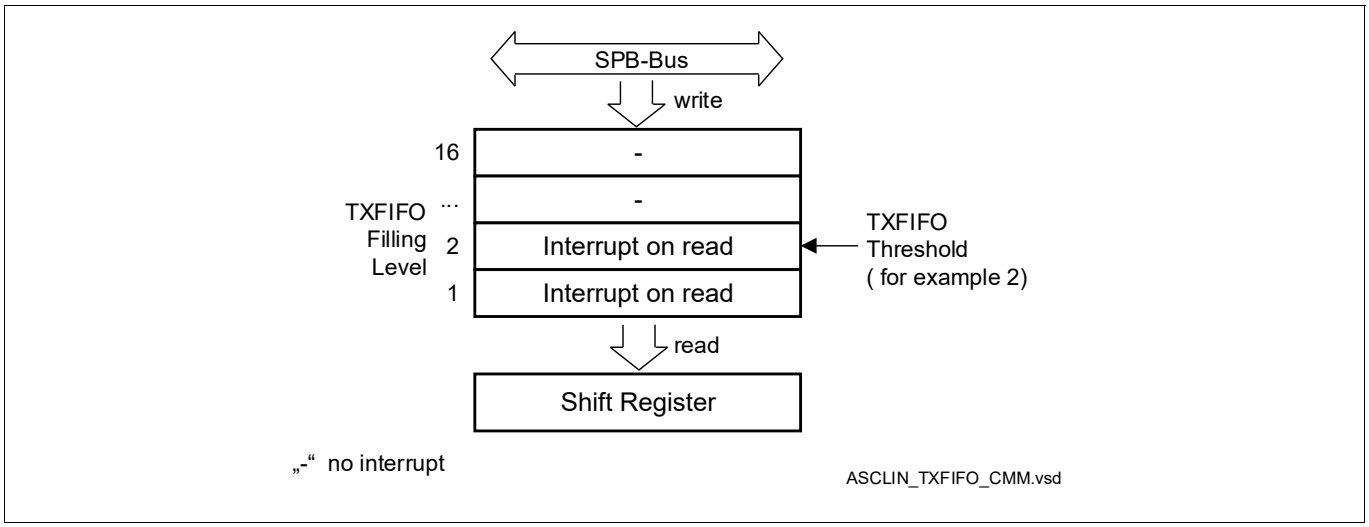


Figure 425 TXFIFO - Interrupt Generation in the Combined Mode

36.3.2.4 RxFIFO Overview

The Rx FIFO has the ability to pack two up-to-8-bit frames or one up-to-16-bit frame to one 16-bit write to the FPI bus. It also has the ability to pack four up-to-8-bit or two up-to-16-bit frames to one 32-bit write to the FPI bus.

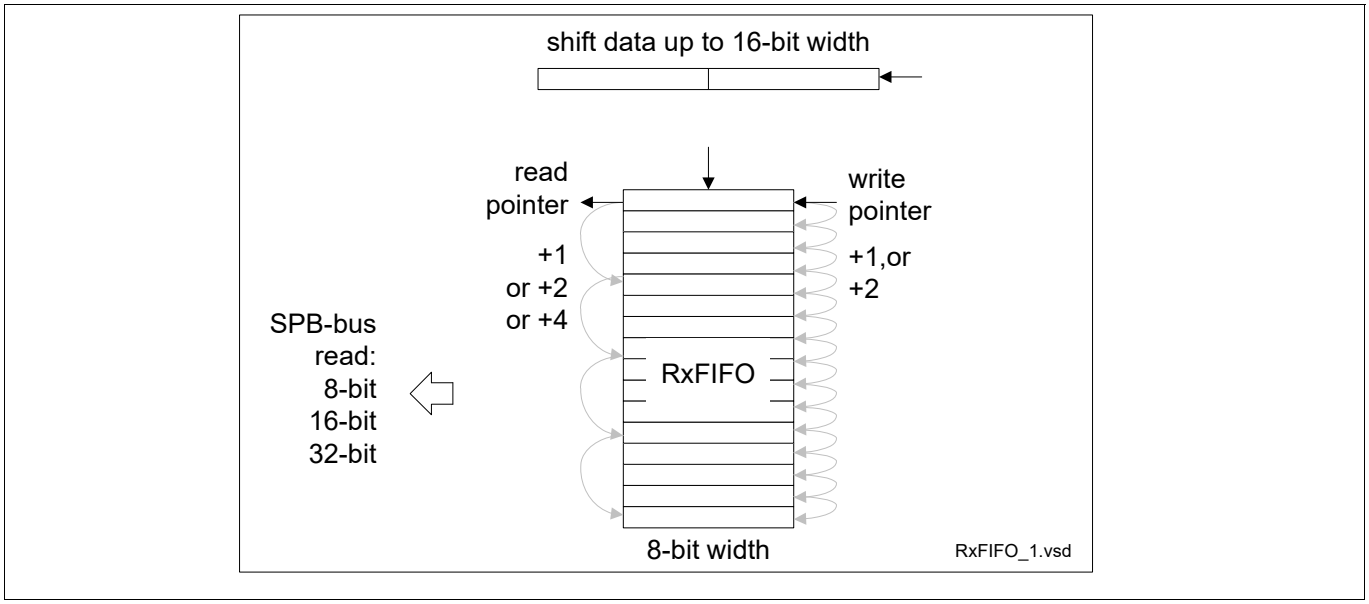


Figure 426 RXFIFO Overview

The number of bytes taken out of the RxFIFO is always RXFIFOCON.OUTW (**RX FIFO Configuration Register**), does not depend on the SPB bus read width, which should be normally equal to OUTW.

If the SPB read width is shorter than the OUTW, then the excessive part is lost.

If the SPB read width is longer than the OUTW, then the missing part is padded with zeros.

On the shift register side, the RxFIFO is always filled with a number of bytes as needed for the data width field DATCON.DATLEN (**Data Configuration Register**).

The table **Outlet Width versus SPB Read Width** describes all the possible combinations. The bytes in the RxFIFO are named A, B, C and D, A being the first one to take out. The padding byte is 0.

*Note: In case of an underflow, i.e. when a read access is performed to RXDATA (**Receive Data Register**) and the number of bytes to be taken out of the FIFO as given by the setting of RXFIFOCON.OUTW (**RX FIFO Configuration Register**) exceeds the number of bytes that are actually stored in the FIFO, as indicated by RXFIFOCON.FILL, then the RXFIFO delivers the available data padded with zeros up to the read access width. However, the data remain in the FIFO and the filling level RXFIFOCON.FILL is not changed. To remove the data, the user software must flush the RXFIFO. The bit FLAGS.RFU (**Flags Register**) is set.*

Table 325 Outlet Width versus SPB Read Width

SPB-Read Data, taken from the RxFIFO		SPB Bus Read Width		
		8-Bit	**16-Bit**	**32-Bit**
Outlet Width OUTW	**8-Bit**	A	0A	000A
	16-Bit	A	BA	00BA
	32-Bit	A	BA	DCBA

36.3.2.5 Using the RxFIFO

36.3.2.5.1 Standard ASC Mode

The received data is read from the address RXDATA (**Receive Data Register**).

In case of data width of 7 or 8 bits, one read on this address delivers one byte and empties the FIFO for one element, if the Rx_Outlet_Width is 8-bit. The Rx_Inlet_Width is 8-bit.

In case of data width of 9 bits, one read on this address delivers two byte and empties the FIFO for two elements, if the Rx_Outlet_Width is 16-bit. The Rx_Inlet_Width is 16-bit.

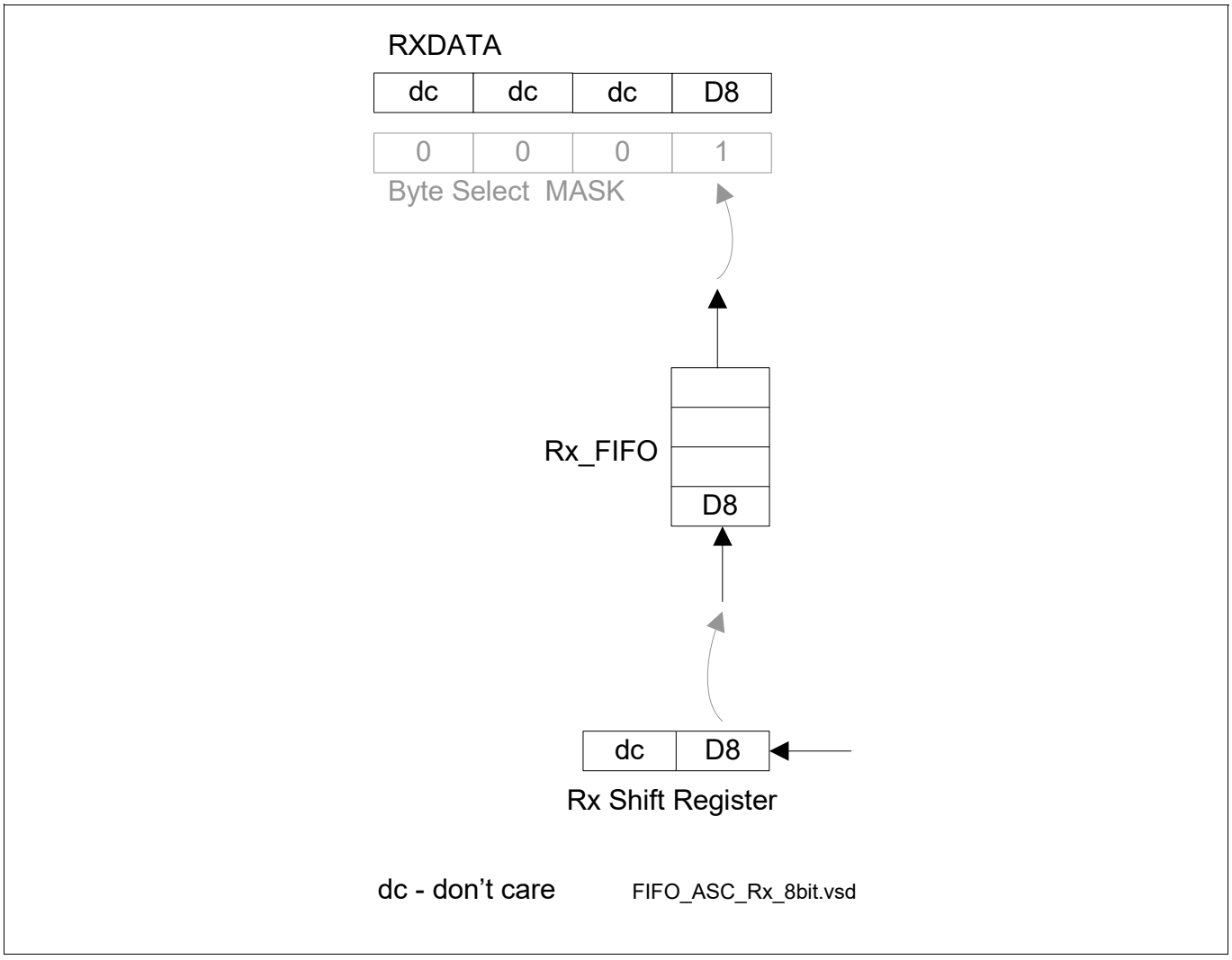


Figure 427 8-Bit ASC Reception

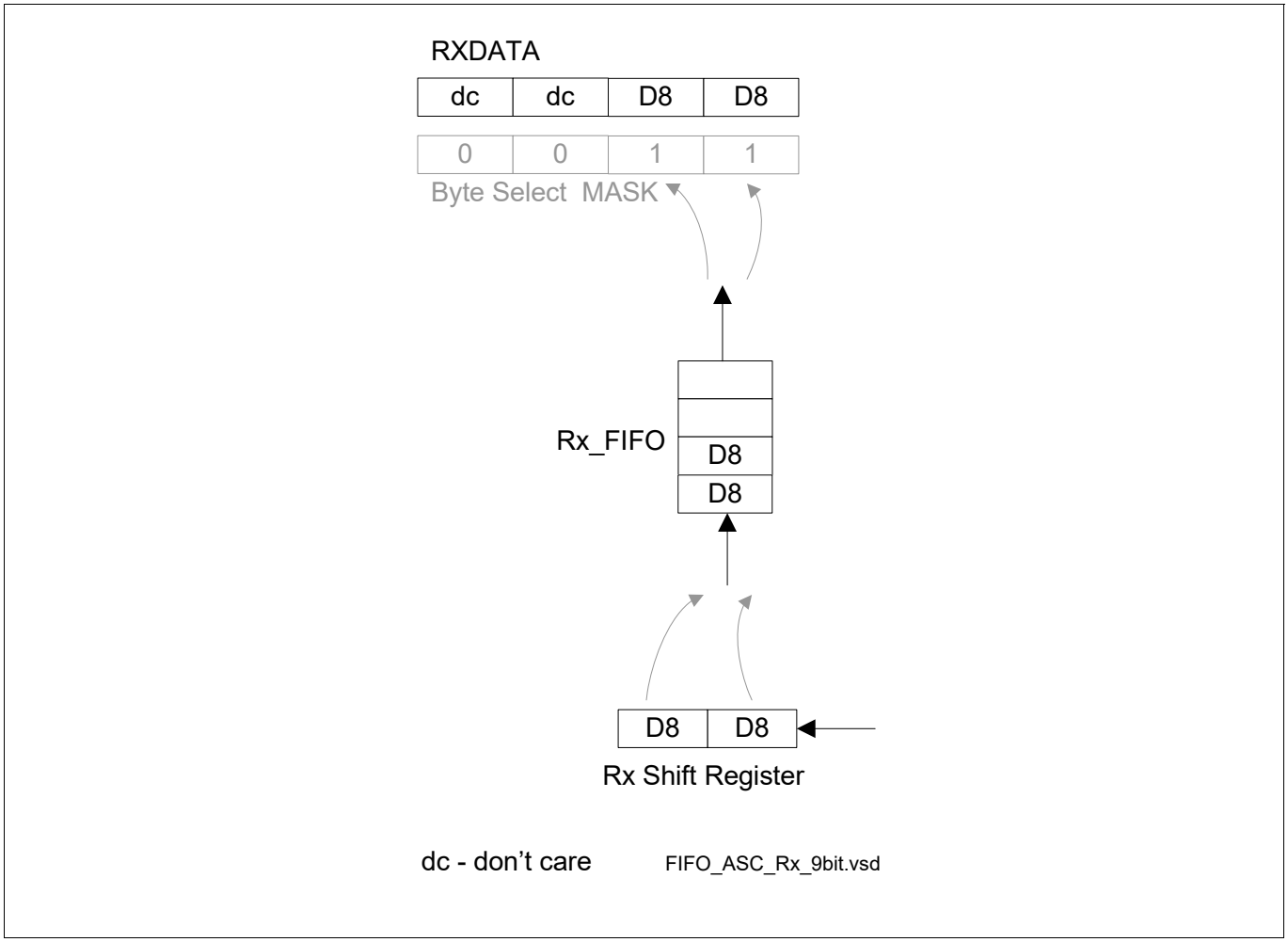


Figure 428 9-Bit ASC Reception

36.3.2.5.2 High Speed ASC Mode

The received data is read from the address RXDATA (**Receive Data Register**).

In case of data width of 7 or 8 bits, one read on this address can deliver (one or two or) four bytes and empties the FIFO by four elements, if the Rx_Outlet_Width is 32-bit. The Rx_Inlet_Width is 8-bit.

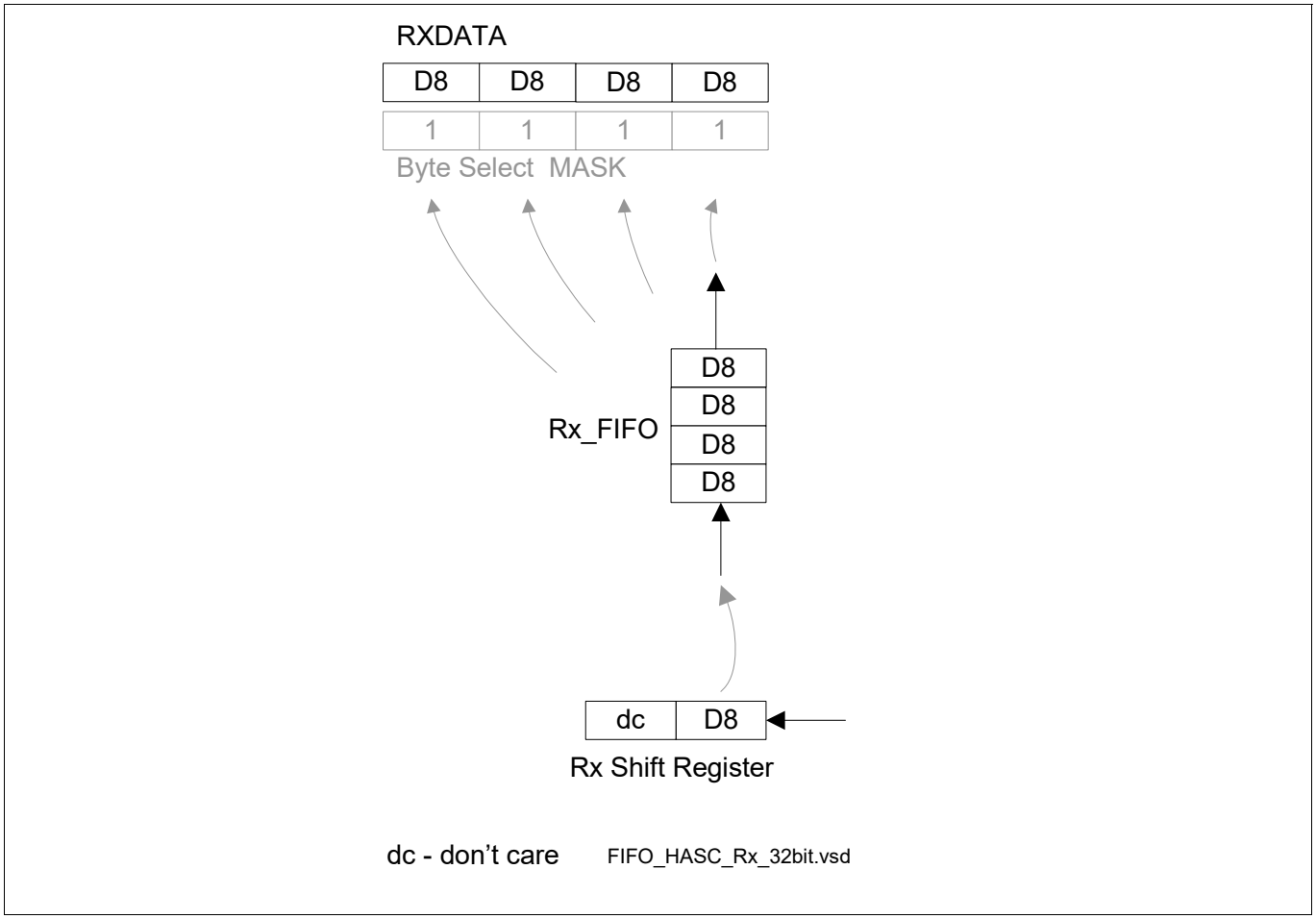


Figure 429 High Speed ASC Communication

36.3.2.5.3 LIN Mode

The received data is read from the address RXDATA (**Receive Data Register**).

The read data width is 8 bits (or 16 bit or 32 bit), one read on this address delivers one byte and empties the FIFO by one element, if the Rx_Outlet_Width is 8-bit. The Rx_Inlet_Width is 8-bit

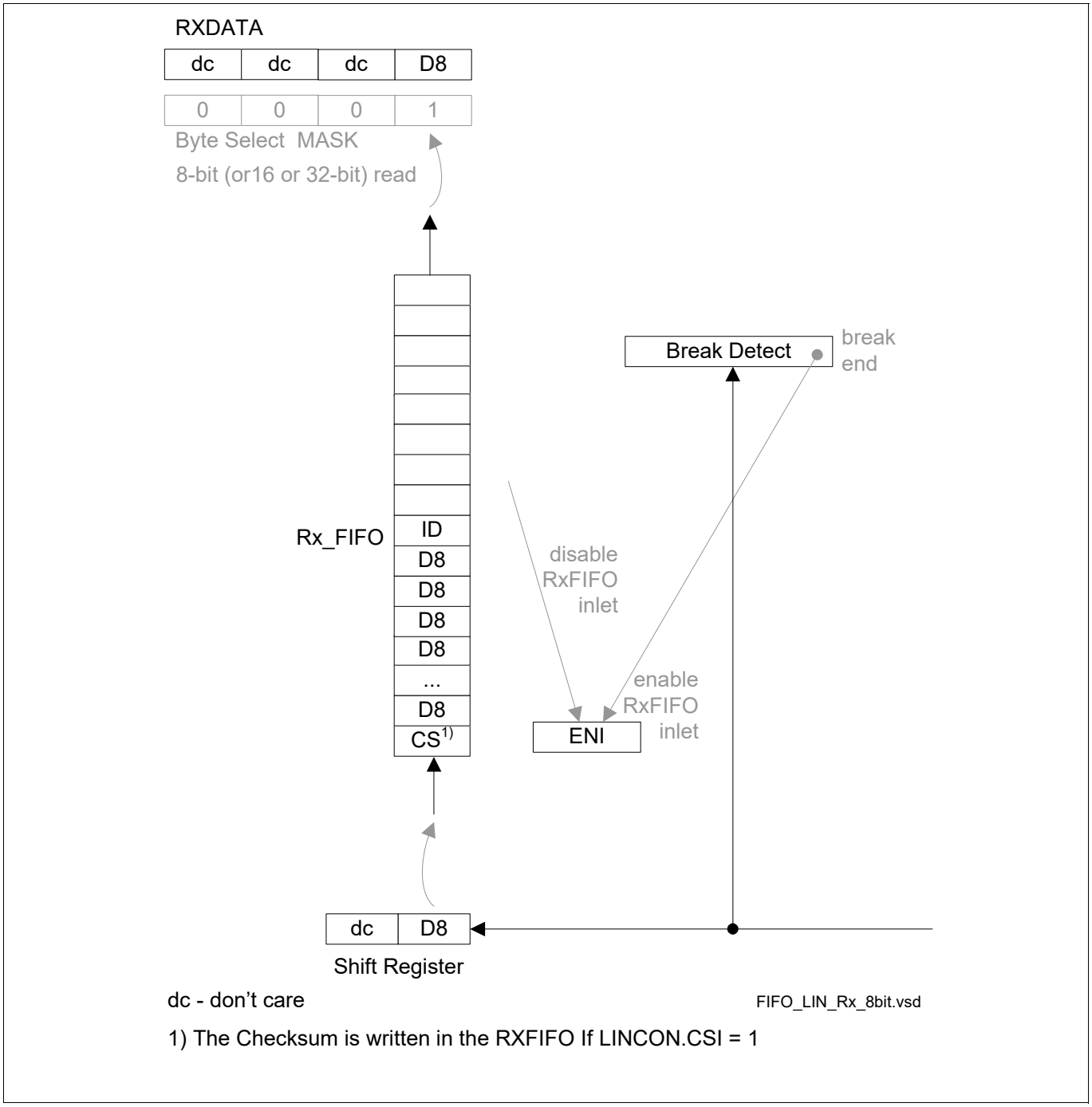


Figure 430 RXFIFO Operation in LIN Mode

36.3.2.5.4 SPI Mode

The received data is read from the address RXDATA (**Receive Data Register**).

In case of data width of 16 bits, one read on this address delivers two bytes and empties the FIFO by two elements, if the Rx_Outlet_Width is 16-bit. The Rx_Inlet_Width is 16-bit

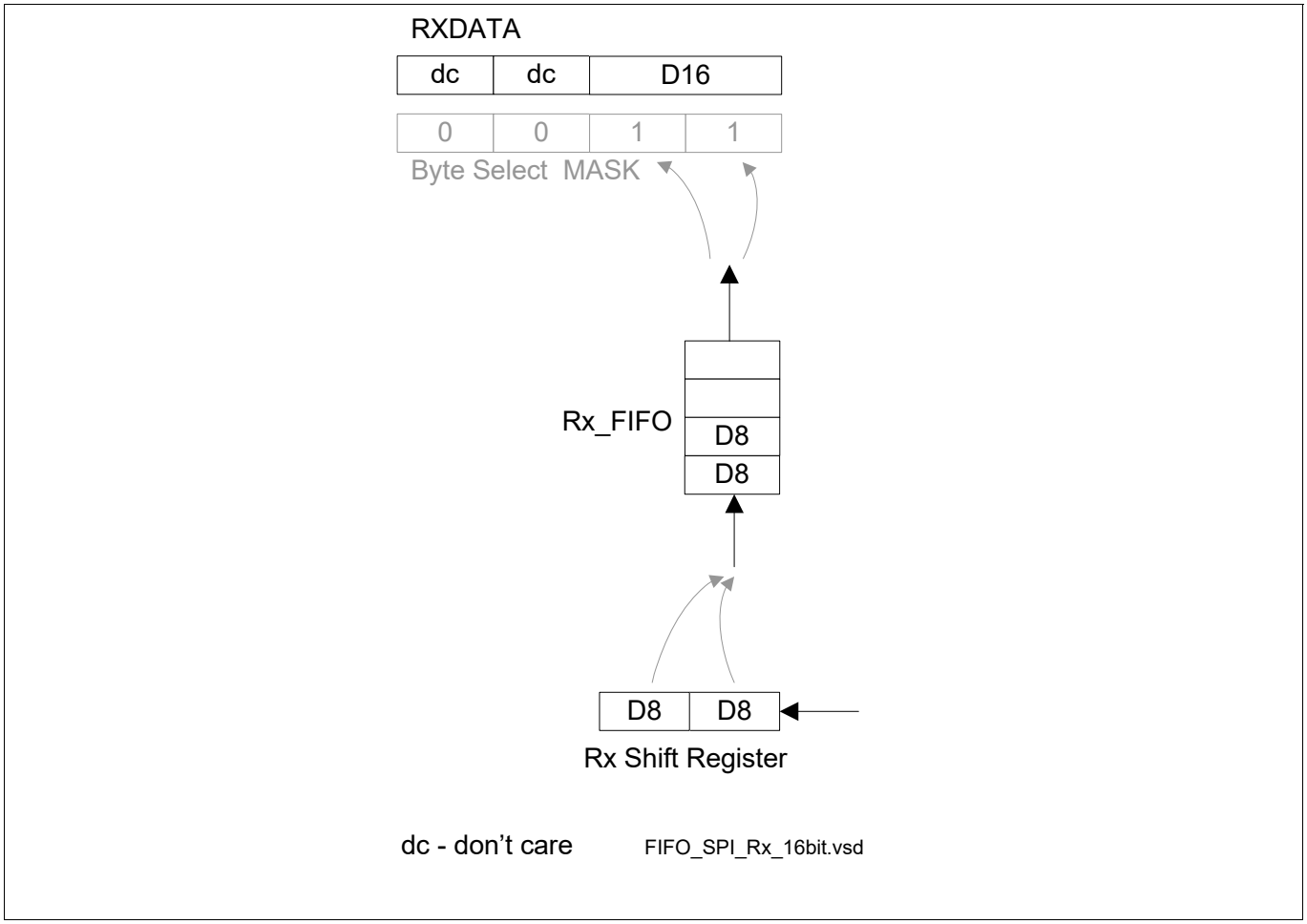


Figure 431 RXFIFO Operation in SPI Mode

36.3.2.6 RTS / CTS Handshaking

A receiver deactivates the RTS (Request to Send) output signal when the RXFIFO is almost full, in order to avoid its overload. When the emptying of the RXFIFO starts, for example by a DMA, and the filling level falls below the threshold level, the RTS output is activated again. The threshold level is fixed to RXFIFO size minus four, in order to support byte, two bytes and four bytes DMA transfers.

The transmitter receives the RTS output of the receiver on its CTS input and accordingly pauses and resumes the transmission.

36.3.2.7 RXFIFO Interrupt Generation

The RXFIFO provides three interrupt generation modes, selected by the bit field RXFIFOCON.FM (**RX FIFO Configuration Register**):

- Single Move Mode
- Batch Move Mode
- Combined (Compatibility) Mode

Single Move Mode

The purpose of the Single Move Mode is to keep the RXFIFO as empty as possible, by fetching the received elements one by one as soon as possible.

The single move mode supports primarily a DMA operation using single move per RXFIFO interrupt. In this mode the DMA keeps the RXFIFO as empty as possible. A DMA request is triggered each time a read from the RXFIFO is performed, and the RXFIFO is afterwards still not empty.

If the read:

- makes the RXFIFO empty, or
- the next read would make the RXFIFO underflow taking RXFIFOCON.OUTW (**RX FIFO Configuration Register**) into account

an interrupt is not generated in order to prevent underflow. It is generated later when the shift register writes one element to the RXFIFO, making it sufficiently full to provide one OUTW wide element.

The **Figure 432**, shows the filling levels of the RXFIFO, and the events associated with each filling level triggering a RXFIFO refill interrupt.

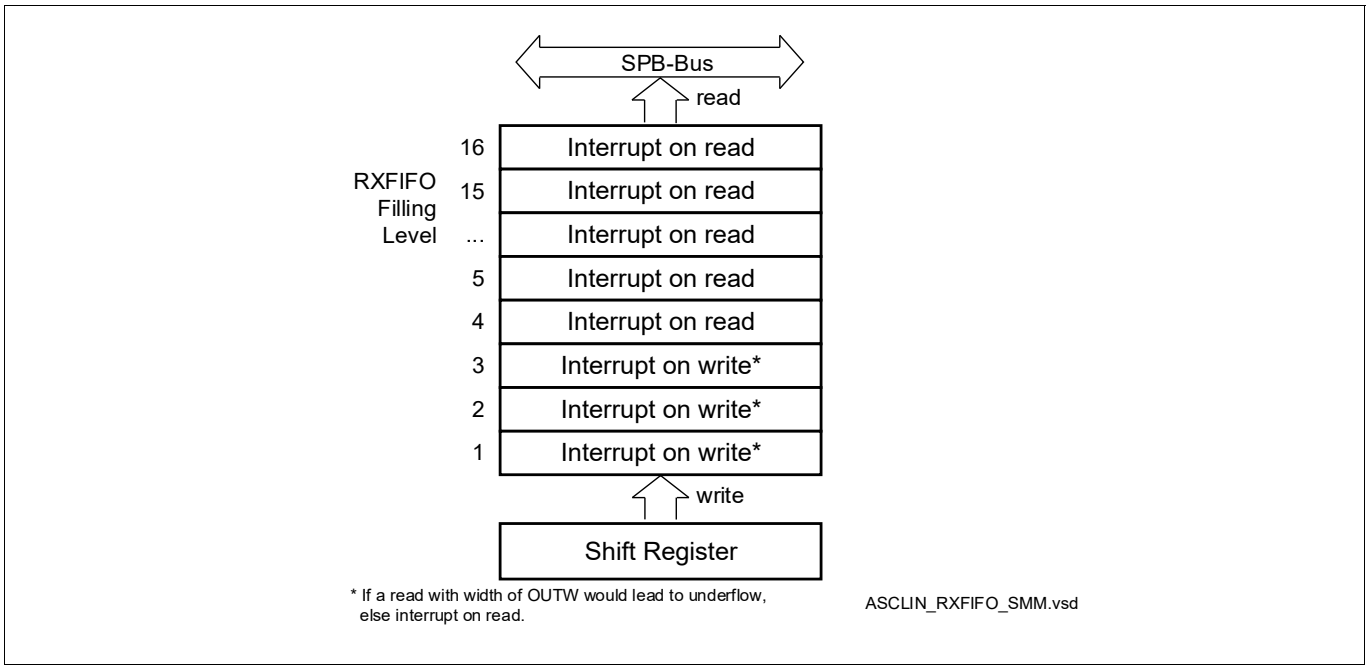


Figure 432 RXFIFO - Interrupt Triggering in the Single Move Mode

The initial RXFIFO interrupt is triggered by the Shift Register, after it delivers the first received element. Afterwards, the DMA trigger-chain of re-fetch interrupts is self sustaining until the whole transaction is over. At the end of the DMA transaction, there is no service request remaining active in the service request node, due to the fact that the read of the last element in the RXFIFO does not trigger an interrupt.

Attention: ***In Single Move Mode multiple software reads or block DMA moves lead to multiple interrupts and (false) transaction lost events. Therefore they should be avoided - only single moves should be used.***

Batch Move Mode

The purpose of the Batch Move Mode is to reduce the number of interrupts by triggering an interrupt when more then one RXFIFO elements are full. For example, a CPU can be interrupted less frequently and it can perform more than one move per interrupt.

Batch Move Mode supports the following use case:

- CPU servicing the RXFIFO

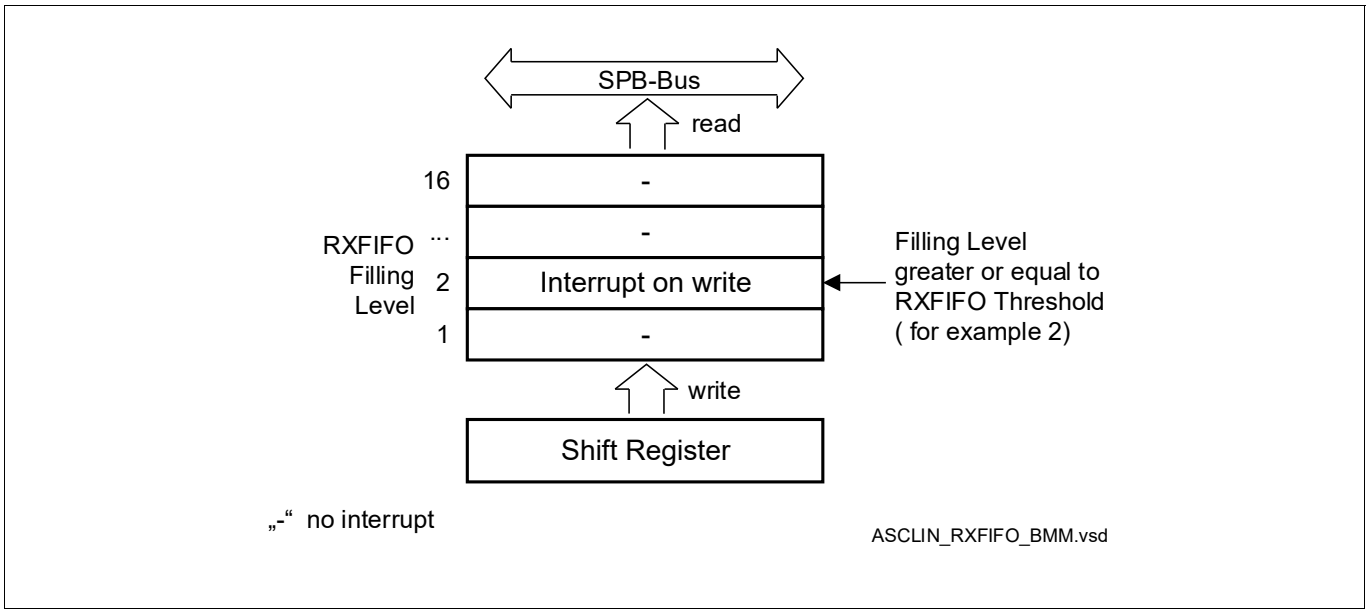


Figure 433 RXFIFO - Interrupt Triggering in the Batch Move Mode

In Batch Move Mode, an interrupt is generated only at one point in the RXFIFO, when the filling level rises to or above the programmed threshold, implying that there are at least the predefined number of full RXFIFO elements available. The description refers to the interpreted value of RXFIFOCON.INTLEVEL (register value +1) (**RX FIFO Configuration Register**).

Attention: ***It must be guaranteed that the CPU keeps emptying the RXFIFO and polling the RXFIFO filling level until the filling level has fallen below the interrupt threshold (or the RXFIFO is empty), in order to guarantee that next interrupt will occur.***

At high baud rates and short frames, it could be possible that more than one frames have been received until the moment the RXFIFO is re-emptied. For example, if the threshold is set to two full elements, then the interrupt will be set when the filling level rises above two. If the emptying moment of the RXFIFO is delayed so long that two additional frames have been received, the RXFIFO will become full, and re-empty sequence of two will not reach the level of two. So, the next interrupt would never come. This effect can not occur with threshold levels of 3 and 4. The threshold level of 1 is possible, but does not make much sense (use single move mode instead).

Combined (Compatibility) Mode

An interrupt is generated when the filling level rises to INTLEVEL or beyond, each time when a data byte is delivered to the FIFO, as defined in the RXFIFOCON.INTLEVEL (**RX FIFO Configuration Register**). The description refers to the interpreted value of RXFIFOCON.INTLEVEL (register value +1). This is the mode which is compatible with the previous versions of the ASCLIN module and described in the RXFIFOCON.INTLEVEL bit field.

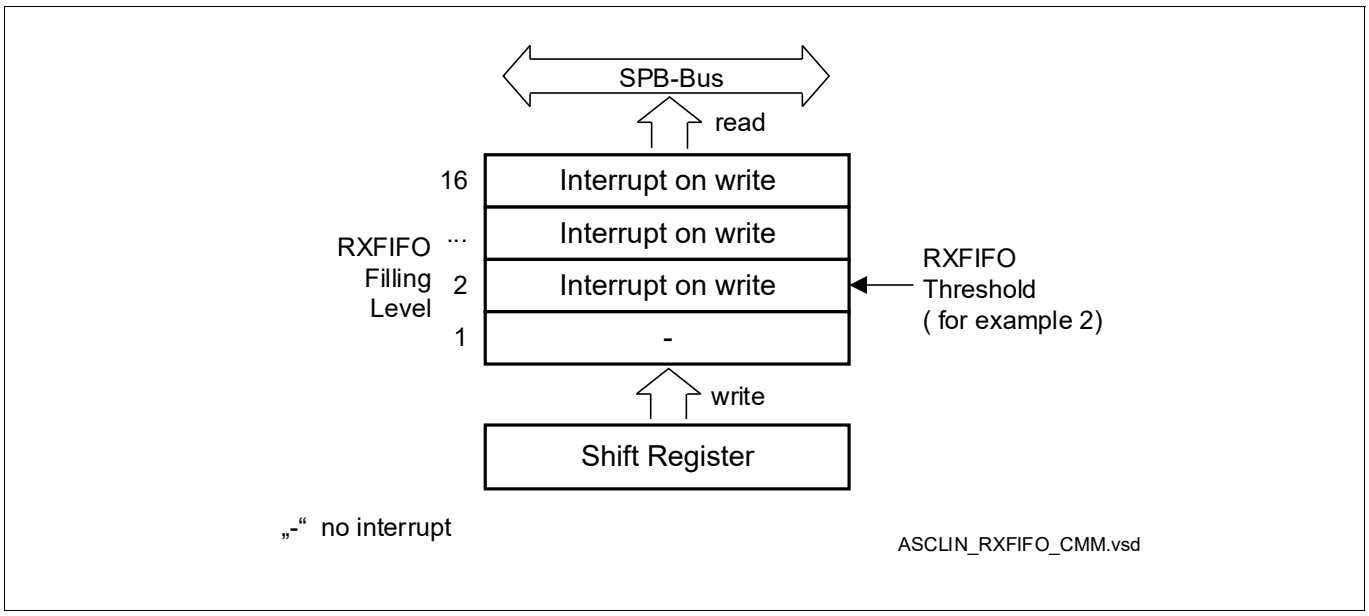


Figure 434 RXFIFO - Interrupt Triggering in the Combined Mode

36.3.3 Clock System

The clock system generates all the clocks needed for the proper operation of the ASCLIN module: digital filter clock, oversampling clock, bit time and serial SPI clock.

The clock used for the clock system f_A is independent from the SPB bus clock and remains constant if the SPB bus clock changes, for example in power saving scenarios. The bit field CSR.CLKSEL (**Clock Selection Register**) selects the clock source for the f_A frequency, which can be synchronous or asynchronous, higher or lower than the SPB bus frequency.

36.3.3.1 Baud Rate Generation

Fractional Divider, n-divider and oversampling divider with configurable sample point.

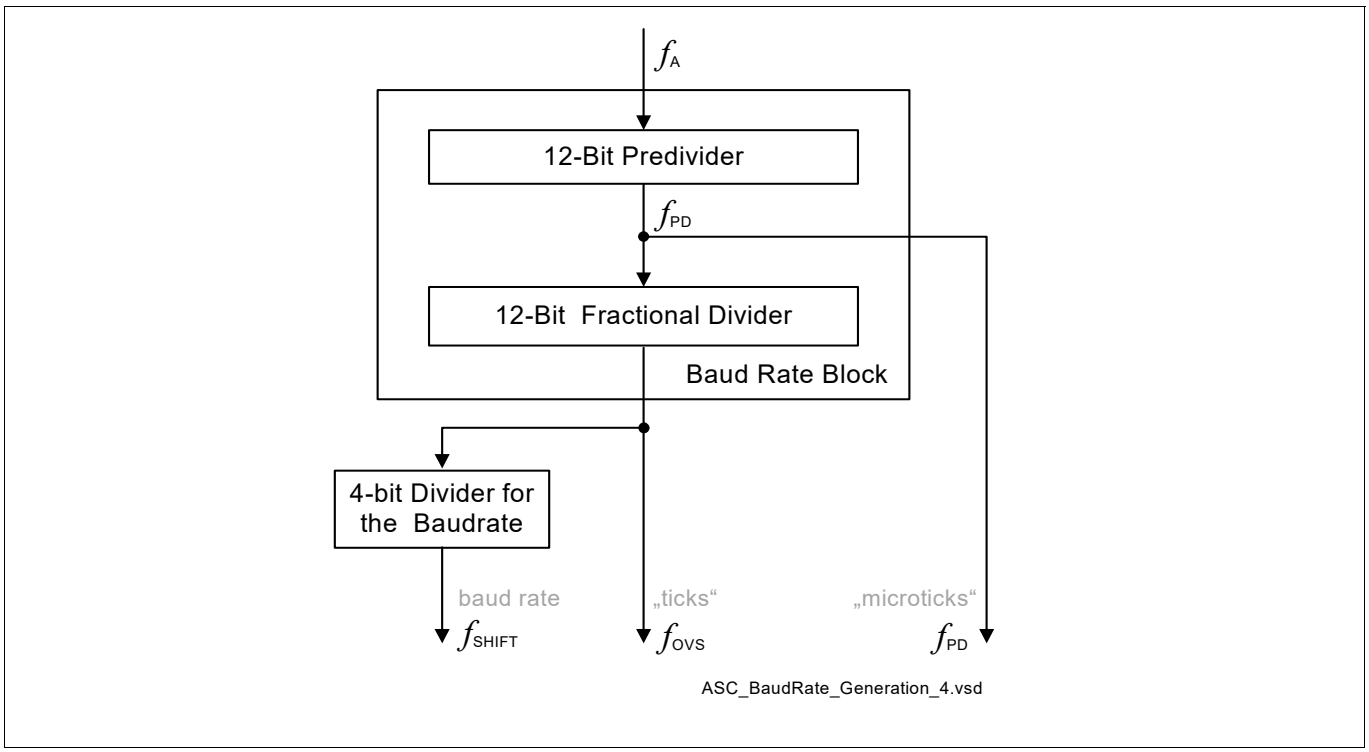


Figure 435 Baud Rate Generation

The following bit fields are available for configuring of the 28-bit baud rate divider chain:

- BITCON.PRESCALER (**Bit Configuration Register**) - the division ratio of the pre-divider
- BRG.NUMERATOR (**Baud Rate Generation Register**) - the numerator of the fractional divider
- BRG.DENOMINATOR (**Baud Rate Generation Register**) - the denominator of the fractional divider
- BITCON.OVERSAMPLING (**Bit Configuration Register**) - the division ratio of the baudrate post divider

The chain of the generated frequencies from f_A down to the f_{SHIFT} (the baudrate) is calculated as follows:

$f_{PD} = f_A$ / (BITCON.PRESCALER + 1)

$f_{OVS} = f_{PD}$ * BRG.NUMERATOR / BRG.DENOMINATOR,

$f_{SHIFT} = f_{OVS}$ / (BITCON.OVERSAMPLING + 1)

*Note: Fractional division requires that BRG.NUMERATOR is less or equal than the BRG.DENOMINATOR. (**Baud Rate Generation Register**)*

The overall formula is given as follows:

$$BAUDRATE = \frac{f_A \times NUMERATOR}{(PRESCALER + 1) \times DENOMINATOR \times (OVERSAMPLING + 1)} \quad (36.1)$$

36.3.3.2 Bit Timing Properties

The ASCLIN module provides flexible programming of the bit oversampling, sampling point and input signal filtering properties.

The oversampling factor for the incoming bit-stream is configurable from 4 to 16 ticks (or time quanta) per bit.

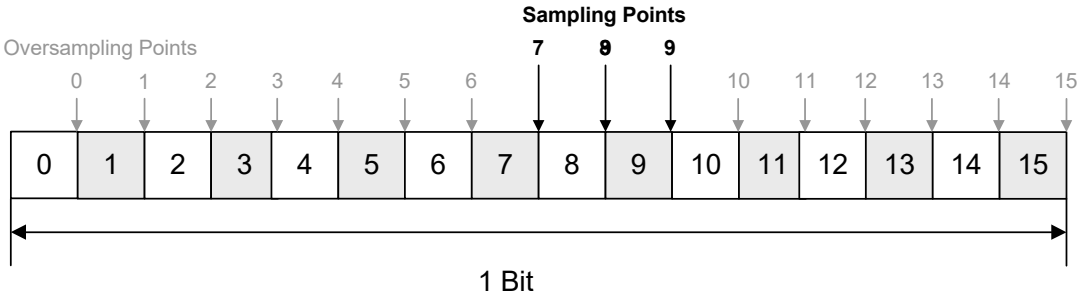
At the same time, using the oversampling frequency, a digital median filter can be enabled to filter the incoming bit stream. The filtering uses the standard majority out of three procedure. If the filter is disabled, then each bit is sampled only once.

The sampling point is also configurable and should be used in conjunction with the oversampling factor. One standard setting is 16x oversampling and using the samples 7, 8 and 9 for the data. Another possible setting could be 8x oversampling and using the samples 3, 4, and 5 for data.

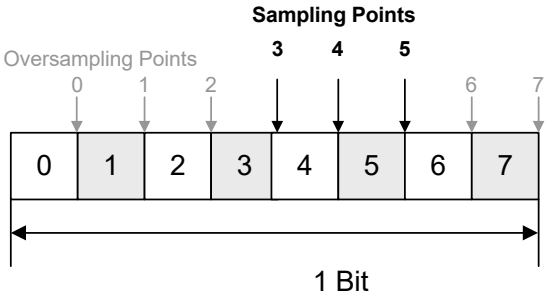
The following bitfields are available for configuring the bit properties:

- BITCON.PRESCALER (**Bit Configuration Register**) - the twelve bit integer divider defining the microtick used by the fractional divider to generate the baud rate, and by the digital filter for the de-glitching of the RX input signal.
- BITCON.SAMPLEPOINT (**Bit Configuration Register**) - the bit field defining the sampling point position, and the duty cycle in the SPI mode.
- BITCON.SM (**Bit Configuration Register**) - the bit enables the digital median filter (majority out of three): 1 or 3 samples per bit.
- IOCR.DEPTH (**Input and Output Control Register**) - the bitfield defining the floating average filter depth: off or 1 to 63 microticks
- BITCON.OVERSAMPLING (**Bit Configuration Register**) - the bitfield defining the number of ticks per bit, in the range of 4 to 16. This is a post-divider located after the fractional divider.

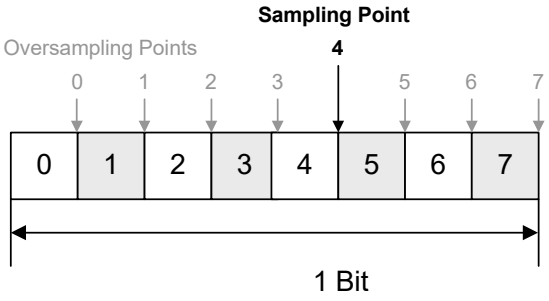
16x Oversampling, 3 sample points, relevant sample position 8



8x Oversampling, 3 sample points, relevant sample position 4



8x Oversampling, 1 sample point, relevant sample position 4



16x Oversampling, 1 sample point, relevant sample position 12

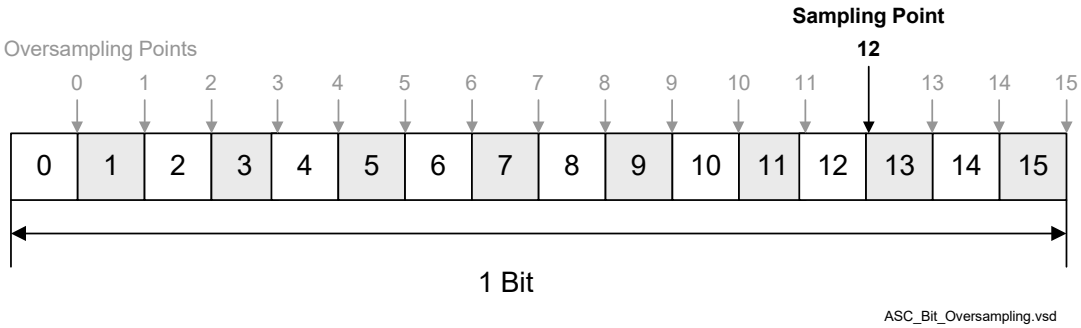


Figure 436 ASCLIN Bit Structure

Generally, the sampling point should be placed in the middle of the bit. This position is optimal in case the baud rate (the oscillator frequency) is not very precise and stable.

If the oscillator precision is very high, which is usually the case when two microcontrollers driven by quartz oscillators communicate, but the signal edges are very unsymmetrical, which is the case if open drain half-duplex connection is used, it can be of advantage to move the sample point somewhere in the second half of the bit. Open drain connection usually causes the “0” bits to be longer than the “1” bits. In such a case, optimizing the sampling point would mean placing it in the middle of the shorter “1” bit.

At the end, different combinations of oscillator precision, asymmetry of the edges, and loop-delays for collision detection result in different optimal positions of the sampling point.

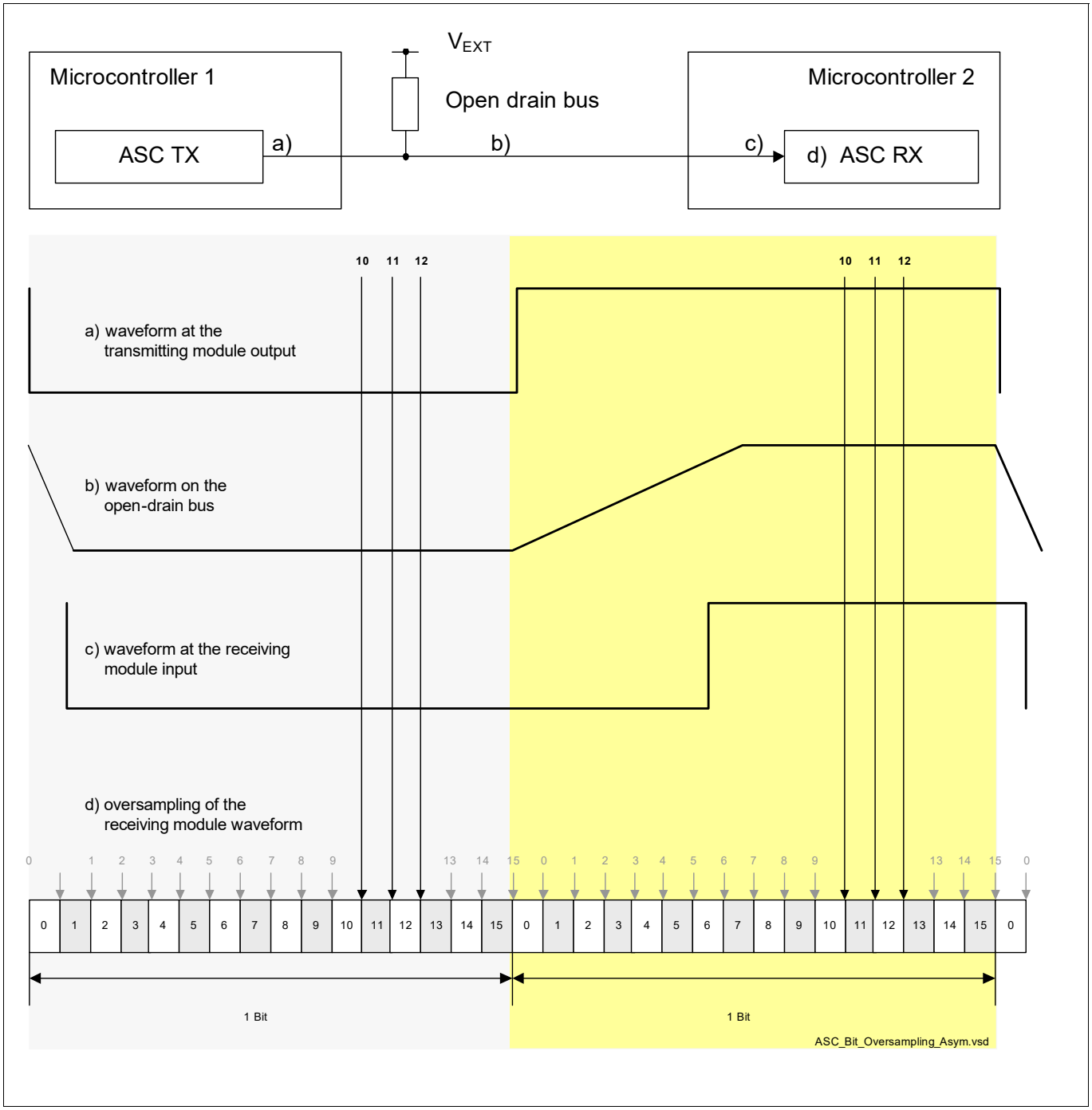


Figure 437 Bit Length Asymmetry and the Sampling Point

36.3.4 Data Frame Configuration

Applicable as transmitter and receiver, the parity scheme is configured in the bit fields FRAMECON.ODD (**Frame Control Register**), the data length in the bit fields DATCON.DATLEN (**Data Configuration Register**) and the stop bits in the bit fields FRAMECON.STOP.

36.3.5 Miscellaneous Configuration

Loop - back

36.3.6 Synchronous Mode

In synchronous mode the module supports the SPI setting of shift edge first, than the latch edge, see **Figure 438**. The module is set in synchronous mode by using the bit field FRAMECON.MODE (**Frame Control Register**).

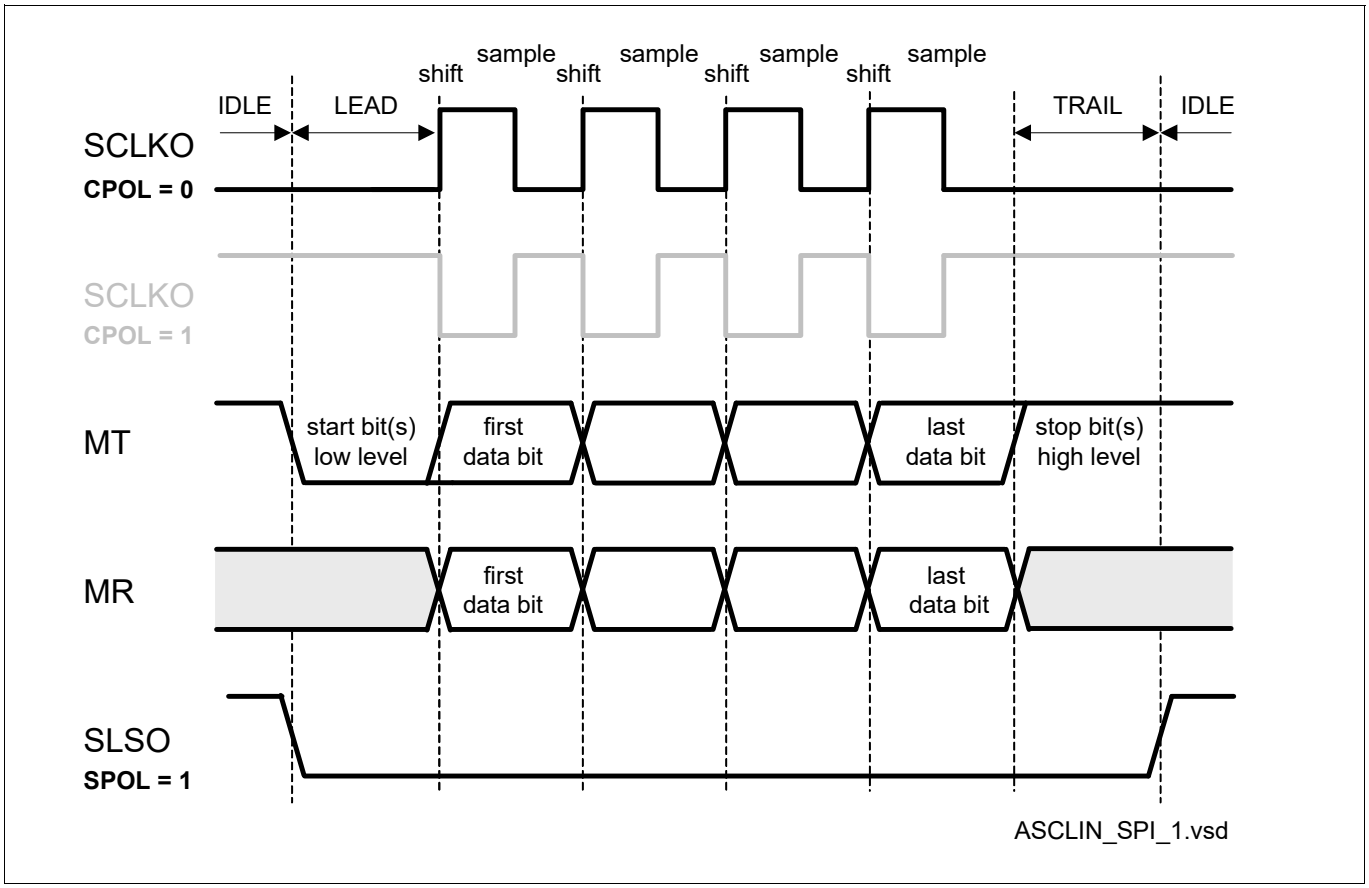


Figure 438 SPI Timing

36.3.6.1 Baud Rate and Clock Generation

The baud rate and clock generation in the synchronous mode uses generally the same counters and principles as in the asynchronous mode. It uses the same prescaler, fractional divider, and oversampling divider with configurable sample point. The only difference is that the shift clock is driven as output signal at the pin SCLKO. If a symmetrical shift clock is required, the oversampling ratio should be an even number, and the sampling point should be set to one half of the oversampling ratio (see the **Bit Configuration Register**).The clock polarity is configured in the bit IOCR.CPOL (**Input and Output Control Register**).

36.3.6.2 Data Frame Configuration

The leading and trailing delays are configured in the bit fields FRAMECON.LEAD (**Frame Control Register**) and FRAMECON.STOP. The IDLE phase duration is configured using the bit field FRAMECON.IDLE. The data length of 2 to 16 bit is defined in the bit field DATCON.DATLEN (**Data Configuration Register**).

36.3.6.3 Slave Selects Configuration

The SPI master activates automatically the slave select output signal for each data word. The polarity of the slave select can be configured by using IOCR.SPOL (**Input and Output Control Register**).

36.3.6.4 Miscellaneous Configuration

Loop - back

36.3.7 LIN Support

The ASCLIN module provides hardware support for the LIN protocol.

It supports all four elementary LIN transactions:

- TxH - Transmission of Header
- TxR - Transmission of Response
- RxH - Reception of Header
- RxR - Reception of Response

By supporting additionally the combinations of these elementary transactions, the module actually supports all LIN use cases: sending and receiving headers and responses as

- LIN master or
- LIN slave

A LIN master is engaged in three elementary transactions: TxH, TxR, RxR. It never engages in RxH, because a master never receives a header, it only transmits a header.

A LIN slave engages also in three elementary transactions: RxH, TxR, RxR. It never engages in TxH, because a slave never transmits a header, it only receives a header.

Each elementary transaction needs some hardware resources in order to be completed with minimum CPU intervention. Here is a list of tasks per transaction, supported by hardware, and the required hardware resources:

- TxH - Transmission of Header - master mode only
  - Break generation: 6-bit bit-field defining the break length in units of bits
  - Sync-field generation: hard coded 55H byte
  - ID transmission with interrupt generation
- TxR - Transmission of Response - master and slave mode
  - Number of bytes parameter: bit-field of length 4
  - Checksum generation: hardware engine, supporting classic and enhanced checksum, which can be enabled or disabled
- RxH - Reception of Header - slave mode only
  - Optional auto-baud detection: fractional divider with programmable nominator and denominator
  - Number of bytes parameter: bit-field of length 4
  - Interrupt at end of header (necessary to set the number of bytes for the RxR phase)
  - Timeout on overflow: 8-bit timer

  - Break detection:8 bit timer with programmable threshold in units of bits
- RxR - Reception of Response - master and slave mode
  - Number of bytes parameter: bit-field of length 4
  - Checksum detection: hardware that supports classic and enhanced checksum, which can be enabled or disabled; a checksum error is flagged, and an error interrupt can be triggered, if enabled
  - Checksum: the received checksum is optionally delivered in the RXFIFO
  - Timeout on overflow: 8-bit timer

The break detection feature is always active.

Wake-up signal generation in both slave and master mode.

For many LIN configuration parameters, the hardware of the ASCLIN module provides wider ranges than the LIN standard parameters. Therefore the application software shall take care that appropriate LIN standard values are used to configure the module. Such configuration parameters are:

- Break length
- Data width
- Break threshold
- Wake-up length
- Header, frame and response timeout
- Idle time

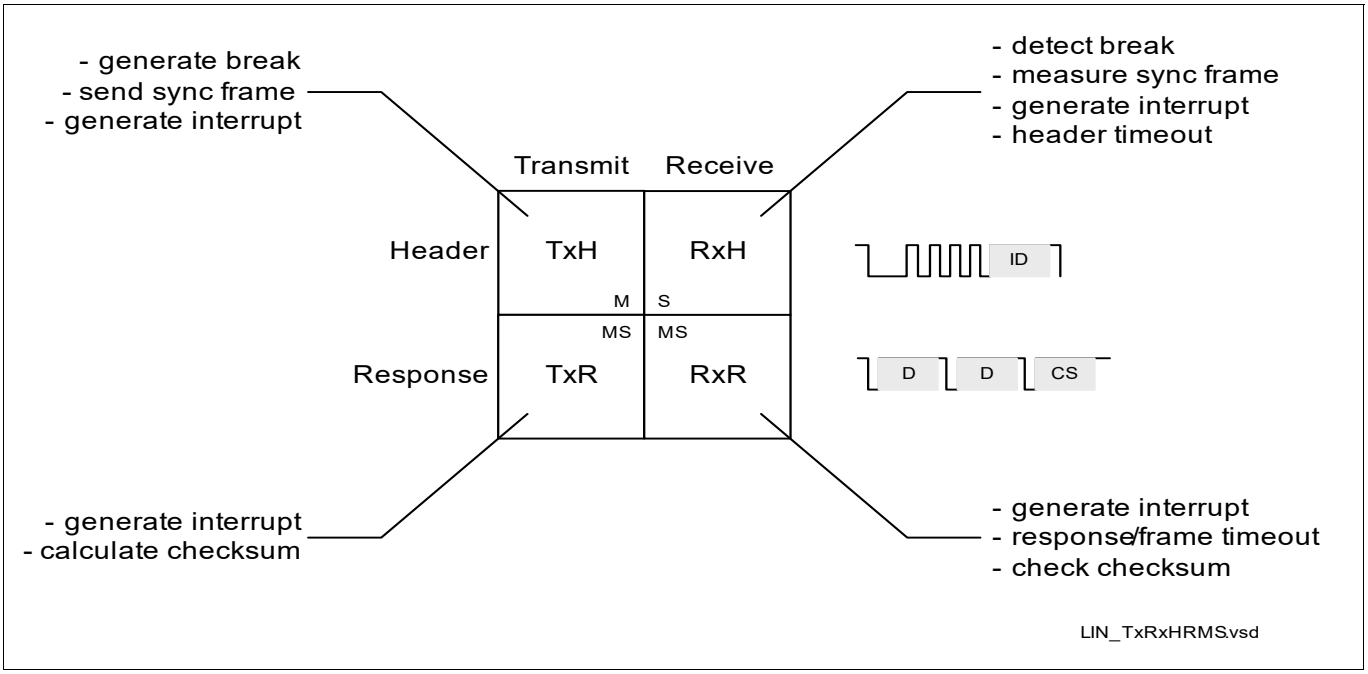


Figure 439 Overview of the elementary LIN transactions

36.3.7.1 LIN Watchdog

The LIN watchdog monitors the duration of the header, the frame or the response, and appearance of break pulses. It checks against the pre-defined time limits.
If the limits are violated, timeout interrupts are generated.

The LIN watchdog is necessary in slave mode.

The wake pulse generation is performed using the shift register and an appropriate data byte containing several consecutive zero bits. The wake pulse detection is done using falling edge detection.

For monitoring the bus for long idle or zero states there are flags for rising and falling edge available. These flags can be polled with some appropriate time raster (in microseconds or milliseconds range).

The header timeout value is known at module initialization time and remains constant for all frames.

The frame or response timeout value depends on the length of the response (1 to 8 bytes) and must be set by software depending on the received ID, after the header has been received. The initial value of this timeout should be set by software to the maximum allowed by the timer, that is 256.

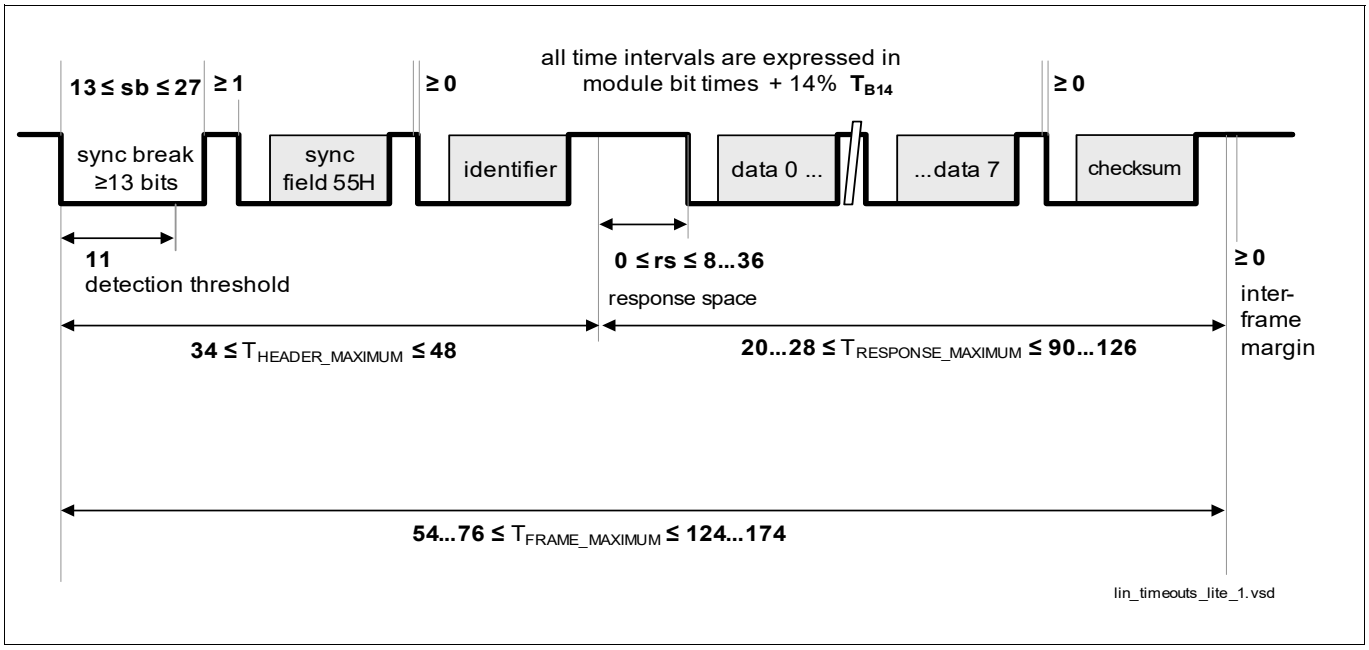


Figure 440 Duration of the Elements of a LIN Frame

The module provides timer blocks running in parallel, performing watchdog functions on specific timing requirements of the LIN protocol.

36.3.7.1.1 LIN Break, Wake, Stuck Handling

This subsection describes:

- the monitoring of the LIN Bus
- Break Pulse detection and generation and
- Wake Pulse detection and generation.

Monitoring the Bus

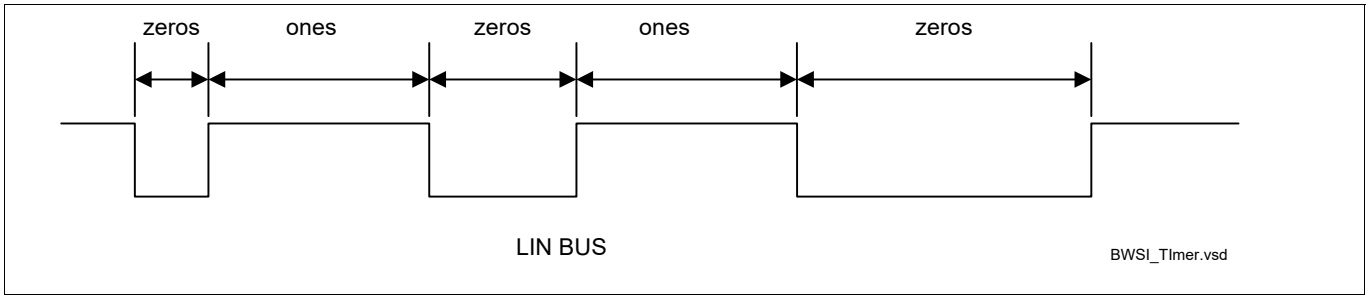


Figure 441 LIN Bus View as Sequence of Sequences of Zeros and Ones

Monitoring the bus for long idle periods (or stuck periods) is supported by providing the bit fields FLAGS.RED (**Flags Register**) and FLAGS.FED. Polling these fields periodically within some time raster defined by the operating system or timer for example each 1 ms or each 10 ms can be used for detecting very long inactive periods in the range, for example, of 150 ms to 10 s. Additionally, an interrupt on edge can be enabled by using the corresponding enable bits in the FLAGSENABLE (**Flags Enable Register**) registers. Setting and clearing the flags can be done by software by using the FLAGSSET (**Flags Set Register**) and FLAGSCLEAR (**Flags Clear Register**) register.

Note: The low break or wake up pulses do not cause a dummy byte to be transferred to the receive FIFO and do not cause a frame violation error interrupt. After the pulse the shift register goes into initial state and waits for the falling edge of the start bit of the next frame.

Break and Wake Pulse Detection

The detection of the break pulse, which is defined as low pulse with a minimum duration of 13 bit times, is performed using a threshold of either 10 or 11 bit times. The break detection threshold is set using the bitfield LINBTIMER.BREAK (**LIN Break Timer Register**).

Detecting a break pulse anywhere in the frame resets the LIN state machine to the “Break Delimiter Detected” state BDD and the whole sequence inclusive the watchdog timeout counting starts from the beginning.

Note: Detecting a break pulse anywhere in the frame could result in setting the Frame Error Flag (FE)

Wake low-pulse detection is done by monitoring the bus for a falling edge in sleep mode. If an early wake-up shall be suppressed, the parameter IOCR.DEPTH (glitch filter) (**Input and Output Control Register**) may be used.

Break and Wake Pulse Generation

The generated break low-pulse duration is defined by the bit field LINBTIMER.BREAK (**LIN Break Timer Register**) in the range of 1 to 64 bits.

Wake low-pulse is nominal 5 bit times long, but can be set to any value between 1 and 9, by writing the appropriate character in the TXFIFO and requesting its transmission as the wake pulse by setting FLAGS.TWRQ (**Flags Register**).

Note: Injecting the low pulse disables the reception with the shift register, so that this low pulse is not detected or treated as a normal ASC frame.

36.3.7.1.2 LIN Header and Response Timers

The LIN header and response times can be monitored separately.

- Header duration measurement
  - In master mode starting point of the time measurement is falling edge of the break pulse.
  - In slave mode starting point of the time measurement is the moment of detecting that there is a break pulse going on, that is the moment of detection of a low pulse longer than either 10 or 11 bit times, as configured in the LINBTIMER (**LIN Break Timer Register**) register.
- Response duration measurement
  - In master and slave mode the response duration measurement starts with the end of the header and ends with the end of the response.

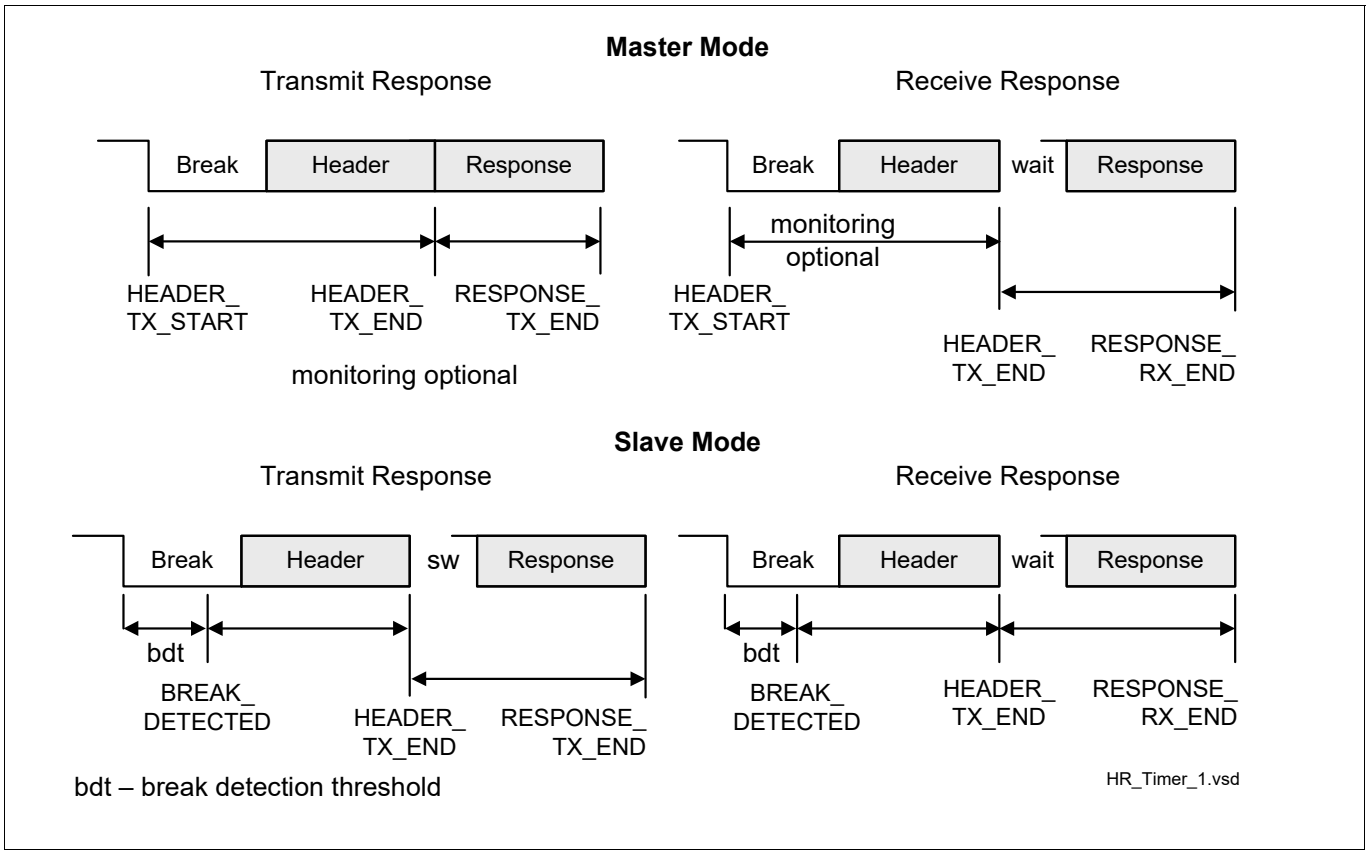


Figure 442 Duration Measurement of the Header and Response

Note: Transmitting master node can optionally monitor its own header or response timeouts, in order to detect some error conditions, like TXFIFO not containing the ID or data to be transmitted. However, these error conditions can be detected also in other ways.

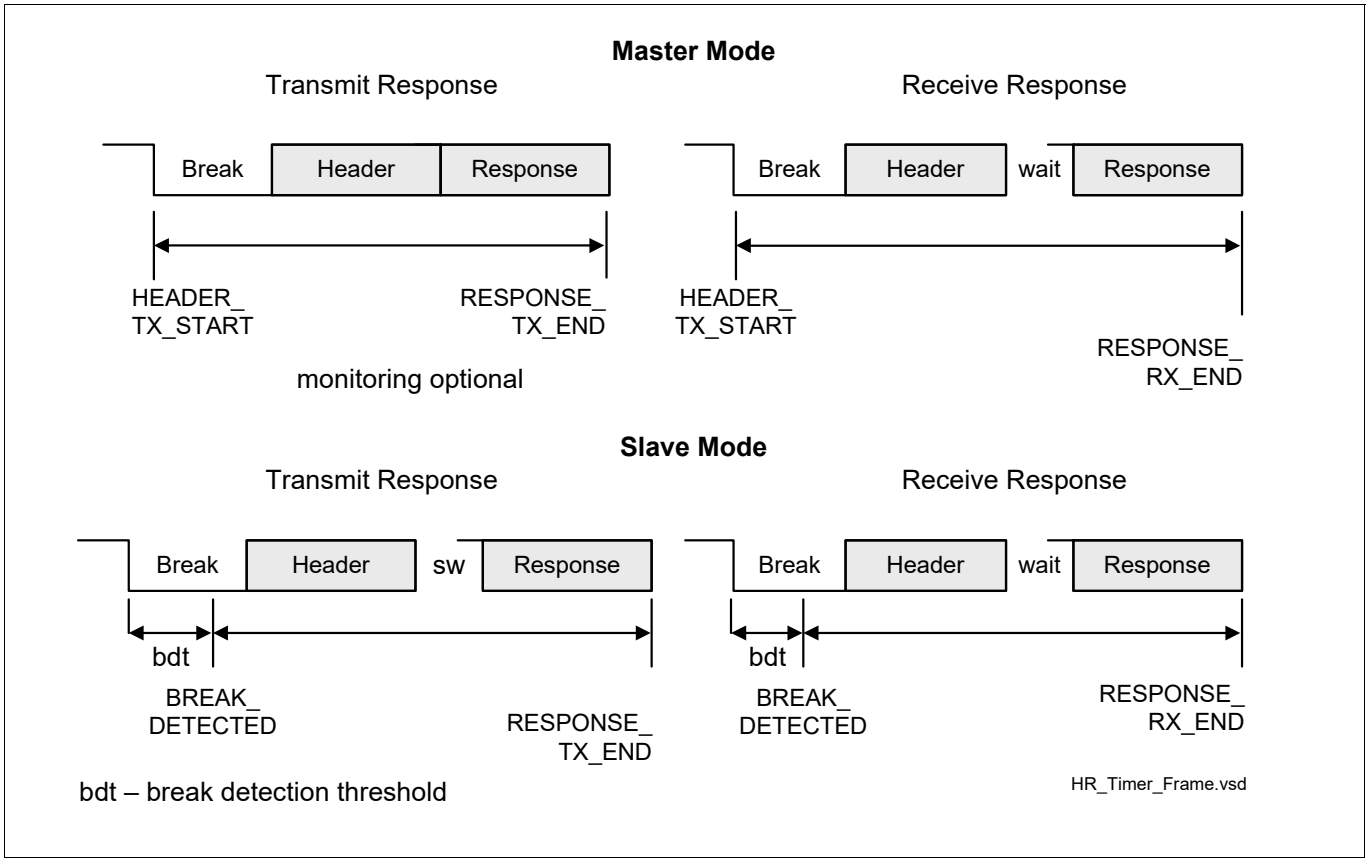


Figure 443 Duration Measurement of the Response or Frame

According to the setting of the bit DATCON.RM (**Data Configuration Register**), the DATCON.RESPONSE bit field defines response or frame duration threshold.

36.3.7.2 LIN Master Sequences

The sequences described below are examples how the elementary LIN transactions can be executed. There exist other ways to do them, for example by using other events to control the protocol flow: fifo level events instead of end of header / response events. Some alternatives are indicated in the lists below with brackets.

Initialize the module in LIN master mode

- Deactivate the clock source : **CSR**.CLKSEL = 0 (off)
- Wait or poll for **CSR**.CON = 0 (see also Clock Reconfiguration)
- Enter the INIT mode : **FRAMECON**.MODE = INIT
- Activate the clock source : set **CSR**.CLKSEL
- Wait or poll for **CSR**.CON = 1
- Deactivate the clock source : **CSR**.CLKSEL = 0 (off)
- Wait or poll for **CSR**.CON = 0
- Select the operation mode : **FRAMECON**.MODE = 3 (LIN)
- Configure the master mode : **LINCON**.MS = 1
- Configure the baud rate : **BRG**
- Configure the bit timing : **BITCON**.PRESCALER and OVERSAMPLING
- Configure the bit sampling : **IOCR**.DEPTH, **BITCON**.SM, SAMPLEPOINT
- Configure the frame parameter : **FRAMECON**.PEN=0 (no parity), STOP = 1
- Configure the break length : **LINBTIMER**.BREAK = 13 (typ)
- Configure the delay parameter : **FRAMECON**.LEAD, **FRAMECON**.IDLE
- Configure the checksum mode : **LINCON**.CSEN, **LINCON**.CSI
- Activate the clock source : set **CSR**.CLKSEL
- Wait or poll for **CSR**.CON = 1

Send only a header [master task]

- Configure the watchdog timer : **DATCON**.RESPONSE =256 (max)
- Configure the RXFIFO : **RXFIFOCON**.ENI = 0 (or 1), OUTW, BUF=0, flush
- Configure the TXFIFO : **TXFIFOCON**.ENO = 0, INW, flush
- Clear the event flags : **FLAGSCLEAR**.THC, TRC, RRC, CEC, FEC, HTC, RTC, LPC, LC, TFOC/LC, RFOC/UC/LC
- Enable the TX interrupt event : **FLAGSENABLE**.THE
- Enable the EX interrupt events : **FLAGSENABLE**.HTE, CEE, (FEE, LPE)
- Write into the TXFIFO : **TXDATA** = ID byte
- Start the header transmission : **FLAGSSET**.THRQS = 1
- Configure the TXFIFO : **TXFIFOCON**.ENO = 1

Note: If the software needs the sent ID in the RXFIFO, it should add the setting of ENI=1 to the sequence above - before triggering the header transmission. In such a case after the header has been transmitted the ID byte will be available in the RXFIFO, but not the sync byte.

React to the interrupt(s) indicating the end of transmission (and reception) of the ID byte:

- Check the error flags : **FLAGS**.HT, CE (FE, LP)

- Read from the RXFIFO : **RXDATA**=received ID, if no error has been detected

Check the transmitted / received ID to determine with which response sequence: transmit, receive or ignore to continue (same procedure as in the slave mode).

Send a response to the latest header [slave task]:

- Configure the response count : **DATCON**.DATLEN
- Configure the watchdog timer : **DATCON**.RESPONSE
- Configure the checksum mode: **DATCON**.CSM
- Configure the RXFIFO : **RXFIFOCON**.ENI =0
- Configure the TXFIFO : **TXFIFOCON**.ENO = 1
- Clear the event flags : **FLAGSCLEAR**.TRC, TFOC/UC, CEC, RTC, BDC
- Enable the TX interrupt event : **FLAGSENABLE**.TRE
- Enable the EX interrupt events: **FLAGSENABLE**.TFOE, CEE (RTE), BDE
- Write into the TXFIFO : **TXDATA** = data bytes D0, D1, … (and optionally the checksum byte if the hardware checksum generation is disabled)
- Start response transmission : **FLAGSSET**.TRRQS = 1
- The corresponding interrupts signal an error or the end of the response transmission

Receive a response to the latest header [slave task]

- Configure the response count : **DATCON**.DATLEN
- Configure the watchdog timer : **DATCON**.RESPONSE
- Configure the checksum mode: **DATCON**.CSM
- Configure the RXFIFO : **RXFIFOCON**.ENI = 1
- Clear the event flags : **FLAGSCLEAR**.RRC, RFOC/UC, FEC, LCC, RTC, FEDC, REDC, BDC
- Enable the RX interrupt event : **FLAGSENABLE**.RRE
- Enable the EX interrupt events: **FLAGSENABLE**.RFOE/UE, FEE, (LCE), RTE, BDE
- The corresponding interrupts signal that the whole response has been received (= end of the frame) or an error has occurred.

Note: If checksum injection has been enabled, the received (custom) checksum is also written into the RXFIFO and should be taken into account.

- Fetch from the RXFIFO : **RXDATA** = received data, if no error occurred

*Note: The hardware checksum check can be enabled using the bit **LINCON**.CSEN. The received checksum write to the RXFIFO can be enabled using the **LINCON**.CSI.*

Ignore the latest header [slave task]

- Configure the response timer **DATCON**.RESPONSE=0xFF
- Configure the RXFIFO : **RXFIFOCON**.ENI = 0
- Set the header only mode : **DATCON**.HO = 1
- Clear the event flags : **FLAGSCLEAR**.FEDC, REDC, BDC

The LIN master always knows if the header is followed by a response transmission (by himself) or a response reception (by a slave) or the response transmission is from slave to slave, except in case of a slave that does not respond always, but driven by internal events.

In case of addressing an always responding slave, it is recommended first to configure the header and response (transmission or reception), and afterwards to start the header transmission and optionally, at the same time, the response transmission.

To start a transaction consisting of sending a header and sending or receiving a response, which makes one whole LIN frame [master node]:

- Configure the response count : **DATCON**.DATLEN
- Configure the watchdog timer : **DATCON**.RESPONSE
- Configure the checksum mode : **DATCON**.CSM
- Configure the RXFIFO : **RXFIFOCON**.BUF = 0, flush, ENI = 0
- Configure the TXFIFO : **TXFIFOCON**.flush, ENO = 0
- Clear the interrupt event flags : **FLAGSCLEAR**.THC, TRC, RRC, CEC, FEC, HTC, RTC, LPC, LCC, TFOC/UC/LC, RFOC/UC/LC, FEDC, REDC, BDC
- Send case
  - Enable the TX interrupt event : **FLAGSENABLE**.TRE, (THE, HTE)
  - Enable the EX interrupt events: **FLAGSENABLE**.TFOE, CEE, (FEE, RTE, LPE, BDE)
- Receive case
  - Enable the RX interrupt event: **FLAGSENABLE**.RRE. (THE, HTE)
  - Enable the EX interrupt events: **FLAGSENABLE**.RFOE/UEE, CEE, FEE, RTE, (LCE, BDE)
- Write into the TXFIFO : **TXDATA** = ID byte
- Send case
  - Write to TXFIFO : **TXDATA** = data bytes (and optionally checksum byte if the hardware checksum is disabled)
- Start the header transmission : **FLAGSSET**.THRQS = 1
- Configure the TXFIFO : **TXFIFOCON**.ENO = 1

Note: If the software needs the sent ID in the RXFIFO, it should add the setting of ENI=1 to the sequence above - before triggering the header transmission. In such a case after the header has been transmitted the ID byte will be available in the RXFIFO, but not the sync byte.

- Send case
  - Start transmit response : also set **FLAGSSET**.TRRQS = 1
- The corresponding interrupt signals an error or the end of the response transmission
- Read from the RXFIFO : if no error detected, **RXDATA** = received ID
- Receive case
  - Fetch from the RXFIFO : if no error has been detected, **RXDATA** = received data bytes and optionally the checksum byte

36.3.7.3 LIN Slave Sequences

The sequences described below are examples how the elementary LIN transactions can be executed. There exist other ways to do them, for example by using other events to control the protocol flow: fifo level events instead of end of header / response events. Some alternatives are indicated in the lists below with brackets.

Initialize the module in slave mode

- Deactivate the clock source : **CSR**.CLKSEL = 0 (off)
- Wait or poll for **CSR**.CON = 0 (see also Clock Reconfiguration)
- Enter the INIT mode : **FRAMECON**.MODE = INIT
- Activate the clock source : set **CSR**.CLKSEL
- Wait or poll for **CSR**.CON = 1
- Deactivate the clock source : **CSR**.CLKSEL = 0 (off)
- Wait or poll for **CSR**.CON = 0
- Select the operation mode : **FRAMECON**.MODE = 3 (LIN)
- Configure the slave mode : **LINCON**.MS = 0
- Configure the baud rate : **BRG** , autobaud detection enable, **BRD**.UPPER / LOWERLIMIT
- Configure the bit timing : **BITCON**.PRESCALER and OVERSAMPLING
- Configure the bit sampling : **IOCR**.DEPTH , **BITCON**.SM , SAMPLEPOINT
- Configure the frame parameter : **FRAMECON**.PEN = 0 (no parity), STOP= 1
- Configure the break length : **LINBTIMER**.BREAK = 11 (typ)
- Configure the header timeout : **LINHTIMER**.HEADER
- Configure the delay parameter : **FRAMECON**.IDLE
- Configure the checksum mode : **LINCON**.CSEN, **LINCON**.CSI
- Activate the clock source : set **CSR**.CLKSEL
- Wait or poll for **CSR**.CON = 1

In slave mode, the module waits for a header from the master, that is, it waits for a break pulse followed by the sync byte and the ID.

Configure for header reception [slave task]

- Configure the watchdog timer : **DATCON**.RESPONSE =256 (max)
- Configure the RXFIFO : **RXFIFOCON** bufmode = 0, flush, ENI = 0
- Clear the interrupt flags : **FLAGSCLEAR**.RHC, TRC, RRC, FEC, HTC, RTC, BDC, LPC, LAC, LCC, TFOC/UC/LC, RFOC/UC/LC
- Enable the RX interrupt event : **FLAGSENABLE**.RHE
- Enable the EX interrupt event : **FLAGSENABLE**.HTE, CEE, FEE, LAE, LPE

React to the interrupt(s) generated after receiving the ID byte:

- Check error flags : **FLAGS**.HT, FE, LA, LP
- Read from the RXFIFO : if no error detected, **RXDATA** = received ID

Check the received ID to determine which response sequence (send, receive or ignore header) to use next.

The LIN slave does not know in advance if it will respond to the header with a transmission himself or it will receive a response by another slave. It looks up the received ID and decides if this ID is associated with a transmit or receive response or if it is don't care (it is for some other slave). This look-up and the subsequent configuration of the module must be performed by software within the allowed response time.

Send a response to the latest header [slave task]

(same sequence as in master mode)

Receive a response to the latest header [slave task]

(same sequence as in master mode)

Ignore the latest header (if the received ID contains an error or is not for this slave) [slave task]

(same sequence as in master mode)

36.3.7.4 Using the ENI and HO Bits

Both the ENI (Enable Input) and HO (Header Only) bits affect the reception of the various byte types in the LIN frame.

The HO bit is normally used in cases where the response part of a frame should be ignored; the module waits for the next break signal. The ENI bit can be set by the user software, but is also set by the hardware in slave mode after the sync byte reception, in order to enable the transfer of the ID byte of the header in the RXFIFO.

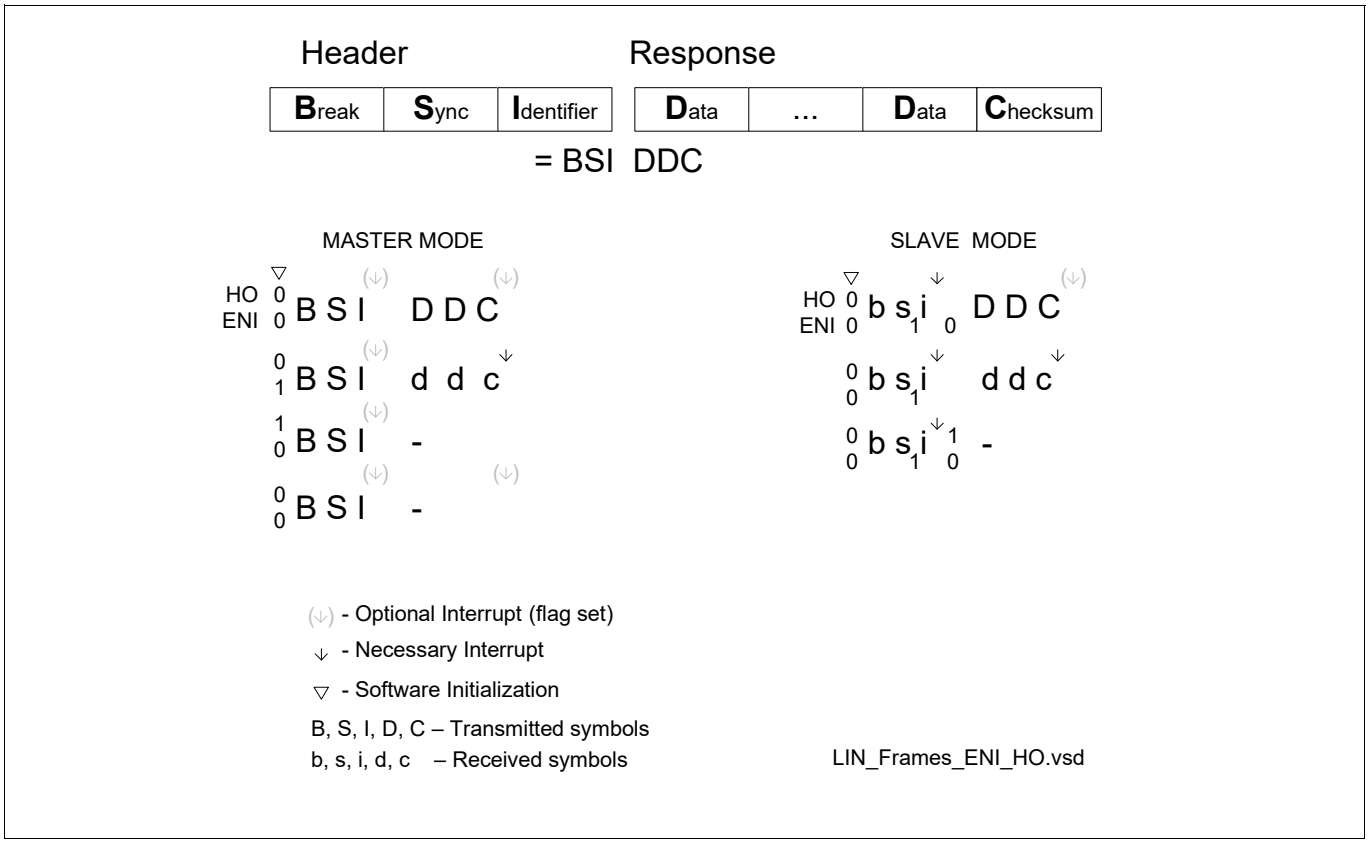


Figure 444 ENI and HO bits

36.3.7.5 LIN Error Recovery

This section describes the behavior of the module in case of errors detected during reception and during transmission of a LIN frame.

Reception Related Errors

In case of a reception error, the receive state machine goes into the state “waiting for break”, and the corresponding interrupt is triggered.

ID Parity error

Checksum error

Timeout error

Framing error

Baud Rate error

Transmission Related Errors

Collision error (LIN2.1 mandatory). If the collision detection mechanism is enabled FRAMECON.CEN (**Frame Control Register**), the frame will be aborted and the transmitter state machine goes to the idle state.

36.3.7.6 LIN Sleep and LIN Wake-Up

Wake up low pulse in duration of 250 µs to 5 ms can be generated by the module.

Wake up low pulse in duration longer than 150 µs wakes up a sleeping module.

A master node which has received a wake up pulse can start polling the slaves, it can start sending break pulses and headers, and send or receive the corresponding responses.

A slave which has been woken up shall be capable of receiving LIN headers after a wake-up time of maximum 100 ms. A slave issuing a wake-up pulse expects to receive a header within 150 ms to 250 ms after the end of the wake-up pulse. If the header does not come, the slave issues a wake up pulse again.

36.3.8 Auto Baud Rate Detection

Auto baud rate detection is active in slave mode during the reception of the sync field in the LIN header. It measures the longest time interval between two falling edges in the 55H sync field. The measured value is loaded in the denominator of the fractional divider and used afterwards for generating the baudrate for the reminder of this frame if the auto baud rate usage is enabled (LINCON.ABD = 1) (**LIN Control Register**).

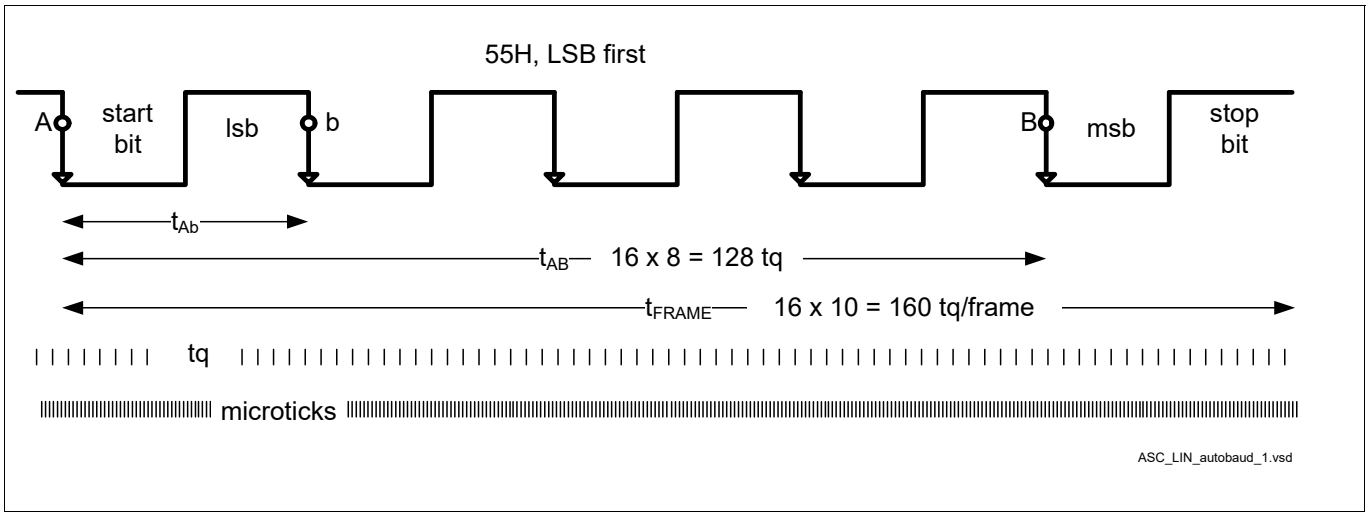


Figure 445 Measurement of the Sync field

The following bitfields are available for monitoring the autobaud detection:

- BRD.MEASURED - the measured time interval between the first and the fifth falling edge of the sync byte
- BRD.UPPERLIMIT - In case the LIN autobaud detection measures a baud rate 14% lower than the nominal one, that is it measures a time interval longer than the UPPERLIMIT, a baud rate error event is triggered that, if enabled, generates an interrupt.
- BRD.LOWERLIMIT - In case the LIN autobaud detection measures a baud rate 14% higher than the nominal one, that is it measures a time interval shorter than the LOWERLIMIT, a baud rate error event is triggered that, if enabled, generates an interrupt.

Auto Baud Rate Operation

The BRD.UPPERLIMIT defines the maximum allowed duration for 8 bits in microticks, and therefore the minimum allowed baud rate. In order to define 14% lower baud rate, 8-bit duration 16% longer than the nominal must be entered in the UPPERLIMIT bit field. This is due to the fact that BaudRate = 1 / BitTime, and UPPERLIMIT defines the time.

The BRD.LOWERLIMIT defines the minimum allowed duration for 8 bits in microticks, and therefore the maximum allowed baud rate. In order to define 14% higher baud rate, 8-bit duration 12% shorter than the nominal must be entered in the LOWERLIMIT bit field. This is due to the fact that BaudRate = 1 / BitTime, and LOWERLIMIT defines the time.

If the autobaud is activated, the fractional divider uses a numerator value of 8 x (BITCON.OVERSAMPLING + 1) (**Bit Configuration Register**) and ignores the bit field BRG.NUMERATOR (**Baud Rate Generation Register**). For the standard LIN protocol oversampling of 16 (BITCON.OVERSAMPLING = 15), the numerator value used internally is 128 and ignores the bit field BRG.NUMERATOR. Nevertheless, programming 128, or 8 x (OVERSAMPLING + 1) in BRG.NUMERATOR may increase the clarity of the software.

Regarding the denominator of the fractional divider, defined in BRG.DENOMINATOR (**Baud Rate Generation Register**), its initial value is the nominal value and is set by the application software. During the operation of the

module, the BRD.MEASURED value is automatically loaded into the denominator of the fractional divider, as long as it is within the limits.

Note: For correct behaviour of the fractional divider, It has to be ensured that the numerator value i.e., 8 X (OVERSAMPLING + 1) is less than or equal to the denominator value (BRD.MEASURED) which is calculated by the Auto Baud Detection mechanism. There is no check implemented by the hardware.

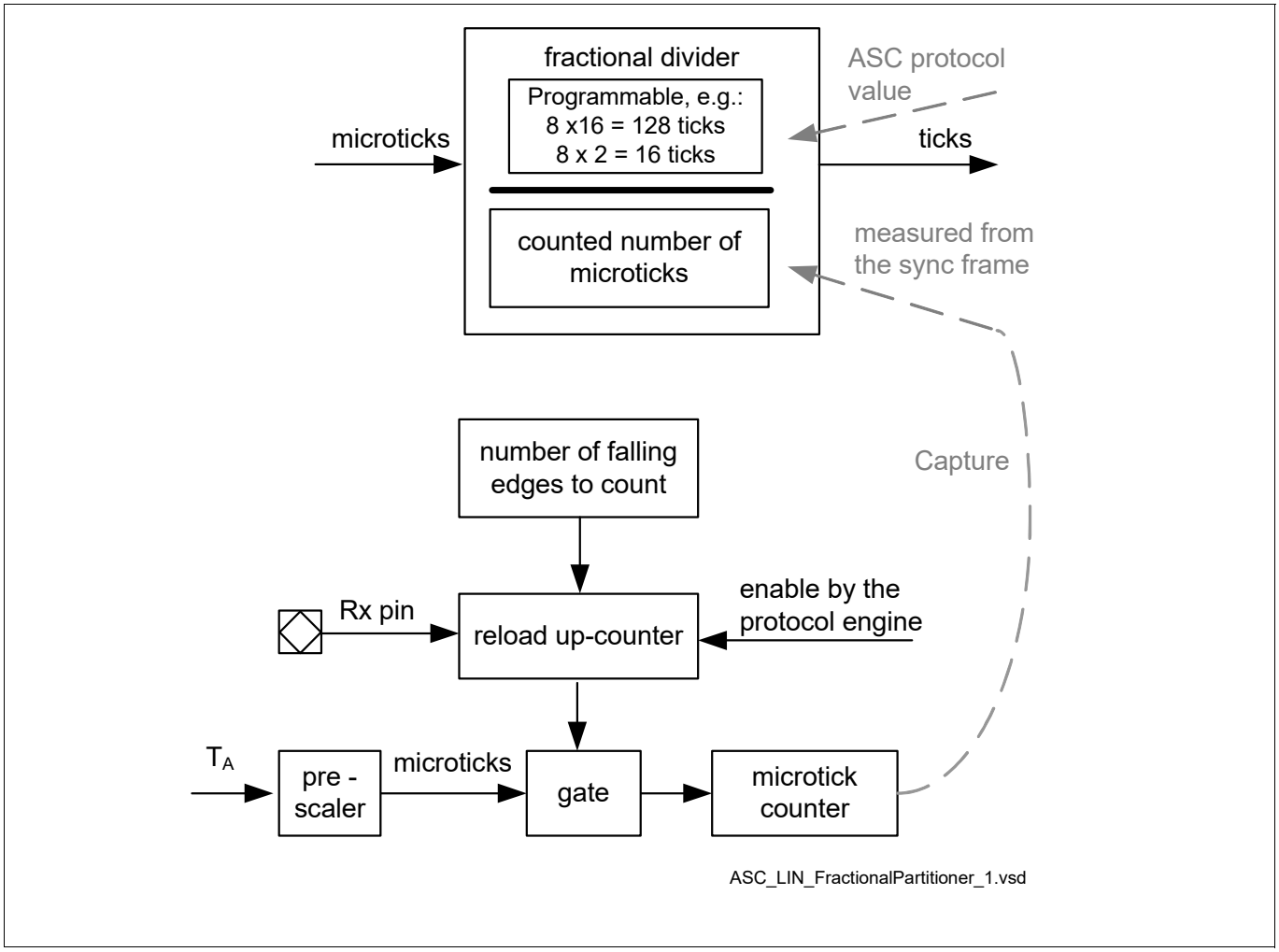


Figure 446 Overview of the LIN Auto Baud Detection Principle

Auto baud rate detection measures the time between the first and the fifth falling edge of the sync field in T_{SYS} units and loads this number in the denominator. The LIN protocol uses baud rates in a range between 2400 Baud to 19200 Baud.The expected time at 19.2 KBaud is 8 * 52.1 us = 416.8 us and the expected denominator value at T_A = 10ns is in the range of 417us / 10ns = 41 700.

The numerator operates internally with the time quanta for 8 bits, 16 times oversampling for LIN protocol: 8*16=128. The bit field BRG.NUMERATOR (**Baud Rate Generation Register**) is ignored.

36.3.9 Collision Detection

Collision detection monitors the consistency of transmitted and the echoed received bytes in LIN mode and half duplex SPI modes.

36.3.10 LIN Protocol Control

There is a central LIN protocol state machine. It is connected to the receiver and the transmitter shift registers, the Tx and Rx FIFOs, the checksum logic and the watchdog timers. The machine takes care of generating the sync byte of 55_H and the automatic handling of the checksum. Both the classic LIN V1.3 and the enhanced LIN V2.0 / V2.1 checksum are supported. The hardware checksum feature is switchable on and off, and the choice between using the classical and the enhanced checksum is done by software with the DATCON.CSM (**Data Configuration Register**) bit on a frame by frame basis. As can be seen in the **Figure 448**, for the LIN version 2.0 and 2.1, the enhanced checksum is calculated for the identifiers 0…59, and the classical checksum for the identifiers 60…63.

Additionally, the parity of the ID field is generated in master mode, and checked in slave mode. In slave mode, in case of a mismatch between the received and the calculated parity, an error interrupt is raised.

In receive case, if LINCON.CSI = 1 (**LIN Control Register**), the received checksum byte is written to the RXFIFO after the last data byte.

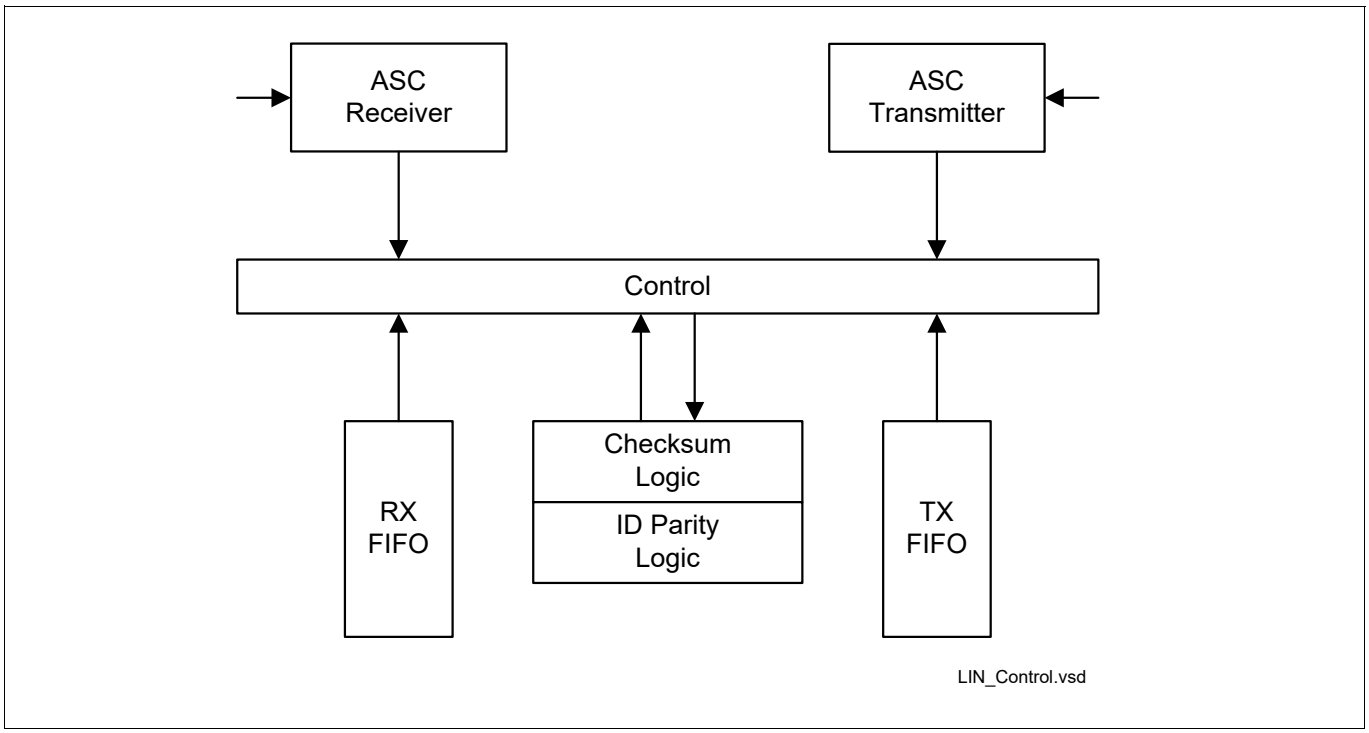


Figure 447 Block Diagram of the LIN Control Subsystem

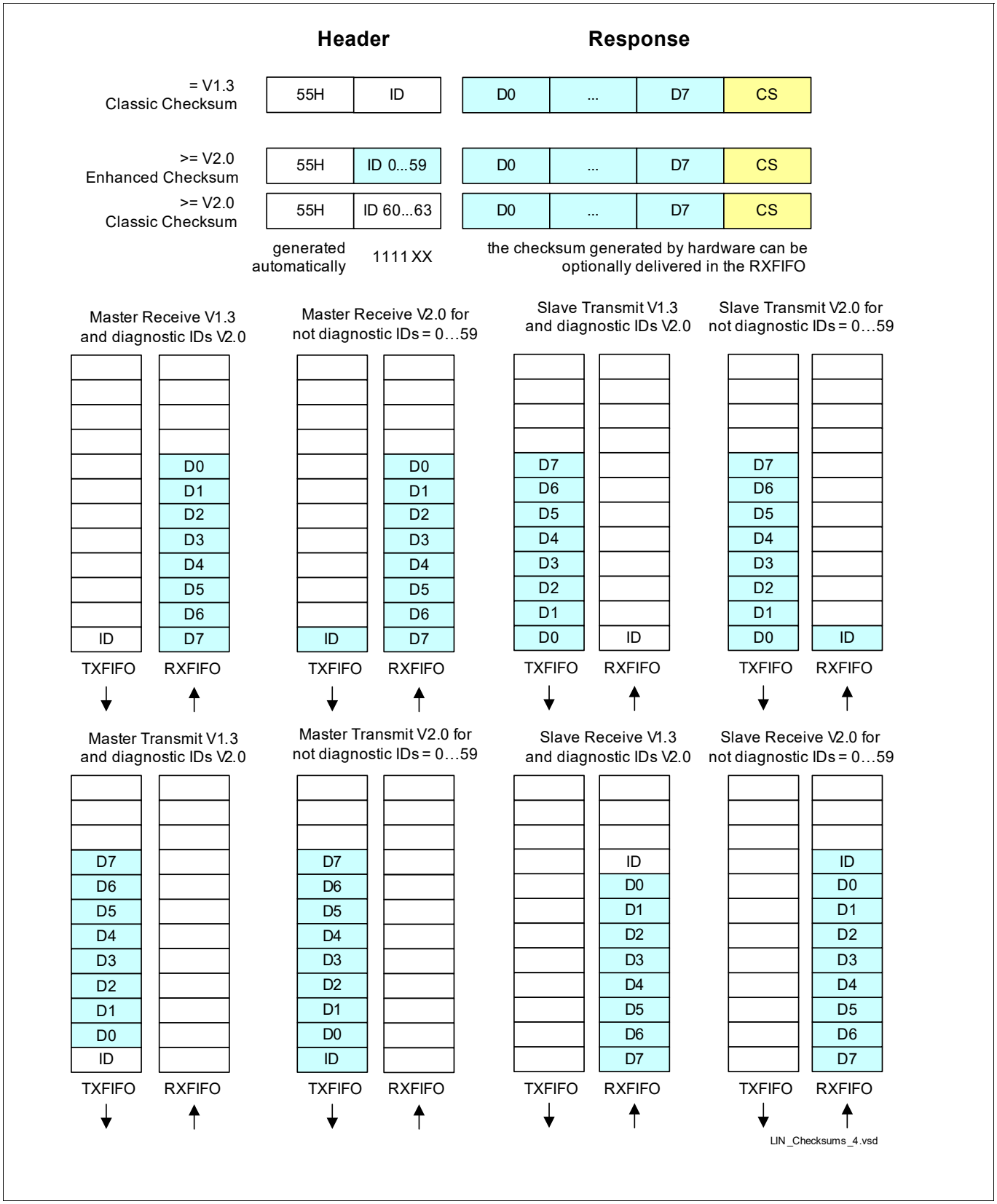


Figure 448 Coverage of the Checksum in Different Use Cases

36.3.11 Interrupts

The ASCLIN module generates three interrupts:

- TX - Transmit Interrupt
  - Signals the TxFIFO level event (FLAGS.TFL) (**Flags Register**)
- RX - Receive Interrupt
  - Signals the RxFIFO level event (FLAGS.RFL) (**Flags Register**)
- EX - Extended Error Interrupt
  - Signals every error (FLAGS.PE, FE, CE, RFO, RFU, TFO) (**Flags Register**) and
  - Some additional events (FLAGS.FED, RED)

The LIN events related to transmission and reception of header and response are mapped to the corresponding transmit, receive and extended error interrupts

- TX:
  - Transmission of the header in master mode completed (FLAGS.TH) (**Flags Register**)
  - Transmission of a response completed (FLAGS.TR)
- RX:
  - Reception of the header in a slave mode completed (FLAGS.RH) (**Flags Register**)
  - Reception of the response completed (FLAGS.RR)
- EX:
  - LIN protocol (FLAGS. BD, TC) (**Flags Register**) and
  - LIN error events (FLAGS.HT, RT, LP, LA, LC)

In order to determine which event is the source of a LIN interrupt, the FLAGS (**Flags Register**) register must be polled.

Triggering a DMA

The interrupt signals are used also as DMA trigger signals. The interrupt signals are connected to the Interrupt Router Module, which routes the interrupts either to a CPU or to a DMA. There are no separate DMA trigger signals.

Relationship between the Service Request Nodes and the Event Nodes

There are several events mapped to each service request node.

Each service request (interrupt) node contains an SRC (Service Request Control) register containing a sticky flag bit and the associated set, clear and enable bits. The SRC registers are located in the IR (Interrupt Router) module.

The service request flags in the SRC registers are set by hardware, and if the enable bit is set, will be cleared by hardware when the interrupt servicing starts.

One interrupt node may be triggered by more than one events. Each event which causes an interrupt also has a sticky flag bit and associated set, clear and enable bits. These bits are distributed in four registers: FLAGS (**Flags Register**), FLAGSSET (**Flags Set Register**), FLAGSCLEAR (**Flags Clear Register**), and FLAGSENABLE (**Flags Enable Register**) located in the ASCLIN module. Each set of four bits associated to one event builds a virtual “event node”, see **Figure 449**.

The event flags are set by hardware and if enabled, trigger an interrupt. They are not cleared by hardware. They are cleared only by software in the corresponding interrupt service routine, which usually polls the flags to find the cause for the interrupts. The event flags can be also set by software for test purposes.

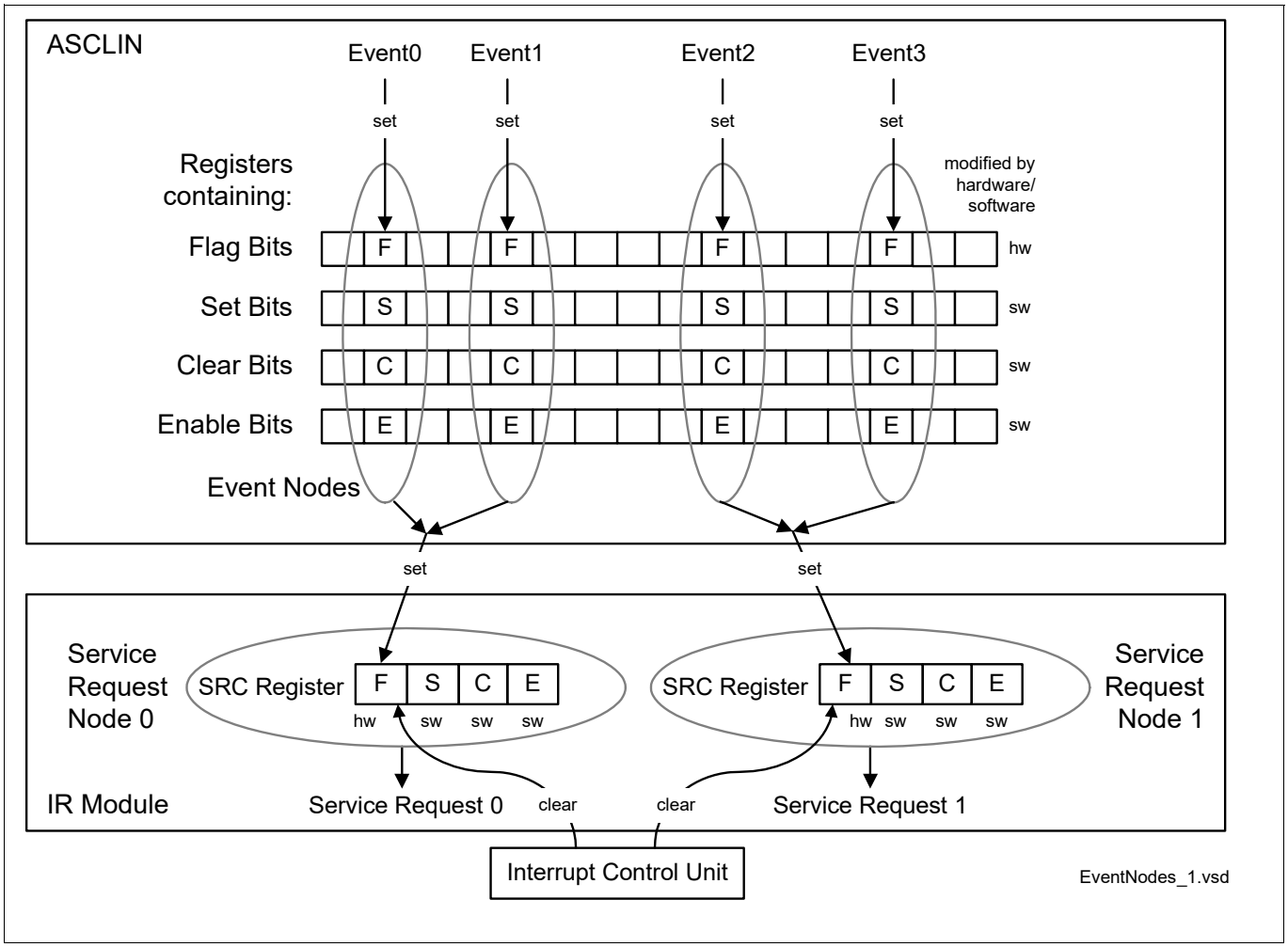


Figure 449 Relationship between Event Nodes and Service Request Nodes

Some flags do not generate interrupts and do not have enable bits. They are used either for issuing requests, which are subsequently acknowledged by hardware, or only for polling by software, see **Figure 450**.

Clearing a request / acknowledge flag by software is ignored (for example FLAGS.TRRQ, THRQ, TWRQ (**Flags Register**)).

There are no pure polled only flags in the ASCLIN module, although by letting the enable bits disabled, all interrupt event flags can be used as polled only (for example idle and stuck monitoring by using FLAGS.FED, RED (**Flags Register**)). Setting such bits per software in such a case is meaningless, except for test purposes.

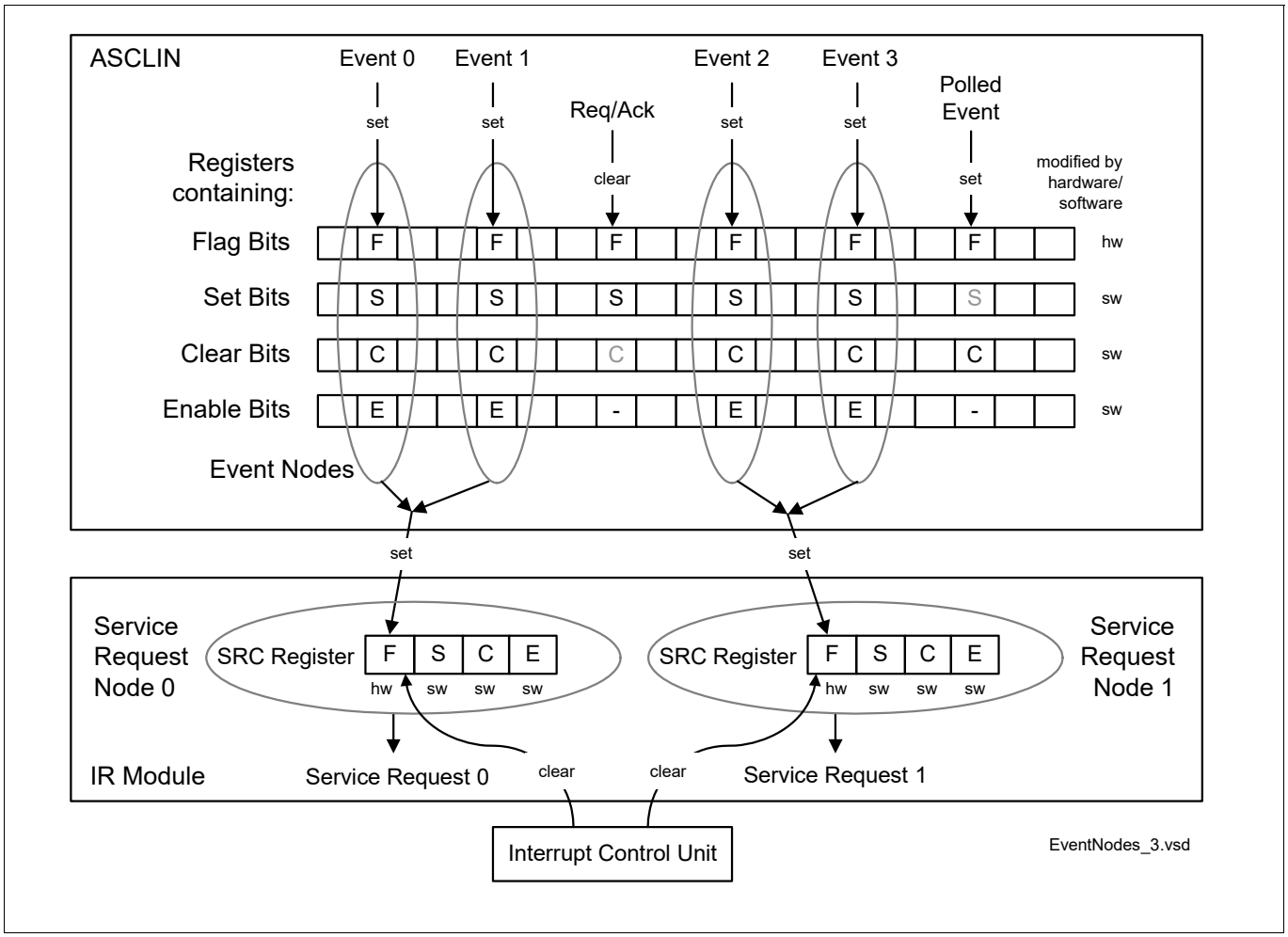


Figure 450 Acknowledge Flags and Polled Flags

36.3.12 Digital Glitch Filter

The digital glitch filter removes short glitches from the input data signal by using an increment / decrement counter with programmable threshold. On the other hand, the filter introduces a delay into the signal path, depending on the digital filter sampling frequency f_{PD} and the threshold programmed in the bit field IOCR.DEPTH (**Input and Output Control Register**). Hence the glitch filter should not be enabled in SPI mode.

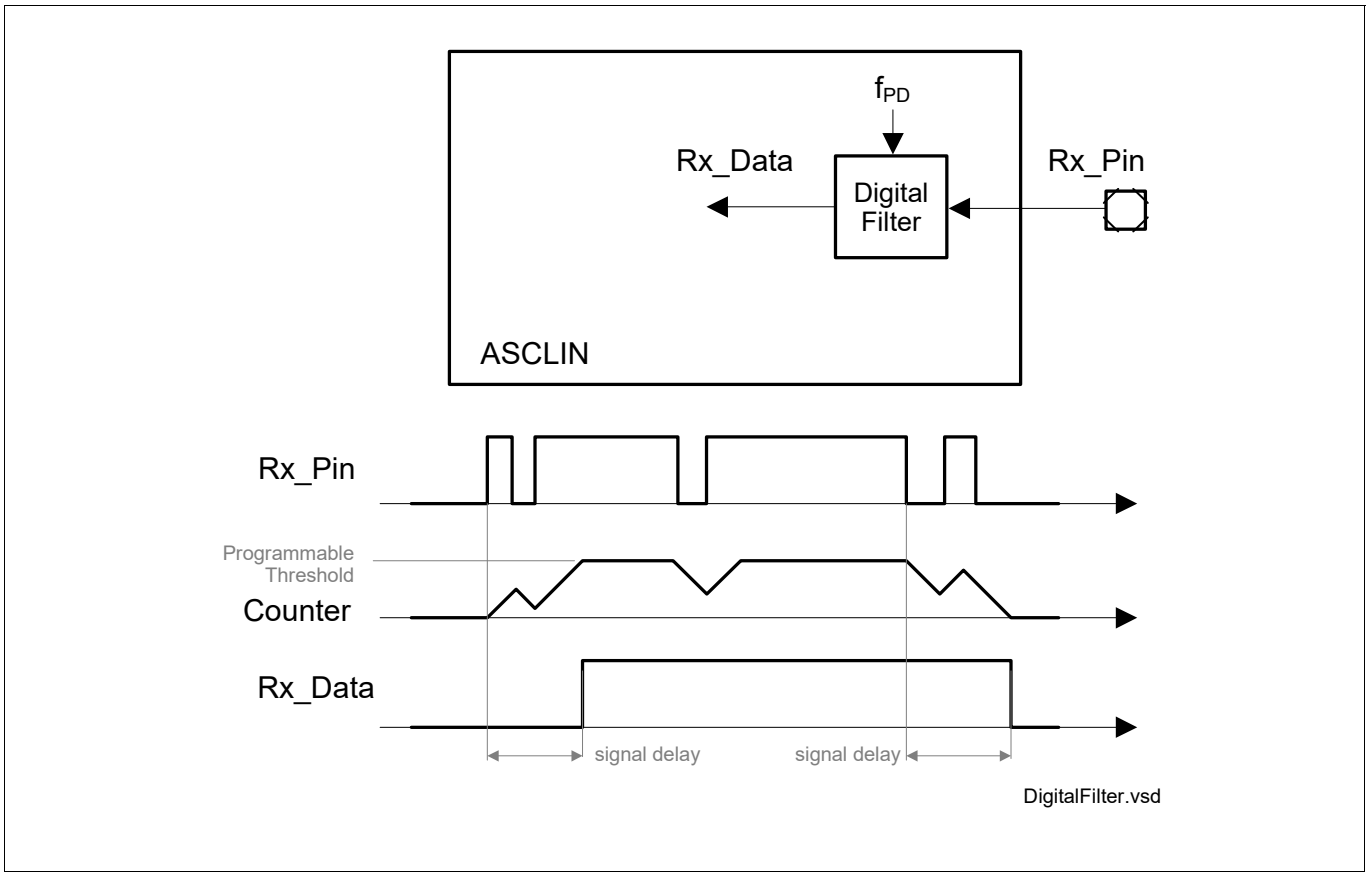


Figure 451 Digital Filter

36.3.13 Suspend, Sleep and Power-Off Behavior

Generally, the ASCLIN module transmits short data frames, not longer than 16 bits, mostly 8 bit long, sometimes with very low baud rates, down to few KBaud. In LIN mode, the data frames build LIN frames.

Simply freezing the module in the middle of an asynchronous ASC frame mostly results in an erroneous data at the receiver side, depending on the data content.

Freezing the module inside the LIN frame transmission, but between the transmission of single bytes results in a timeout error at the receiver side.

If the ASCLIN module gets a request for switching off the clock, it disables immediately all request lines towards the DMA in order to stop further DMA transfers.

In order to power down the module in a well defined way, the user software must take care that the power down request is issued when there is no ongoing single frame or LIN frame transfer and that both FIFOs are empty.

36.3.13.1 OCDS Suspend

OCDS soft suspend request suspends the module activity at the end of the current transaction. In ASC and SPI cases, this is the end of the current frame. In LIN case, this is the end of the current LIN frame (regular end of response, header timeout, or response timeout).

OCDS hard suspend request, for debugging purposes, immediately freezes the ASCLIN module in the current register state. The SPB clock feeding the kernel clock f_{CLC} is immediately switched off and the asynchronous clock f_A remains switched on.

Note: Reading and writing of registers is possible but will enable the kernel clock f_{CLC} for a few cycles. Attention: register accesses with clocking in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so an ASCLIN kernel reset might not be sufficient to bring the system into a defined state.

36.3.13.2 Sleep Mode

Going into sleep mode implies that the ASCLIN module has finished all activities and that the TX and RX FIFOs are empty. It is the responsibility of the user software to issue a sleep request in a safe time interval, where no race conditions or pipeline effects between the ASCLIN module and the DMA can occur.

The ASCLIN module behaves in the same way as in case of disable request.

36.3.13.3 Disable Request (Power-Off)

Going into power off implies that the ASCLIN module has finished all activities and that the TX and RX FIFOs are empty. It is the responsibility of the user software to issue a power-off request in a safe time interval, where no race conditions or pipeline effects between the ASCLIN module and the DMA can occur. This is no issue if the power-off procedure is one-way, that is no continuation of operation without re-initialization of the module is expected.

The ASCLIN module sets immediately all outputs to inactive, stops reacting to inputs, the internal state machines go to initial state, the registers remain unchanged, and then the module acknowledges the request. All clocks to the module are switched off.

36.3.14 Reset Behavior

There are two sources of reset:

- BPI (Bus Peripheral Interface)
- Reset bit in the kernel register KRSTx0.RST and KRSTx1.RST.

KRSTx.RST bit resets the kernel only (not the BPI registers).

BPI reset executes an Application Reset, which resets all the registers except the OCS (**OCDS Control and Status**) register. All output signals get the reset value.

The OCS register (**OCDS Control and Status**) is reset by the Debug Reset.

36.3.15 Implementation

This section describes the product specific configuration of the ASCLIN module and its interconnection with the rest of the system. The exact amount of ASCLIN modules is documented inside the appendix.

36.3.15.1 BPI_FPI Module Registers

36.3.15.1.1 System Registers

Figure 452 shows all registers associated with the BPI_FPI module, configured for one kernel.

BPI_FPI Registers Overview

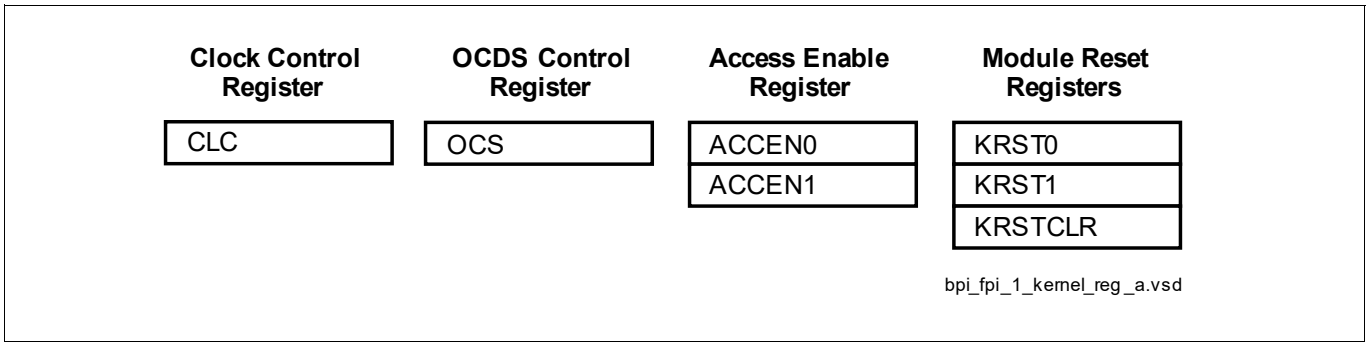


Figure 452 BPI_FPI Registers

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the f_A module clock signal, sleep mode and disable mode for the module.

CLC

Clock Control Register **(000_H)** **Application Reset Value: $0000\ 0003_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode, that is if the sensitivity of the module to the sleep signal is enabled to react to it or disabled to ignore it. 0_B Enabled 1_B Disabled
0	2, 31:4	r	**Reserved** Read as 0; must be written with 0.

OCDS Control and Status

The OCDS control and status register OCS controls the module's behavior in suspend mode (used for debugging). The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS value will not change. When OCDS is disabled the OCS suspend control is ineffective.

OCS

OCDS Control and Status **($0E8_H$)** **Debug Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		SUSSTA	SUS_P	SUS				0							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. 2_H Soft suspend **others**, Reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended

Field	Bits	Type	Description
0	23:0, 31:30	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The register ACCEN0 provides one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, … , EN31 -> TAG ID 011111B

ACCEN0

Access Enable Register 0 **(0FC$_H$)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

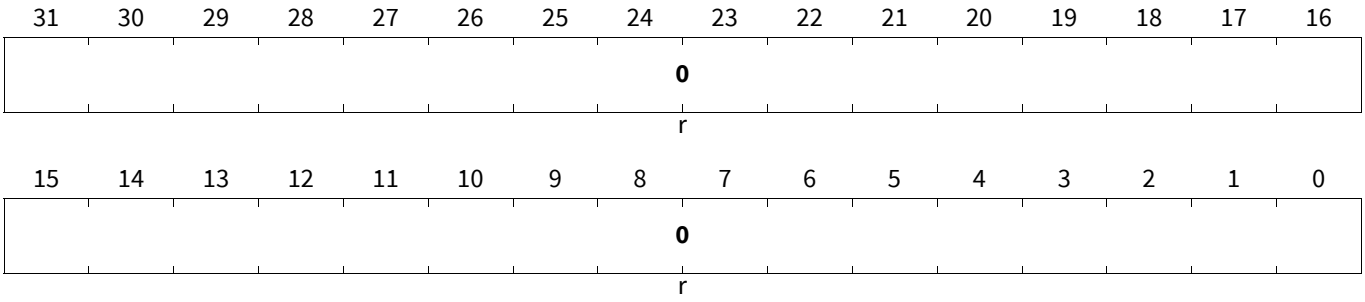
The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, …, EN31 -> TAG ID 111111B.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

ACCEN1

Access Enable Register 1 **(0F8$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0

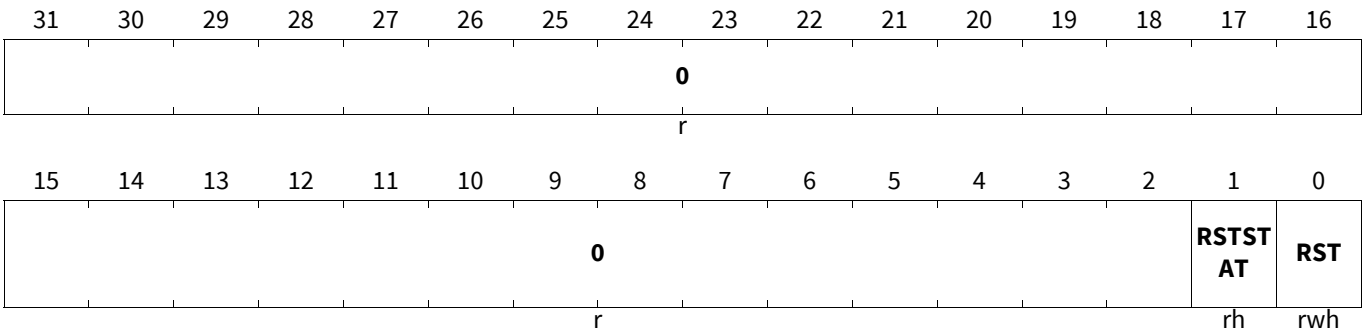
The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0

Kernel Reset Register 0 **(0F4$_H$)** **Application Reset Value: 0000 0000$_H$**



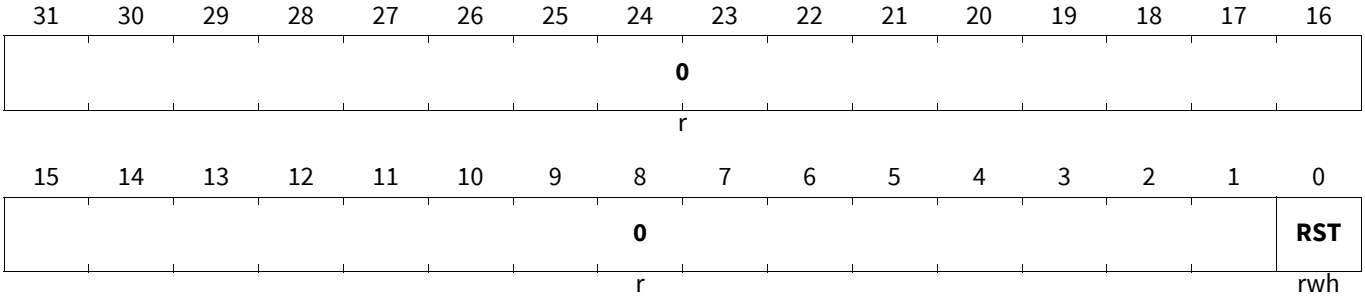
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested

Field	Bits	Type	Description
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($0F0_H$)** **Application Reset Value: $0000\ 0000_H$**



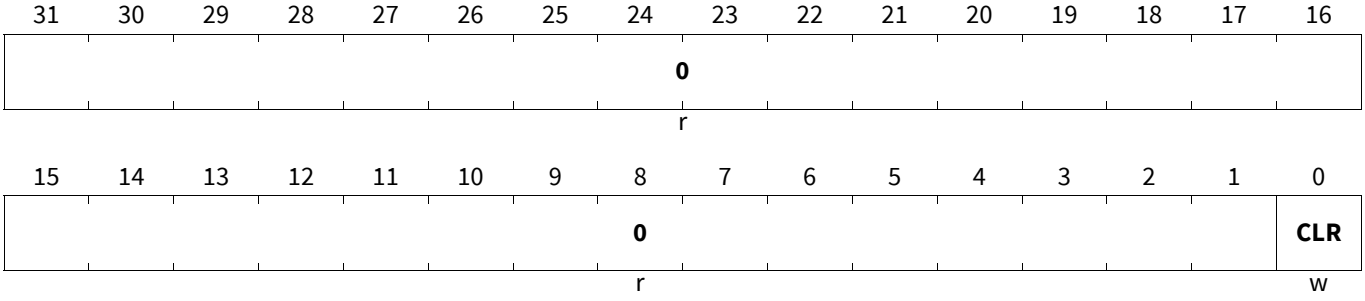
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **($0EC_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

36.3.16 On-Chip Connections

This section describes the on-chip connections of the ASCLINx module instances.

Port/Pin Connections

The connections of the ASCLIN modules to the pins / ports is described in the chapters describing the pinning and the ports.

Input signals not connected to ports/pins are connected as follows:

- The unconnected ARX signals in the lower half range (RXA to RXD) are connected to (active) low level, “0”
- The unconnected ARX signals in the upper half range (RXE to RXH) are connected to (inactive) high level, “1”
- The unconnected ACTS signals are connected to low level “0”, which is after reset the inactive level (but disabled via IOCR.CTSEN (**Input and Output Control Register**) bit field, meaning don’t care)

ASCLIN Connection to Itself

the following connection is defined for each ASCLIN instance:

ARTS -> ACTSD

This connection can be useful in the SPI mode, where frequently the transmission and the reception of data are performed in parallel. If this connection is selected by using IOCR.CTS (**Input and Output Control Register**), the TXFIFO will deliver data for transmission only if there is a free space in the RXFIFO. If not, the TXFIFO will wait for emptying of the RXFIFO by software and then automatically continue the transmission.

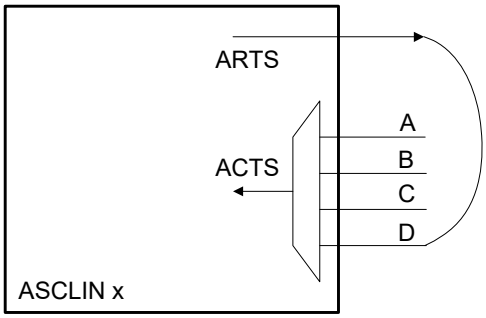


Figure 453 ASCLIN ARTS to ACTS Connection

36.3.17 ASC at CAN Support

The ASC Tx and Rx signals are overlaid with CAN Rx and Tx signal. See the Ports chapter and the Pinning chapter for the particular pin assignments.

A virgin device with a completely erased flash memory can be booted via ASC at CAN pins. See the Boot chapter for details on the implementation.

36.4 Registers

This section describes the kernel registers of the ASCLIN module. All ASCLIN kernel register names described in this section will be referenced in other parts of this device by the module name prefix “ASCLIN0_” for the ASCLIN0 interface and “ASCLIN1_” for the ASCLIN1 interface.

All registers in the ASCLIN address spaces are reset with the application reset (definition see SCU section “Reset Operation”).

ASCLIN Kernel Register Overview

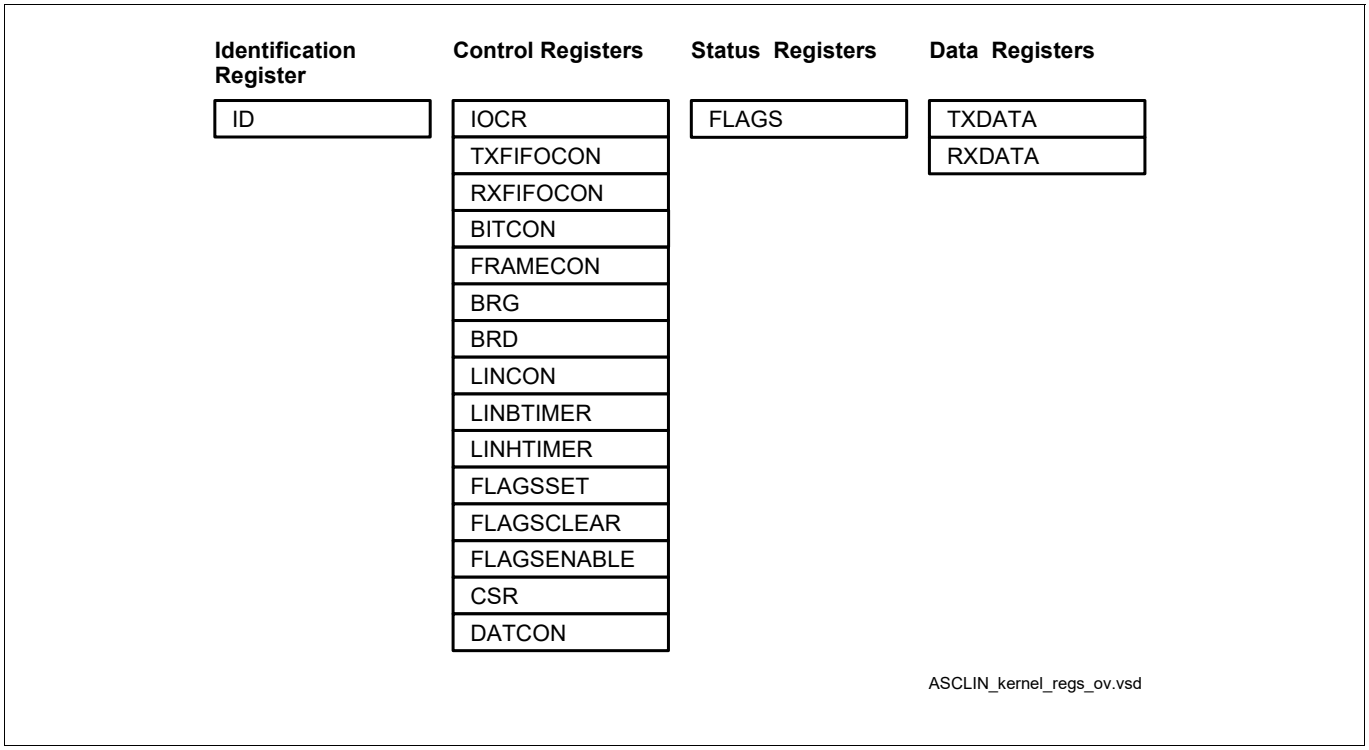


Figure 454 ASCLIN Kernel Registers

Note: Absolute Register Address
= Module Base Address (Refer Appendix)
*+ Offset Address (**Register Overview - ASCLIN (ascending Offset Address)**).*

Table 326 Register Overview - ASCLIN (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	000_H	U,SV	SV,E,P	Application Reset	50
IOCR	Input and Output Control Register	004_H	U,SV	SV,U,P	Application Reset	59
ID	Module Identification Register	008_H	U,SV	BE	Application Reset	61
TXFIFOCON	TX FIFO Configuration Register	$00C_H$	U,SV	SV,U,P	Application Reset	62

Table 326 Register Overview - ASCLIN (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
RXFIFOCON	RX FIFO Configuration Register	010_H	U,SV	SV,U,P	Application Reset	63
BITCON	Bit Configuration Register	014_H	U,SV	SV,U,P	Application Reset	65
FRAMECON	Frame Control Register	018_H	U,SV	SV,U,P	Application Reset	66
DATCON	Data Configuration Register	$01C_H$	U,SV	SV,U,P	Application Reset	68
BRG	Baud Rate Generation Register	020_H	U,SV	SV,U,P	Application Reset	69
BRD	Baud Rate Detection Register	024_H	U,SV	SV,U,P	Application Reset	70
LINCON	LIN Control Register	028_H	U,SV	SV,U,P	Application Reset	70
LINBTIMER	LIN Break Timer Register	$02C_H$	U,SV	SV,U,P	Application Reset	71
LINHTIMER	LIN Header Timer Register	030_H	U,SV	SV,U,P	Application Reset	72
FLAGS	Flags Register	034_H	U,SV	BE	Application Reset	72
FLAGSSET	Flags Set Register	038_H	U,SV	SV,U,P	Application Reset	76
FLAGSCLEAR	Flags Clear Register	$03C_H$	U,SV	SV,U,P	Application Reset	79
FLAGSENABLE	Flags Enable Register	040_H	U,SV	SV,U,P	Application Reset	82
TXDATA	Transmit Data Register	044_H	U,SV	SV,U,P	Application Reset	85
RXDATA	Receive Data Register	048_H	U,SV	BE	Application Reset	85
CSR	Clock Selection Register	$04C_H$	U,SV	SV,U,P	Application Reset	86
RXDATAD	Receive Data Debug Register	050_H	U,SV	BE	Application Reset	87
OCS	OCDS Control and Status	$0E8_H$	U,SV	SV,P,OEN	Debug Reset	51
KRSTCLR	Kernel Reset Status Clear Register	$0EC_H$	U,SV	SV,E,P	Application Reset	54
KRST1	Kernel Reset Register 1	$0F0_H$	U,SV	SV,E,P	Application Reset	54
KRST0	Kernel Reset Register 0	$0F4_H$	U,SV	SV,E,P	Application Reset	53

Table 326 Register Overview - ASCLIN (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
ACCEN1	Access Enable Register 1	$0F8_H$	U,SV	SV,SE	Application Reset	52
ACCEN0	Access Enable Register 0	$0FC_H$	U,SV	SV,SE	Application Reset	52

List of Access Protection Abbreviations

U - User Mode
SV - Supervisor Mode
BE - Bus Error
nBE - no Bus Error
P - Access Protection, as defined by the ACCEN Register
E - ENDINIT
SE - Safety ENDINIT

36.4.1 Kernel Registers

Input and Output Control Register

The Input and Output Control Register IOCRx determines several properties of the RX and TX signal path:

For the RX signal:

- The alternate input
- The filter depth

For the TX signal:

- The trigger source

Note: IOCR is only writable, if CSR.CLKSEL=0.

IOCR

Input and Output Control Register (004_H) Application Reset Value: $X000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXM	RXM	CTSEN	LB	SPOL	CPOL	RCPOL	0							CTS	
rh	rh	rw	rw	rw	rw	rw	r							rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						DEPTH						0		ALTI	
r						rw						r		rw	

Field	Bits	Type	Description
ALTI	2:0	rw	**Alternate Input Select** Selects the alternate input for the RX signal: 000_B Alternate Input A selected 001_B Alternate Input B selected 010_B Alternate Input C selected 011_B Alternate Input D selected 100_B Alternate Input E selected 101_B Alternate Input F selected 110_B Alternate Input G selected 111_B Alternate Input H selected
DEPTH	9:4	rw	**Digital Glitch Filter Depth** DEPTH determines the number of port input samples clocked with microticks that are taken into account for the calculation of the floating average. The higher the DEPTH is chosen to be, the longer the glitches that are suppressed and the longer the delay of the input signal introduced by this filter. 00_H off, default 01_H 1 … $3F_H$ 63
CTS	17:16	rw	**CTS Select** Selects the CTS input pin out of maximum four possible. 00_B ACTSA 01_B ACTSB 10_B ACTSC 11_B ACTSD
RCPOL	25	rw	**RTS CTS Polarity** RCPOL defines the active level or the RTS and CTS signals. Active means ready/clear to send. 0_B Active high 1_B Active low
CPOL	26	rw	**Clock Polarity in Synchronous Mode** CPOL defines the idle level of the clock signal if the module is set in the SPI mode. The idle level is the level outside the data transmission time intervals. Default is low level. 0_B Idle low 1_B Idle high
SPOL	27	rw	**Slave Polarity in Synchronous Mode** Defines the idle level of the SLSO signal, which is the level outside the data transmission, leading and trailing time intervals. 0_B Idle low 1_B Idle high

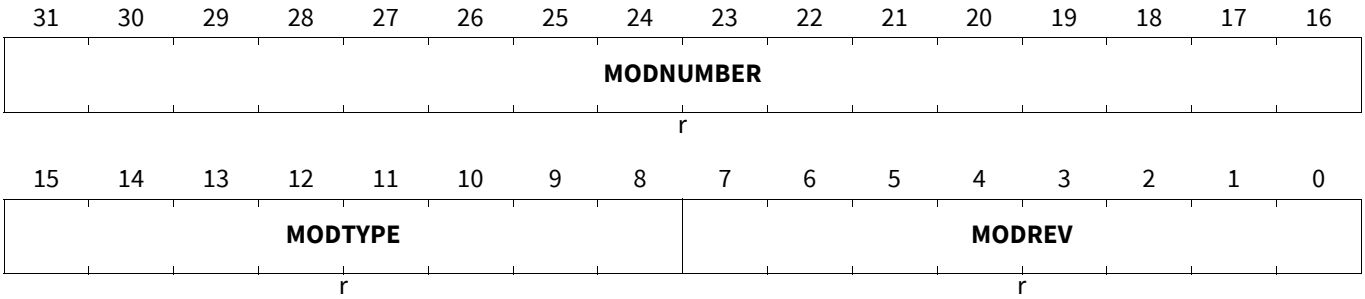
Field	Bits	Type	Description
LB	28	rw	**Loop Back Mode** Enables the in module connection of the transmit signal to receive signal. If Loop-back is enabled, the module can be run and tested without an external connection, in ASC and SPI modes. In LIN mode, loopback should not be used, because the module can be either master or slave. 0_B Disabled 1_B Enabled
CTSEN	29	rw	**Input Signal CTS Enable** Enables the sensitivity of the module to the external CTS signal. If disabled, the CTS signal is considered being permanently active. 0_B Disabled 1_B Enabled
RXM	30	rh	**Receive Monitor** Shows the status of the receive signal. 0_B Current signal is low. 1_B Current signal is high.
TXM	31	rh	**Transmit Monitor** Shows the status of the transmit signal. 0_B Current signal is low. 1_B Current signal is high.
0	3, 15:10, 24:18	r	**Reserved** Read as 0; should be written with 0.

Module Identification Register

The Module Identification Register ID contains read-only information about the module version.

ID

Module Identification Register **(008_H)** **Application Reset Value: $00C1\ C0XX_H$**



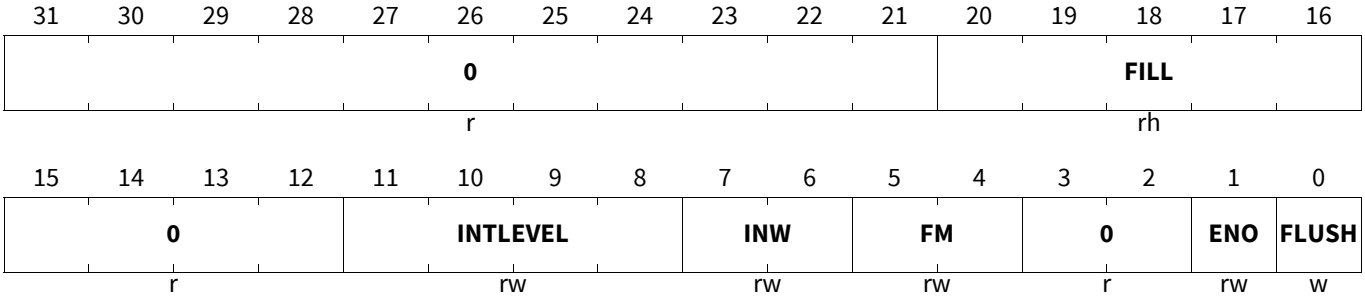
Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** MODREV defines the module revision number. The value of a module revision starts with 01_H (first revision).
MODTYPE	15:8	r	**Module Type** This bit field is $C0_H$. It defines a 32-bit module.

Field	Bits	Type	Description
MODNUMBER	31:16	r	**Module Number Value** This bit field together with MODTYPE uniquely identifies a module.

TX FIFO Configuration Register

TXFIFOCON

TX FIFO Configuration Register **($00C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FLUSH	0	w	**Flush the transmit FIFO** Write of 1 brings the Tx FIFO in empty state. Write of 0 has no effect. 0_B No effect / This value will be always read back from the Bit. 1_B Empty the Tx FIFO
ENO	1	rw	**Transmit FIFO Outlet Enable** Enables the TxFIFO outlet. In SPI and ASC modes, data transmission starts immediately when the data is available, whereas in LIN case the transmission start is controlled by the protocol engine. 0_B Disabled. In LIN case, if the protocol engine tries to fetch data. 1_B Enabled. In LIN case, no data is moved to the shift register until it is fetched by the protocol engine.
FM	5:4	rw	**TXFIFO Mode** Selects between the TXFIFO Modes. 00_B Combined Move Mode 01_B Single Move Mode 10_B Batch Move Mode 11_B reserved
INW	7:6	rw	**Transmit FIFO Inlet Width** Defines the number of bytes written to the Tx FIFO with one FPI bus write. 00_B 0 01_B 1 10_B 2 11_B 4

Field	Bits	Type	Description
INTLEVEL	11:8	rw	**FIFO Interrupt Level** Defines the filling level that triggers a re-fill interrupt or DMA access. An interrupt is generated when the filling level falls to INTLEVEL or below, each time when a data byte is taken out of the FIFO. This behavior corresponds to the Combined/Compatibility Mode of interrupt generation. See also Single Move Mode and Batch Move Mode for two additional modes. 0_H 0 … F_H 15
FILL	20:16	rh	**FIFO Filling Level** Read only bit-field containing the current filling level of the FIFO. 00_H 0 … 10_H 16 11_H Reserved … $1F_H$ Reserved
0	3:2, 15:12, 31:21	r	**Reserved** Read as 0; should be written with 0.

RX FIFO Configuration Register

RXFIFOCON

RX FIFO Configuration Register **(010_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BUF	**0**										**FILL**				
rw	r										rh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**INTLEVEL**				**OUTW**		**FM**		**0**		**ENI**	**FLUSH**
r				rw				rw		rw		r		rwh	w

Field	Bits	Type	Description
FLUSH	0	w	**Flush the receive FIFO** Write of 1 brings the Rx FIFO in empty state. Write of 0 has no effect. 0_B No effect / This value will be always read back from the module. 1_B Empty the Rx FIFO

Field	Bits	Type	Description
ENI	1	rwh	**Receive FIFO Inlet Enable** Enables the receiver and the filling of the Rx FIFO through the shift register. In LIN slave mode, this bit is set by hardware after the correct reception of the sync byte. The software can clear this bit after reception of an foreign ID in order to suppress the reception of the following response. 0_B Disabled 1_B Enabled
FM	5:4	rw	**RXFIFO Mode** Selects between the RXFIFO Modes. 00_B Combined Move Mode 01_B Single Move Mode 10_B Batch Move Mode 11_B reserved
OUTW	7:6	rw	**Receive FIFO Outlet Width** Defines the number of bytes read to the Rx FIFO with one FPI bus read. 00_B 0 01_B 1 10_B 2 11_B 4
INTLEVEL	11:8	rw	**FIFO Interrupt Level** Defines the filling level that triggers a drain interrupt or DMA access. An interrupt is generated when the filling level rises to INTLEVEL or beyond, each time when a data byte is delivered to the FIFO. This behavior corresponds to the Combined/Compatibility Mode of interrupt generation. See also Single Move Mode and Batch Move Mode for two additional modes. 0_H 1 … F_H 16
FILL	20:16	rh	**FIFO Filling Level** Read only bit-field containing the current filling level of the FIFO. 00_H 0 … 10_H 16 11_H Reserved … $1F_H$ Reserved
BUF	31	rw	**Receive Buffer Mode** If this bit is zero, then the RXFIFO behaves normally as described in this document. If this bit is set, the RXFIFO behaves as simple 32-bit one stage RX buffer, which is overwritten with each new received data. The received bits appear in the RXDATA register on the lowest bit locations. The upper locations are padded with zeros. 0_B RXFIFO 1_B Single Stage RX Buffer

Field	Bits	Type	Description
0	3:2, 15:12, 30:21	r	**Reserved** Read as 0; should be written with 0.

Bit Configuration Register

The BITCON Register defines the integer timer parameters in the baud rate generation block.
Note: The register is only writable, if CSR.CLKSEL=0.

BITCON
Bit Configuration Register **(014_H)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SM		**0**		**SAMPLEPOINT**				**0**				**OVERSAMPLING**			
rw		r		rw				r				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**PRESCALER**											
r				rw											

Field	Bits	Type	Description
PRESCALER	11:0	rw	**Prescaling of the Fractional Divider** Prescaler bit-field with values in the range of 0 to 4095, defining division ratios from 1 to 4096. Used also as a microtick generator for the input digital filter.
OVERSAMPLING	19:16	rw	**Oversampling Factor** Defines the bit length in ticks in the range of 1 to 16. The lengths of 1 to 3 are not allowed. The position of the sampling points is shown in **Figure 436**. 0_H 1 (not allowed) 1_H 2 (not allowed) 2_H 3 (not allowed) 3_H 4 … F_H 16
SAMPLEPOINT	27:24	rw	**Sample Point Position** Programmable in the range of 0 to 15 according to the **Figure 436**. For example, if three sample points at position 7, 8, 9 are required, this bit field would contain 9. In SPI mode, this bit field + 1 defines the length of the first SCLK half period in ticks. Values equal or higher then the OVERSAMPLING value are forbidden. 0_H 0 (not allowed) 1_H 1 … F_H 15

Field	Bits	Type	Description
SM	31	rw	**Sample Mode** Number of samples per bit. 0_B 1 1_B 3
0	15:12, 23:20, 30:28	r	**Reserved** Read as 0; should be written with 0.

Frame Control Register

The parameters regarding the properties of the message frame of the ASC communication are controlled by the Frame Control Register FRAMECON.
Note: The register is only writable, if CSR.CLKSEL=0.

FRAMECON
Frame Control Register **(018_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ODD	**PEN**	**CEN**	**MSB**	**0**										**MODE**	
rw	rw	rw	rw	r										rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**LEAD**			**STOP**			**IDLE**			**0**					
r	rw			rw			rw			r					

Field	Bits	Type	Description
IDLE	8:6	rw	**Duration of the IDLE delay** Defines the duration of the IDLE delay in bit times. If more characters are available in the TXFIFO, this is the pause inserted between the characters. In the SPI mode, this is the idle time between the frames. In the ASC and LIN mode, this is the pause inserted between transmission of bytes. Idle also applies to the pause between the header and the response (response space). *Note: The collision detection runs in parallel to the idle delay and in LIN master mode it may extend the time between two bytes for one bit length. This effect may occur if the round trip delay including the digital filter delay is longer than the idle delay. For LIN slave mode use IDLE=0.* 000_B 0 … 111_B 7

Field	Bits	Type	Description
STOP	11:9	rw	**Number of Stop Bits** Defines the number of stop bits in ASC and LIN mode, or the trailing delay in SPI mode. In ASC mode, standard values are 1 and 2. In LIN mode, standard value is 1. In SPI mode there is no standard value. Nevertheless, all settings are possible in all modes. 000_B 0 (not allowed in ASC and LIN modes) 001_B 1 … 111_B 7
LEAD	14:12	rw	**Duration of the Leading Delay** Defines the leading delay in bit times in SPI mode. Has no meaning in the ASC mode. In LIN mode, this is a delay inserted between the end of the break and the start of the sync character. 000_B 0 (not allowed in LIN mode and 4-Wire SPI mode) 001_B 1 … 111_B 7
MODE	17:16	rw	**Mode Selection** This bit field defines the basic operating mode of the module. In INIT mode, all outputs are at inactive level, and the module does not respond to the input signals. Changing the mode of the module must be done by switching first to INIT mode, and then to the other mode. The SCLK signal generated by the module is active only in the SPI mode. The CTS output generated by the module is active only in the ASC mode. 00_B INIT mode 01_B ASC mode 10_B SPI mode 11_B LIN mode
MSB	28	rw	**Shift Direction** Defines the shift direction of the shift register. Relevant for the SPI mode. In ASC and LIN modes, should be set to zero. Parity bit is shifted out last independently of the shift direction. 0_B LSB first 1_B MSB first
CEN	29	rw	**Collision Detection Enable** Enables the collision detection mechanism. 0_B Disabled 1_B Enabled
PEN	30	rw	**Parity Enable** Enables the parity bit attached to the data bits. Parity bit can be used for ASC and SPI protocols. The standard LIN bytes do not use this parity bit. 0_B Disabled 1_B Enabled

Field	Bits	Type	Description
ODD	31	rw	**Parity Type** Defines the type of parity bit attached to the data bits. This setting is valid for all modes of operation (ASC, LIN, SPI). 0_B Even 1_B Odd
0	5:0, 15, 27:18	r	**Reserved** Read as 0; should be written with 0.

Data Configuration Register

DATCON register defines the number of bits in the ASC (affecting the LIN protocol, if used) and SPI frames, data length or the number of data bytes in a LIN response (if any) and the checksum mode. It additionally defines the time window for the LIN response.

In case the whole LIN response does not fit in this time window, an error interrupt is generated. The measurement starts at the end of the last bit of the header, and ends at the end of the last bit of the response.

DATCON
Data Configuration Register **($01C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								RESPONSE							
r								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSM	RM	HO	0									DATLEN			
rw	rw	rw	r									rw			

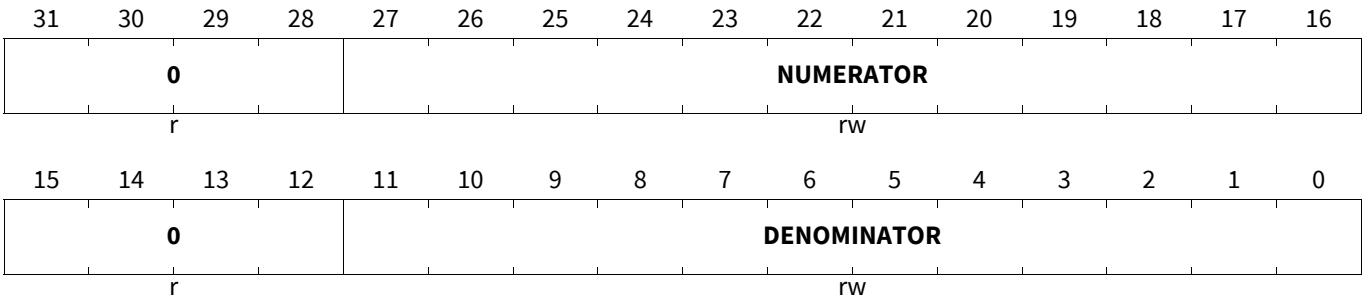
Field	Bits	Type	Description
DATLEN	3:0	rw	**Data Length** Defines the number of bits in a character. In the ASC mode, standard length is 7, 8, or 9 bits. In the SPI mode, there is no standard length. In ASC and SPI modes, any length from 2 to 16 bits is possible, although not standard for some protocols. In LIN mode, standard length is 8 bits per character. Therefore, this field defines the number of data bytes of the response. 0_H 1 … F_H 16
HO	13	rw	**Header Only** Defines if the LIN frame shall consist of a header and response or of a header only. 0_B Header and response expected 1_B Header only expected, response ignored

Field	Bits	Type	Description
RM	14	rw	**Response Mode** Defines if the RESPONSE bit field defines a LIN Response or LIN Frame timeout threshold. See **Figure 443**. 0_B Frame 1_B Response
CSM	15	rw	**Checksum Mode** Defines if the classic or the enhanced checksum will be calculated by the checksum block. 0_B Classic 1_B Enhanced
RESPONSE	23:16	rw	**Response Timeout Threshold Value** Defines the timer limit in the range of 1 to 256 bit times.
0	12:4, 31:24	r	**Reserved** Read as 0; should be written with 0.

Baud Rate Generation Register

Configures the numerator and the denominator of the fractional divider in the baud rate generation block.
Note: The register is only writable, if CSR.CLKSEL=0.

BRG
Baud Rate Generation Register **(020_H)** **Application Reset Value: $0000\ 0000_H$**

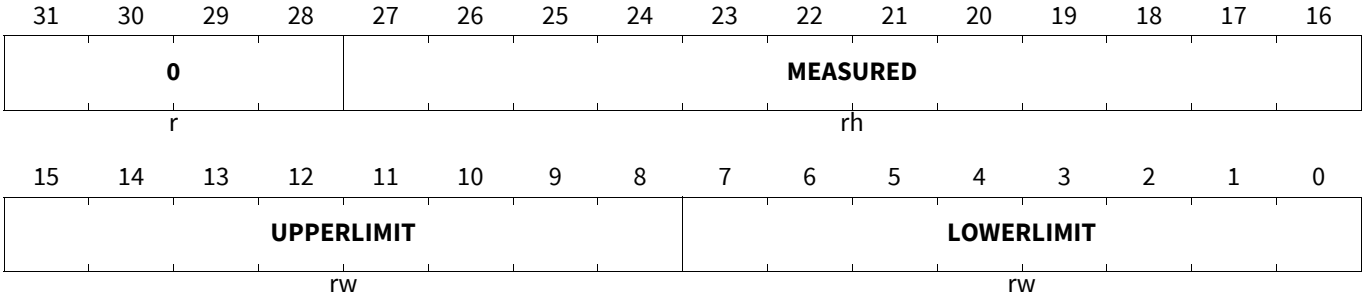


Field	Bits	Type	Description
DENOMINATOR	11:0	rw	**Denominator** Programmed by software, in a range of 0 to 4095. The setting of 0 is not allowed If the module is used as ASC, SPI, LIN master and LIN slave without autobaud detection, this value determines the baud rate. In slave mode with autobaud detection, it contains the nominal value. For the value measured by the autobaud detection hardware, see the **BRD** register.
NUMERATOR	27:16	rw	**Numerator** Defines the numerator of the fractional divider in a range of 0 to 4095. Programmed by software. The setting of 0 is not allowed.
0	15:12, 31:28	r	**Reserved** Read as 0; should be written with 0.

Baud Rate Detection Register

The BRD defines the properties specific for the automatic baud rate detection when the module operates as a LIN slave.
Note: The register is only writable, if CSR.CLKSEL=0.

BRD
Baud Rate Detection Register **(024_H)** **Application Reset Value: 0000 0000$_H$**

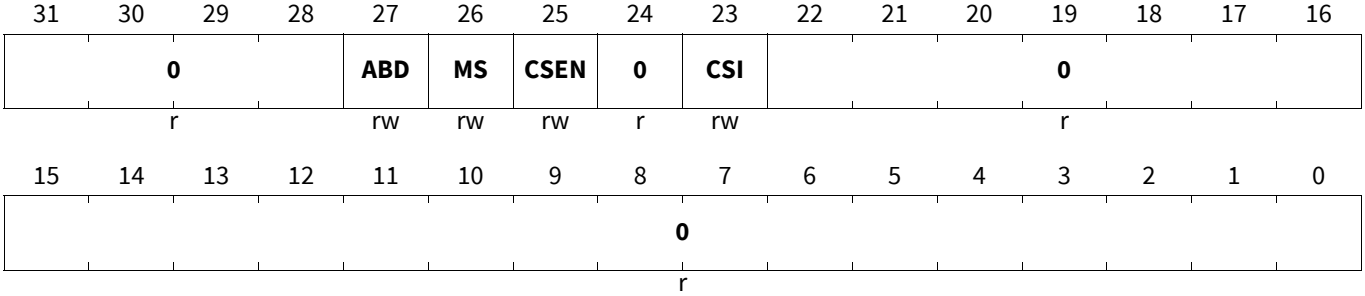


Field	Bits	Type	Description
LOWERLIMIT	7:0	rw	**Lower Limit** This field defines the 8 most significant bits of the 12 bit compare value. The lower four bits are 1000_B. See **Auto Baud Rate Detection**.
UPPERLIMIT	15:8	rw	**Upper Limit** This field defines the 8 most significant bits of the 12 bit compare value. The lower four bits are 1000_B. See **Auto Baud Rate Detection**.
MEASURED	27:16	rh	**Measured Value of 8-bits from Sync Field** This bit field contains the measured value of the duration of 8-bits form the sync field of the LIN header in microticks. It is automatically loaded in the denominator of the fractional divider, in case of LIN slave operation with autobaud detection.
0	31:28	r	**Reserved** Read as 0; should be written with 0.

LIN Control Register

LINCON contains bits that control LIN specific features of the module.
Note: The register is only writable, if CSR.CLKSEL=0.

LINCON
LIN Control Register **(028_H)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CSI	23	rw	**Checksum Injection** Defines if the received checksum byte is written into the RXFIFO or not. See **LIN Protocol Control**. 0_B Not written 1_B Written
CSEN	25	rw	**Hardware Checksum Enable** Enables the hardware checksum generation and checking. 0_B Disabled 1_B Enabled
MS	26	rw	**Master Slave Mode** Configures if the module in LIN mode operates as master or as slave. 0_B Slave 1_B Master
ABD	27	rw	**Autobaud Detection** Enables the autobaud detection feature in LIN slave mode. In all other operating modes of the module (LIN master, ASC, SPI) not effective. If the autobaud detection is disabled (the oscillator precision of the slave is sufficient), the sync field (byte field and stop bit) is checked if correct. If not correct, a framing error is triggered. 0_B Disabled 1_B Enabled
0	22:0, 24, 31:28	r	**Reserved** Read as 0; should be written with 0.

LIN Break Timer Register

This register defines

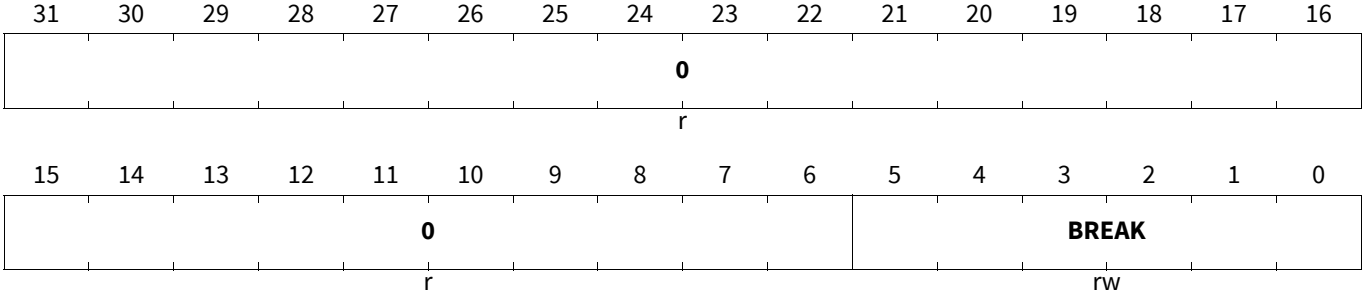
- break detection limit if the module operates as LIN slave or
- length of the generated break pulse if the module operates as LIN master.

The break timer, if enabled, monitors the bus continuously.

Note: The register is only writable, if CSR.CLKSEL=0.

LINBTIMER

LIN Break Timer Register **($02C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
BREAK	5:0	rw	**Break Pulse Generation and Detection** In LIN slave mode, this bit field defines the duration of the detection threshold for the break pulse. In LIN master mode, this bit field defines the duration of the transmitted break pulse. The time unit is bit time.
0	31:6	r	**Reserved** Read as 0; should be written with 0.

LIN Header Timer Register

LINHTIMER register defines the time windows for the header of a LIN frame.

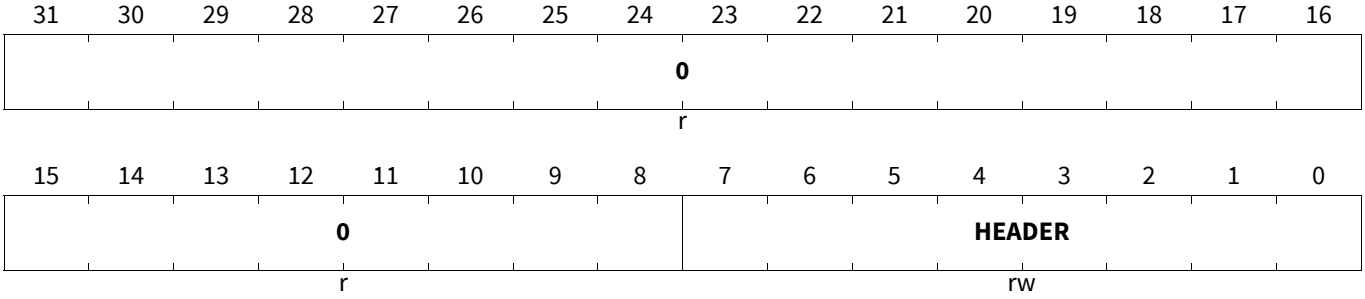
In master mode, the timer starts counting at the falling edge of the break pulse, and stops counting when the last bit of the header (including the stop bits) has been transmitted. If the predefined time in the HEADER bit field is violated, an error interrupt is generated (if enabled).

In slave mode, the timer starts counting when the break pulse has been detected, after a low time of 10 or 11 bit times, and stops counting when the last bit of the header has been received. If the predefined time in the HEADER bit field is violated, an error interrupt is generated (if enabled).

Note: The register is only writable, if CSR.CLKSEL=0.

LINHTIMER

LIN Header Timer Register **(030_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
HEADER	7:0	rw	**Header Timeout Threshold Value** Defines the timer limit in the range of 0 to 255 bit times.
0	31:8	r	**Reserved** Read as 0; should be written with 0.

Flags Register

The FLAGS register contains all the flag bits of the ASCLIN module: the LIN phase flags (header and response transmit and receive), the overflow flags of the LIN timers, and the standard ASC error flags.

A corresponding interrupt triggering can be enabled using the FLAGSENABLE register.

FLAGS

Flags Register **(034$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFL	**TFO**	**0**	**RFL**	**RFU**	**RFO**	**CE**	**LC**	**LA**	**LP**	**BD**	**RT**	**HT**	**FE**	**TC**	**PE**
rh	rh	r	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRRQ	**THRQ**	**TWRQ**	**0**						**RED**	**FED**	**0**	**RR**	**RH**	**TR**	**TH**
rh	rh	rh	r						rh	rh	r	rh	rh	rh	rh

Field	Bits	Type	Description
TH	0	rh	**Transmit Header End Flag** Signals the HEADER_TX_END event. Set by hardware, clear by software. If enabled, a transmit interrupt is triggered. 0_B No HEADER_TX_END event since the last clear by software 1_B New HEADER_TX_END event since the last clear by software
TR	1	rh	**Transmit Response End Flag** Signals that RESPONSE_TX_END event. Set by hardware, clear by software. If enabled, a transmit interrupt is triggered. 0_B No RESPONSE_TX_END event since the last clear by software 1_B New RESPONSE_TX_END event since the last clear by software
RH	2	rh	**Receive Header End Flag** Signals that HEADER_RX_END event. Set by hardware, clear by software. If enabled, a receive interrupt is triggered. 0_B No HEADER_RX_END event since the last clear by software 1_B New HEADER_RX_END event since the last clear by software
RR	3	rh	**Receive Response End Flag** Signals that RESPONSE_RX_END event. Set by hardware, clear by software. If enabled, a receive interrupt is triggered. 0_B No RESPONSE_RX_END event since the last clear by software 1_B New RX_RESPONSE_END event since the last clear by software
FED	5	rh	**Falling Edge from Level 1 to Level 0 Detected** This bit is set by hardware when a falling edge is detected on the RX line. 0_B No falling edge detected 1_B Falling edge detected
RED	6	rh	**Rising Edge from Level 0 to Level 1 Detected** This bit is set by hardware when a rising edge is detected on the RX line. 0_B No rising edge detected 1_B Rising edge detected

Field	Bits	Type	Description
TWRQ	13	rh	**Transmit Wake Request Flag** Signals that transmission of wake has been requested. No interrupt triggered. As soon as the wake pulse transmission starts, the bit is cleared by the hardware. 0_B No Transmit Wake Request pending 1_B Transmit Wake Request pending.
THRQ	14	rh	**Transmit Header Request Flag** Signals that transmission of header has been requested. No interrupt triggered. As soon as the header transmission starts, the bit is cleared by the hardware. 0_B No Transmit Header Request pending 1_B Transmit Header Request pending.
TRRQ	15	rh	**Transmit Response Request Flag** Signals that transmission of response has been requested. No interrupt triggered. As soon as the response transmission starts, the bit is cleared by the hardware. 0_B No Transmit Response Request pending 1_B Transmit Response Request pending.
PE	16	rh	**Parity Error Flag** Signals parity error. If enabled, an error interrupt is triggered. Parity error occurs if the internally calculated parity bit is not equal to the received parity bit. 0_B Last message received error free 1_B Last message received with parity error
TC	17	rh	**Transmission Completed Flag** Signals an end of an ASC or SPI frame. This bit is set after the last stop bit transmission in ASC mode, or after the trailing delay in case of SPI mode. In LIN mode, if the node is transmitting a header this flag is set after each transmission break (incl lead) field, Sync field or PID field. If the node is transmitting a response, this flag is set after each byte is transmitted. If enabled, an EX interrupt is triggered. Should be cleared by software. 0_B No end of frame event occurred 1_B End of frame event occurred
FE	18	rh	**Framing Error Flag** Signals framing error. If enabled, an error interrupt is triggered. Framing error occurs if “0” is received at a stop bit position. If autobaud detection is deactivated, then the sync field is checked for framing error. 0_B Last message received error free 1_B Last message received with error
HT	19	rh	**Header Timeout Flag** Signals violation of the header duration limit. If enabled, an error interrupt is triggered. 0_B No HEADER_OVERFLOW event since the last clear by software 1_B New HEADER_OVERFLOW event since the last clear by software

Field	Bits	Type	Description
RT	20	rh	**Response Timeout Flag** Signals violation of the response or frame duration limit as defined in **DATCON**.RM bit. If enabled, an error interrupt is triggered. 0_B No timeout event since the last clear by software 1_B New timeout event since the last clear by software
BD	21	rh	**Break Detected Flag** Signals a detection of a break pulse. If enabled, an error interrupt is triggered. Slave mode only. 0_B No Break/Wake/Stuck event since the last clear by software 1_B New Break/Wake/Stuck event since the last clear by software
LP	22	rh	**LIN Parity Error Flag** Signals parity error in the LIN identifier. If enabled, an error interrupt is triggered. Applies to LIN mode only. LIN parity error occurs if the internally calculated parity bits are not equal to the received parity bits. 0_B Last ID received error free 1_B Last ID received with parity error
LA	23	rh	**LIN Autobaud Detection Error Flag** Signals baudrate outside the range defined by **BRD**.LOWERLIMIT and **BRD**.UPPERLIMIT. 0_B No autobaud detection error 1_B Autobaud detection error
LC	24	rh	**LIN Checksum Error Flag** Signals checksum error when receiving response, if the internally calculated checksum is different than the received checksum. If enabled, an error interrupt is triggered. 0_B Last checksum error free 1_B Last checksum shows an error
CE	25	rh	**Collision Detection Error Flag** When transmitting, signals if the transmitted data differs from the received data. If enabled, an error interrupt is triggered in case of a mismatch. Collision detection is mandatory only when supporting LIN version 2.1. 0_B No mismatch 1_B Mismatch detected
RFO	26	rh	**Receive FIFO Overflow Flag** Signals an overflow error. If enabled, an error interrupt is triggered. 0_B No overflow error occurred 1_B Overflow error occurred
RFU	27	rh	**Receive FIFO Underflow Flag** Signals an underflow error. If enabled, an error interrupt is triggered. See also **RxFIFO Overview**. 0_B No underflow error occurred 1_B Underflow error occurred

Field	Bits	Type	Description
RFL	28	rh	**Receive FIFO Level Flag** This flag signals whenever a RXFIFO fill interrupt is generated based on RXFIFOCON.FM mode. 0_B No receive interrupt occurred 1_B Receive interrupt occurred
TFO	30	rh	**Transmit FIFO Overflow Flag** Signals an overflow error. If enabled, an error interrupt is triggered. 0_B No overflow error occurred 1_B Overflow error occurred
TFL	31	rh	**Transmit FIFO Level Flag** This flag signals whenever a TXFIFO refill interrupt (if enabled) is generated based on TXFIFOCON.FM mode. 0_B No transmit interrupt occurred 1_B Transmit interrupt occurred
0	4, 12:7, 29	r	**Reserved** Read as 0; should be written with 0.

Flags Set Register

The FLAGSSET register contains the write only bits used to set the corresponding bits in the FLAGS register by software. Setting a flag bit triggers an interrupt, if the corresponding interrupt enable bit is set. A read access to the register always results in 0.

FLAGSSET
Flags Set Register **(038_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFLS	**TFOS**	**0**	**RFLS**	**RFUS**	**RFOS**	**CES**	**LCS**	**LAS**	**LPS**	**BDS**	**RTS**	**HTS**	**FES**	**TCS**	**PES**
w	w	r	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRRQS	**THRQS**	**TWRQS**	**0**						**REDS**	**FEDS**	**0**	**RRS**	**RHS**	**TRS**	**THS**
w	w	w	r						w	w	r	w	w	w	w

Field	Bits	Type	Description
THS	0	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
TRS	1	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag

Field	Bits	Type	Description
RHS	2	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
RRS	3	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
FEDS	5	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
REDS	6	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
TWRQS	13	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
THRQS	14	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
TRRQS	15	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
PES	16	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
TCS	17	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag

Field	Bits	Type	Description
FES	18	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
HTS	19	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
RTS	20	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
BDS	21	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
LPS	22	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
LAS	23	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
LCS	24	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
CES	25	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
RFOS	26	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag

Field	Bits	Type	Description
RFUS	27	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
RFLS	28	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
TFOS	30	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
TFLS	31	w	**Flag Set Bit** Write of “1” in this bit sets the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Set the corresponding flag
0	4, 12:7, 29	r	**Reserved** Read as 0; should be written with 0.

Flags Clear Register

The FLAGSCLEAR register contains the write only bits used to clear the corresponding bits in the FLAGS register.

FLAGSCLEAR
Flags Clear Register **($03C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFLC	**TFOC**	**0**	**RFLC**	**RFUC**	**RFOC**	**CEC**	**LCC**	**LAC**	**LPC**	**BDC**	**RTC**	**HTC**	**FEC**	**TCC**	**PEC**
w	w	r	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRRQC	**THRQC**	**TWRQC**	**0**						**REDC**	**FEDC**	**0**	**RRC**	**RHC**	**TRC**	**THC**
w	w	w	r						w	w	r	w	w	w	w

Field	Bits	Type	Description
THC	0	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clear the corresponding flag

Field	Bits	Type	Description
TRC	1	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clear the corresponding flag
RHC	2	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clear the corresponding flag
RRC	3	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clear the corresponding flag
FEDC	5	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clear the corresponding flag
REDC	6	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clear the corresponding flag
TWRQC	13	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clear the corresponding flag
THRQC	14	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clear the corresponding flag
TRRQC	15	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clear the corresponding flag
PEC	16	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag

Field	Bits	Type	Description
TCC	17	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
FEC	18	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
HTC	19	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
RTC	20	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
BDC	21	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
LPC	22	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
LAC	23	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
LCC	24	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
CEC	25	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag

Field	Bits	Type	Description
RFOC	26	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
RFUC	27	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
RFLC	28	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
TFOC	30	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
TFLC	31	w	**Flag Clear Bit** Write of “1” in this bit clears the bit at the corresponding position in the FLAGS register. 0_B No action / This value will be always read back from the Bit. 1_B Clears the corresponding flag
0	4, 12:7, 29	r	**Reserved** Read as 0; should be written with 0.

Flags Enable Register

The FLAGSENABLE register contains the read write bits that enable the error interrupt in case the corresponding event has occurred.

FLAGSENABLE
Flags Enable Register **(040_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TFLE	**TFOE**	**0**	**RFLE**	**RFUE**	**RFOE**	**CEE**	**LCE**	**LAE**	**LPE**	**BDE**	**RTE**	**HTE**	**FEE**	**TCE**	**PEE**
rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									**REDE**	**FEDE**	**0**	**RRE**	**RHE**	**TRE**	**THE**
r									rw	rw	r	rw	rw	rw	rw

Field	Bits	Type	Description
THE	0	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
TRE	1	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
RHE	2	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
RRE	3	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
FEDE	5	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
REDE	6	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
PEE	16	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
TCE	17	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
FEE	18	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled

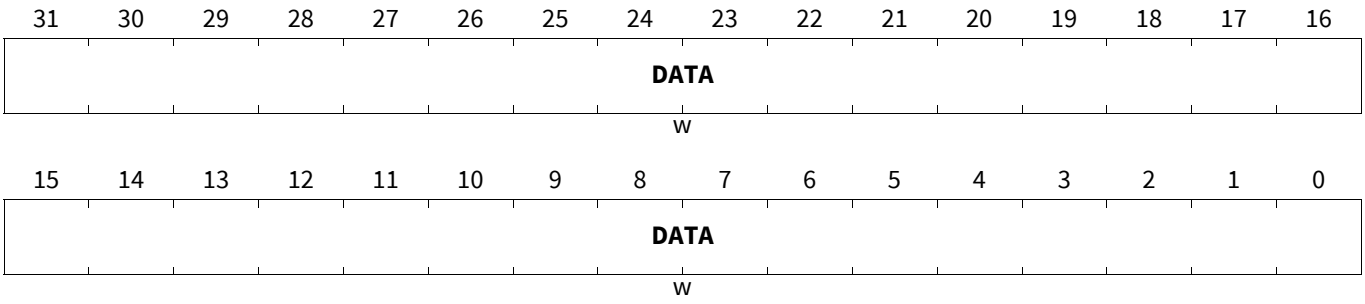
Field	Bits	Type	Description
HTE	19	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
RTE	20	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
BDE	21	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
LPE	22	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
LAE	23	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
LCE	24	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
CEE	25	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
RFOE	26	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
RFUE	27	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled

Field	Bits	Type	Description
RFLE	28	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
TFOE	30	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
TFLE	31	rw	**Flag Enable Bit** This bit enables the interrupt for the flag at the corresponding position in the FLAGS register. 0_B Disabled 1_B Enabled
0	4, 15:7, 29	r	**Reserved** Read as 0; should be written with 0.

Transmit Data Register

Writing data to this register enters the data to the TXFIFO.

TXDATA
Transmit Data Register (044_H) Application Reset Value: $0000\ 0000_H$

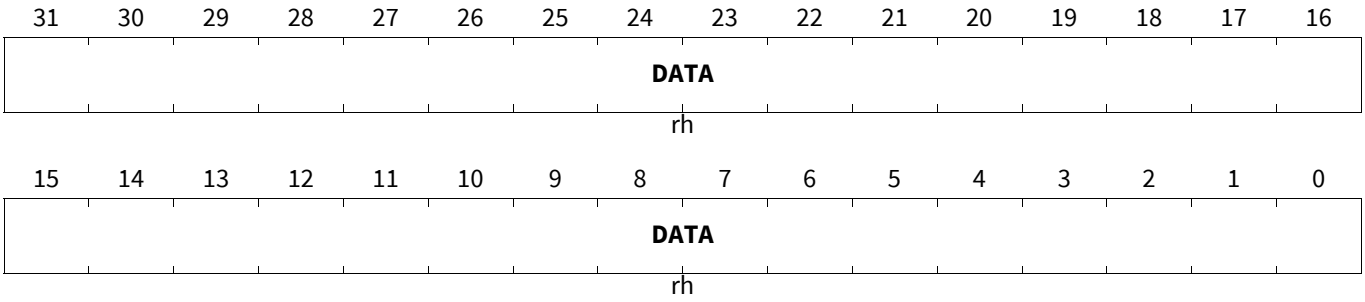


Field	Bits	Type	Description
DATA	31:0	w	**Data** Writing to this bit field writes the content to the TXFIFO, depending on the write width - 8, 16 or 32 bit (configured in TXFIFOCON.INW). A read access to this register returns 0x0.

Receive Data Register

Reading data from this register takes data from the RXFIFO.

RXDATA
Receive Data Register **(048_H)** **Application Reset Value: $0000\ 0000_H$**

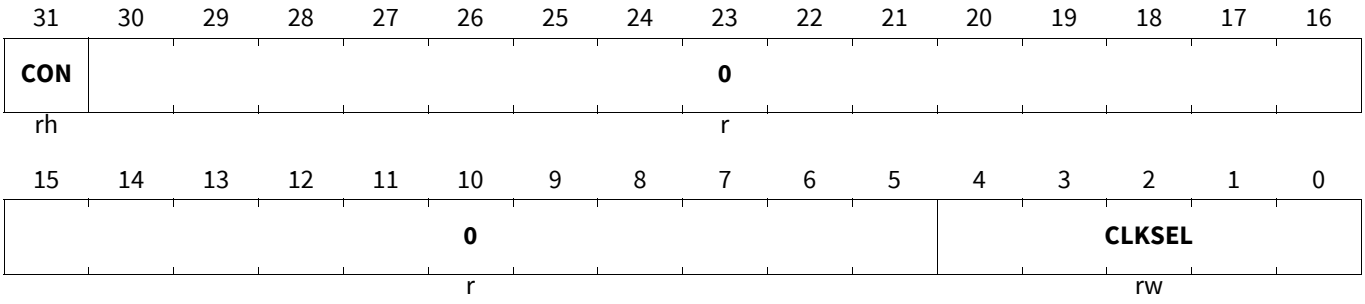


Field	Bits	Type	Description
DATA	31:0	rh	**Data** Reading from this bit field takes content from the RXFIFO, depending on the read width - 8, 16 or 32 bit (configured in RXFIFOCON.OUTW). A write access to this register has no effect.

Clock Selection Register

This register is used to select the clock source for the baud rate generation, detection, timeouts and the SPI delays.

CSR
Clock Selection Register **($04C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CLKSEL	4:0	rw	**Baud Rate Logic Clock Select** 00_H No clock supplied 02_H $f_{ASCLINF}$ 04_H $f_{ASCLINS}$ (and f_{OSC0}, see CCU) **others**, not allowed
CON	31	rh	**Clock On Flag** Shows if the clock in the bit time domain is switched on or off. Many configuration registers can be written only if this bit shows 0 (see header of registers IOCR, BITCON, FRAMECON, BRG, BRD, LINCON, LINBTIMER, LINHTIMER). 0_B Clock is off 1_B Clock is on

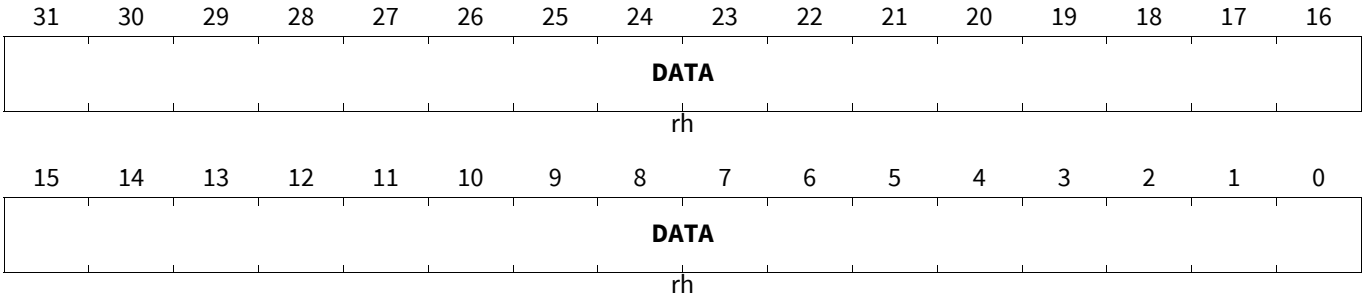
Field	Bits	Type	Description
0	30:5	r	**Reserved** Read as 0; should be written with 0.

Receive Data Debug Register

Reading data from this register takes data from the RXFIFO, but does not influence the read pointer. This virtual register provides non-destructive read to the RXFIFO.

RXDATAD

Receive Data Debug Register **(050_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DATA	31:0	rh	**Data** Reading from this bit field takes content from the RXFIFO, depending on the read width - 8, 16 or 32 bit (see RXFIFOCON.OUTW), but does not influence the read pointer of the RXFIFO. A write access to this register has no effect.

Clock Reconfiguration

The reconfiguration of the clock source has to be done by using two writes: first a write of zero to the CLKSEL bitfield, and then a second write defining the new clock source. Between the first and the second write a delay of minimum T_W ^{3} 4 * $(1/f_A)$ + 2 * $(1/f_{CLC})$ must be inserted by software, where f_A is the frequency being switched off with the first write.

Additionally, always activate the asynchronous clock for at least two f_A cycles:

- after entering the INIT state and
- after device reset

This is an example of a correct sequence:

- CSR.CLKSEL = No_Clock (equals the reset state) (**Clock Selection Register**)
- Wait T_W or poll for CSR.CON = 0 (**Clock Selection Register**)
- FRAMECON.MODE = INIT (**Frame Control Register**)
- CSR.CLKSEL = Clock_On (**Clock Selection Register**)
- Wait T_W or poll for CSR.CON = 1
- CSR.CLKSEL = No_Clock (**Clock Selection Register**)
- Wait T_W or poll for CSR.CON = 0 (**Clock Selection Register**)
- FRAMECON.MODE = ASC (or SPI or LIN) (**Frame Control Register**)
- CSR.CLKSEL = Clock_On (**Clock Selection Register**)

- Wait T_W or poll for CSR.CON = 1 (**Clock Selection Register**)

See also LIN Master Sequences

See also LIN Slave Sequences

Abort Sequence

If a header has been transmitted in ASCLIN master mode, and the corresponding response does not come, the waiting for the response can be aborted by first entering the INIT mode and then entering the LIN mode. The sequence is identical with the example sequence above.

36.5 Use Cases

This section explains the core functionality of the ASCLIN interface with a code example. The example uses the ASC feature of the module to transmit and receive a data set via loop back mode (see also **Architecture Overview**). The following settings are used:

- The module initialized in this example is module 0.
- Data length: 8 bit
- Baud rate: 9600 Bd
- One stop bit, parity bit checking, oversampling factor 16, sampling points 7,8 and 9
- Rx and Tx interrupt

Step description to initialize the ASC module with the settings from above

(Line 1) reset ENDINIT to get access to ENDINIT protected register, here ASCLIN0_CLC (see also ENDINIT protection chapter.

(Line 2) enable the control of the module in the clock control register CLC (**Clock Control Register**)

(Line 3) store CLC (**Clock Control Register**) register in a dummy variable (has to be defined before). Read back to avoid pipeline effects.

(Line 4) set ENDINIT to lock the protected register again.

(Line 5) The clk source gets set in the clock selection register CSR (**Clock Selection Register**). Before setting a source, no clk supply (CSR(4:0)=0) must be set. Final clk selection in Line 20.

(Line 6) This line sets the loop back mode in the input and output control register IOCR (**Input and Output Control Register**). (see also **Figure 416**)

(Line 7) This line clears and enables the Tx FIFO and also defines the writing size of 1 byte per clk. The Tx FIFO filling level to trigger an interrupt gets set to 0 ([11:8] =0). The Tx FIFO triggers a Tx interrupt, if the Tx FIFO filling level falls to or below that defined level. The Tx interrupt get enabled in Line 14. (see also section TxFIFO Overview and register TXFIFOCON (**TX FIFO Configuration Register**))

(Line 8) This line installs the Rx FIFO identically like the Tx FIFO from the line above. (see also register RXFIFOCON (**RX FIFO Configuration Register**) and section RxFIFO Overview)

(Line 9) The oversampling factor (here 16), the sample points (here 7,8 and 9) and the prescaler for the baudrate (here 10) gets configured in the bit configuration register BITCON (**Bit Configuration Register**). (see also section Clock System)

(Line 10) One stop bit and initialize mode as basic operation mode (necessary before switching to another mode) are getting configured in the FRAMECON (**Frame Control Register**) register. The parity bit feature with parity type even gets also enabled.

(Line 11) The data length of 8 bits gets set in the DATCON (**Data Configuration Register**) register.

(Line 12) the clk divider for the baud rate gets set to 48/3125 in the baud rate generation register BRG (**Baud Rate Generation Register**) (see also section Clock System)

*Note: Assuming f_A = 100 MHz and (defined in line 8) prescaler = 10, oversampling = 16: fshift = (fclc/prescaler * 48/3125) / oversampling = 9600 Bd*

(Line 13) Clear all interrupt flags in the FLAGSCLEAR (**Flags Clear Register**) register. Before setting the respective interrupt flags in Line 14

(Line 14) enable the Tx and Rx interrupts in the FLAGSENABLE (**Flags Enable Register**) register. The interrupts getting triggered by the FIFO's (see Line 6/7).

(Line 15) This line enables the Tx interrupt in the service request control register SRC_ASCLIN0TX and sets the interrupt priority to ASC0TX_PRIO (1…255).

(Line 16) This line enables the Rx interrupt in the service request control register SRC_ASCLIN0RX and sets the interrupt priority to ASC0RX_PRIO (1…255).

(Line 17 and 18) The interruptHandlerInstall function gets called (this function has to be written first, see interrupt handler example for more details). This function installs the interrupt service routine (ISR) entry address in the interrupt vector array with the priority ASC0TX_PRIO respectively ASC0RX_PRIO.

(Line 19) setting operating mode finally to ASC (see also Line 9).

Note: your Tx ISR function prototype would be: void ASC0_TX_ irq (void); here could be your ISR code;

Initialization of the ASC interface

```
(1)    SCU_vResetENDINIT (0);              // enable ENDINIT register(2)ASCLIN0_CLC =
0x000;                                  // disable module control (3)dummy = ASCLIN0_CLC;
//read back(4)SCU_vSetENDINIT (0);        // lock ENDINIT register(5)ASCLIN0_CSR = 0;
// clk source, no clk(6)ASCLIN0_IOCR = 0x10000001;       // Loopback(7)
ASCLIN0_TXFIFOCON = 0x00000043;                  // init TXFIFO(8)ASCLIN0_RXFIFOCON
= 0x00000043;                                    // init RXFIFO(9)ASCLIN0_BITCON =
0x830F0009;                                 // OS 16, SP 7,8,9, PS 10(10)
ASCLIN0_FRAMECON = 0x40000200;                   // 1 Stop, Init Mode, P(11)
ASCLIN0_DATCON = 0x7;               //8 Data Bits(12)ASCLIN0_BRG = 0x00300C35; //
divider for Baudrate (13)ASCLIN0_FLAGSCLEAR = 0xFFFFFFFFF;// Clear all Flags(14)
ASCLIN0_FLAGSENABLE = 0xF0000000;                // Enable TX/RX Int.
(15)   SRC_ASCLIN0TX = (1 << 10) | ASC0TX_PRIO;  (16)SRC_ASCLIN0RX = (1 << 10) |
ASC0RX_PRIO;(17)interruptHandlerInstall (ASC0TX_PRIO, & ASC0_TX_irq);
(18)   interruptHandlerInstall (ASC0RX_PRIO, & ASC0_RX_irq);
(19)   ASCLIN0_FRAMECON |= 0x00010000;// Mode ASC
```

Note: Before an interrupt is able to occur, the interrupt system has to be globally enabled. The Interrupt Control Register (ICR) holds the global interrupt enable bit (ICR.IE) which enables the CPU service request system. Most compiler support the attribute (or similar):
__enable();
to set this bit. (See also Architecture Manual for more details)

36.6 IO Interfaces

Table 327 List of ASCLIN Interface Signals

Interface Signals	I/O	Description
TX_INT	out	**ASCLIN Transmit Service Request**
RX_INT	out	**ASCLIN Receive Service Request**
ERR_INT	out	**ASCLIN Error Service Request**

Table 327 List of ASCLIN Interface Signals (cont'd)

Interface Signals	I/O	Description
sleep_n	in	**Negative turn-off request**
ACTSA	in	**Clear to send input**
ACTSB		
ACTSC		
ACTSD		
ATX	out	**Transmit output**
ATXN	out	**Differential Transmit output (low active)**
ARTS	out	**Ready to send output**
ASCLK	out	**Shift clock output**
ASLSO	out	**Slave select signal output**
ATXP	out	**Differential Transmit output (high active)**
ARXA	in	**Receive input**
ARXB		
ARXC		
ARXD		
ARXE		
ARXF		
ARXG		
ARXH		
ARXAP	in	**Differential Receive input (high active)**
ARXBP		
ARXCP		
ARXDP		
ARXEP		
ARXFP		
ARXGP		
ARXHP		
ARXAN	in	**Differential Receive input (low active)**
ARXBN		
ARXCN		
ARXDN		
ARXEN		
ARXFN		
ARXGN		
ARXHN		

36.7 Revision History

Table 328 Revision History

Reference	Change to Previous Version	Comment
V3.2.6		
Page 52	Description added that "ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices."	
V3.2.7		
Page 40	Typo corrected: FLAGSENABLE.CEN (Flags Enable Register) replaced by FRAMECON.CEN (Frame Control Register).	
Page 29	Description of Checksum Detection moved from Reception of Header (LIN Slave) to Reception of Response (LIN Slave)	
V3.2.8		
Page 15	Explanation of OUTW corrected. Affected are two sentences after **Figure 426** and values in **Table 325**.	

The revision history shows the latest changes since AURIX devices.

37 Queued Synchronous Peripheral Interface (QSPI)

The main purpose of the QSPI module is to provide synchronous serial communication with external devices using clock, data-in, data-out and slave select signals. The focus of the module is set to fast and flexible communication: either point-to-point or master-to-many slaves communication.

Parallel requests from on chip bus masters to a module will be executed sequentially via the on chip bus system. Read-modify-write feature provides an atomic read/write sequence where no other master can access the module in between. Module hardware semaphores are not supported.

>> **BPI_FPI Module Registers**

37.1 Feature List

This section describes the features of the QSPI module.

- Master and Slave Mode operation
  - Full-duplex operation
  - Half-duplex operation
  - Automatic slave select control
  - Four-wire and three-wire type of connection
- Flexible data format
  - Programmable number of data bits: 2 to 32 data bits (plus parity: 3 to 33 bits)
  - Programmable shift direction: LSB or MSB shift first
  - Programmable clock polarity: Idle low or idle high state for the shift clock
  - Programmable clock phase: data shift with leading or trailing edge of the shift clock
- Baud rate generation
  - Baud rates generated from high precision PLL clock, asynchronous to the system PLL
- Interrupt generation
  - On a transmitter FIFO event
  - On a receiver FIFO event
  - On an error condition (receive, baud rate, transmit error, parity error)
  - On a phase transition (start of frame, end of frame ...)
- QSPI supports control and data handling by the DMA controller
- Flexible QSPI pin configuration
- Hardware supported parity mode
  - Odd / even / no parity
- Seven slave select inputs $\overline{\text{SLSIB...H}}$ in Slave Mode
- Sixteen programmable slave select outputs SLSO[15:0] in Master Mode
  - Automatic SLSO generation with programmable timing
  - Programmable active level and enable control
  - External demultiplexing of the slave select outputs support
- Several module reset options
  - State machine reset per software (only the state machine)
  - Module reset per software (both FIFOs, all registers and the state machine)
  - Automatic stop option of the state machine in slave mode after baud rate error
- Loop-Back mode
- Interoperability with SSC and USIC modules of Infineon microcontroller families, and with popular (Q)SPI interfaces of multiple suppliers
- Communication stop on RxFIFO full
  - Shift register full and RxFIFO full can pause the communication
  - Interrupt generation
- High Speed Input Capture (HSIC)
  - Provides input capture functionality for ADAS applications
  - 15-bit counter with resolution of f_{PER}

37.2 Overview

This section gives a top-level overview of the QSPI.

37.2.1 Abstract Overview

An abstract overview of the QSPI module is shown in **Figure 455**. This document describes the QSPI module according to this view: the State Machine with its configuration and status capabilities, the User Interface, and the Interrupts (in this order).

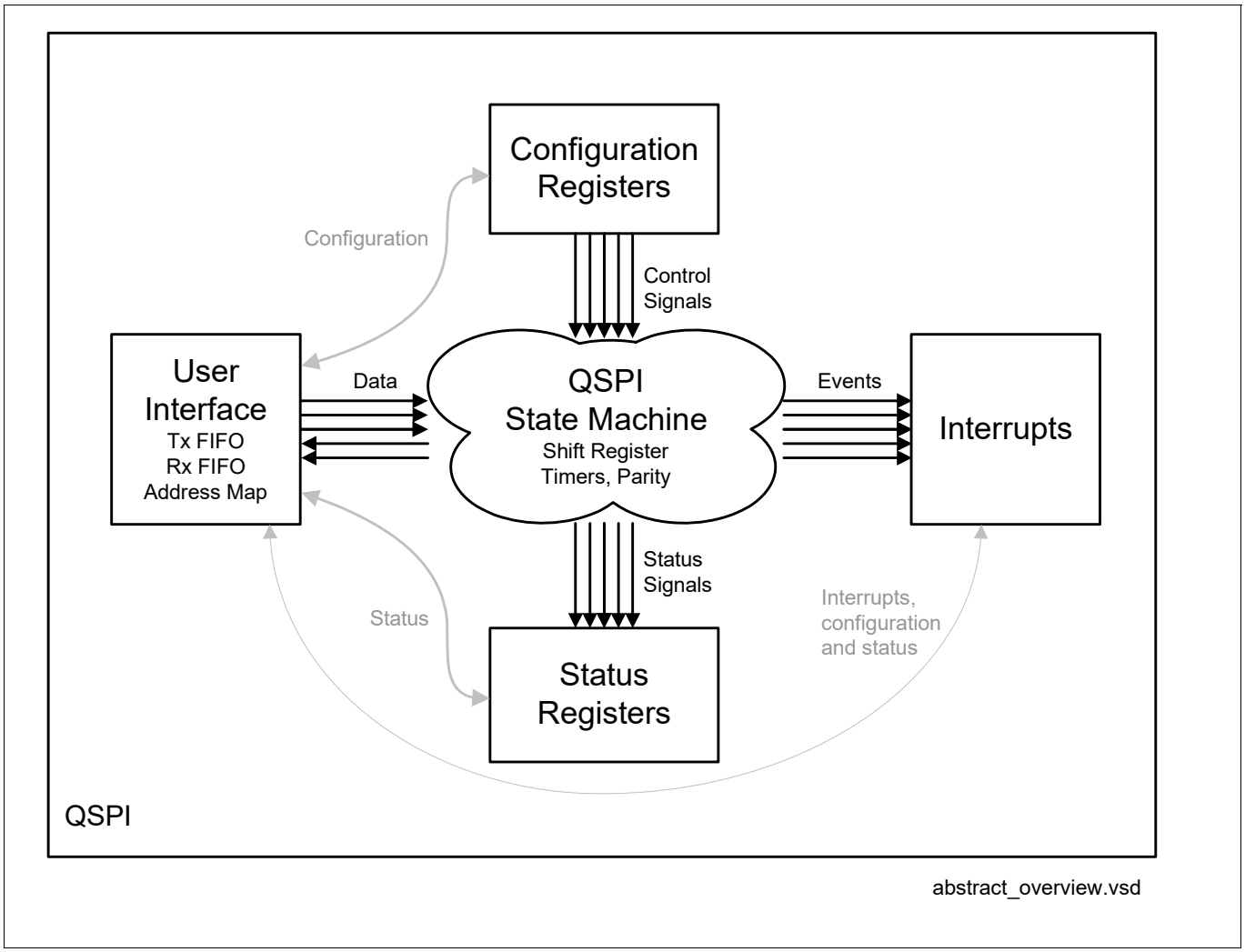


Figure 455 QSPI - Abstract Overview

37.2.2 External Signals

The communication between two devices using QSPI generally uses four signals:

- Serial clock SCLK
- Data in master to slave direction MTSR (Master Transmit Slave Receive)
- Data in slave to master direction MRST (Master Receive Slave Transmit)
- Slave select signal SLS

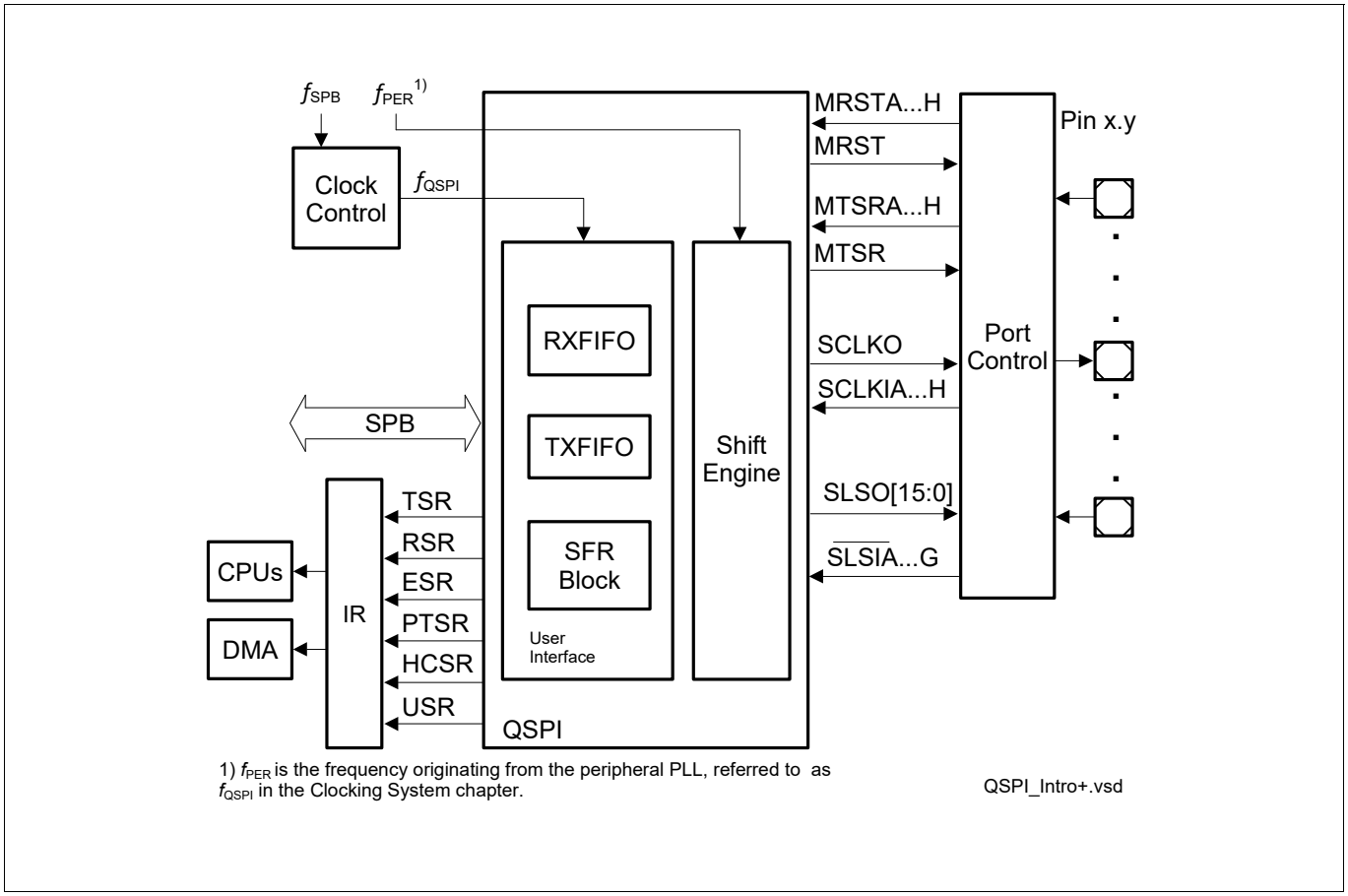


Figure 456 External Signals of the QSPI Module

37.2.3 Operating Modes

The QSPI operates in one of two modes regarding the generation of the serial clock and slave select signals: master or slave mode. The master generates and drives the clock and select signals, the slave receives these signals.

The QSPI operates in one of three modes regarding the direction of the communication and depending whether the transmission and the reception appear simultaneously or not: duplex, half-duplex and simplex mode.

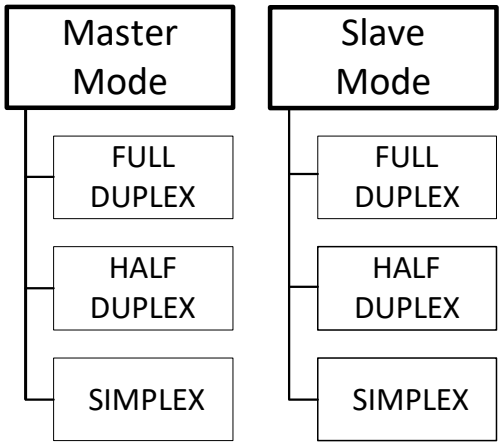


Figure 457 Operating Modes Overview

This gives six possible combinations for the operating mode of the QSPI: master duplex, half-duplex, and simplex; slave duplex, half-duplex, and simplex.

Note: The half-duplex mode can be implemented either by short-cut connection between two different pins on the pcb, one data output and one data input pin, or by using single pins mapped to both data input and data output signals. See the pinning and port chapters for the pinning definition of a particular product. The module itself does not differentiate between the full-duplex, half-duplex or simplex connection.

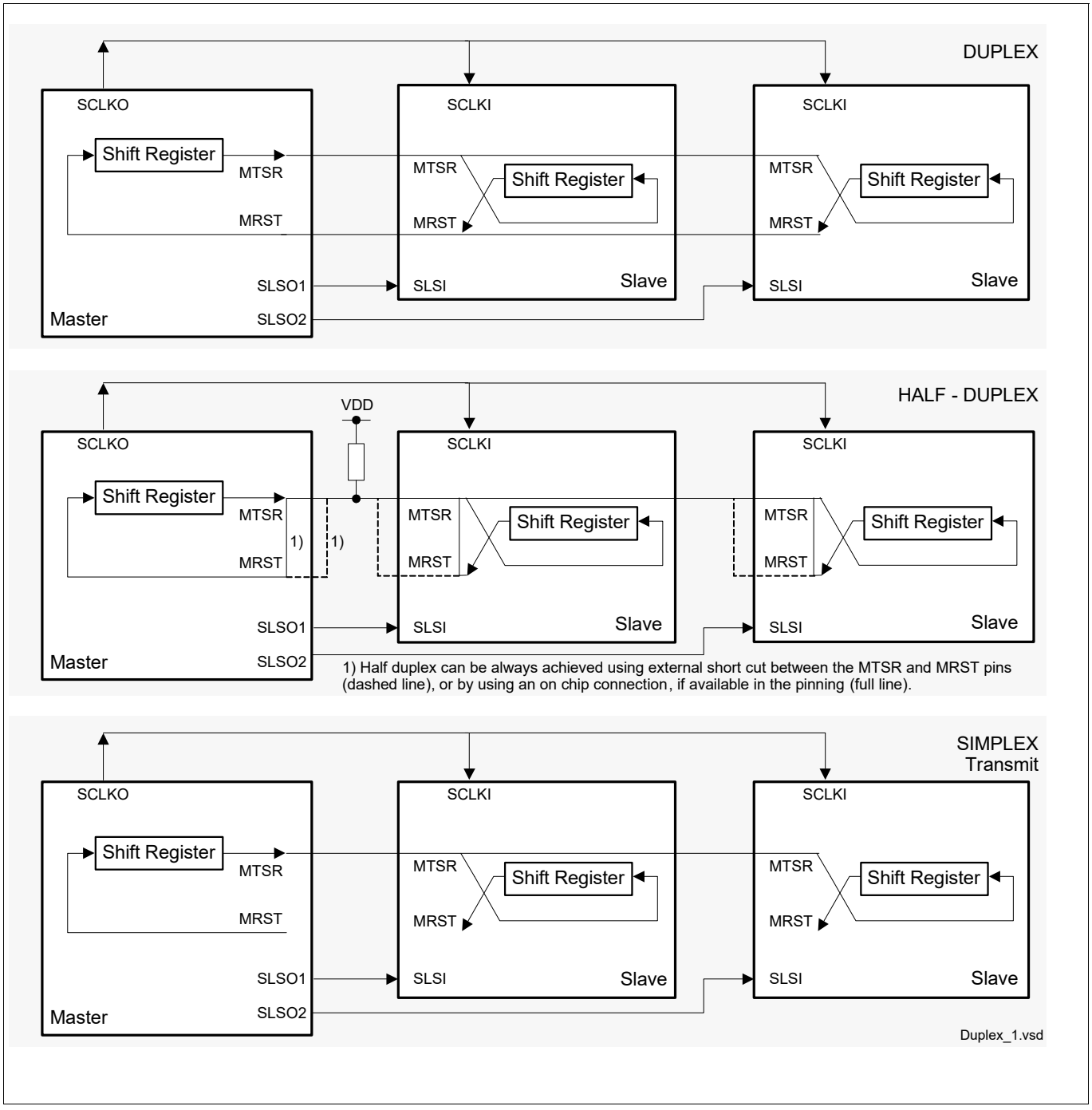


Figure 458 Operating Modes and Types of Connections

37.2.4 Queue Support Overview

The term “Queue Support” is used in this context to describe the functionality implemented for comfortable switching of the timing configuration of the QSPI frames, depending on the slave select signal which is to be activated. The main feature of the module is the possibility to take both the configuration and data to the TXFIFO, and to track down which TXFIFO entry is configuration, and which data. The QSPI module expects 32 basic configuration bits to be moved with one move (for example DMA move) from some on-chip general purpose RAM to the TXFIFO. These 32 configuration bits from the TXFIFO and the configuration bits from the 8 configuration extension registers **ECONz (z=0-7)**z contained in the QSPI module, define together the full configuration of the module (see **Figure 459**). One extension register is used for two slave selects: ECON0 for SLSO0 and SLSO8, ECON1 for SLSO1 and SLSO9...

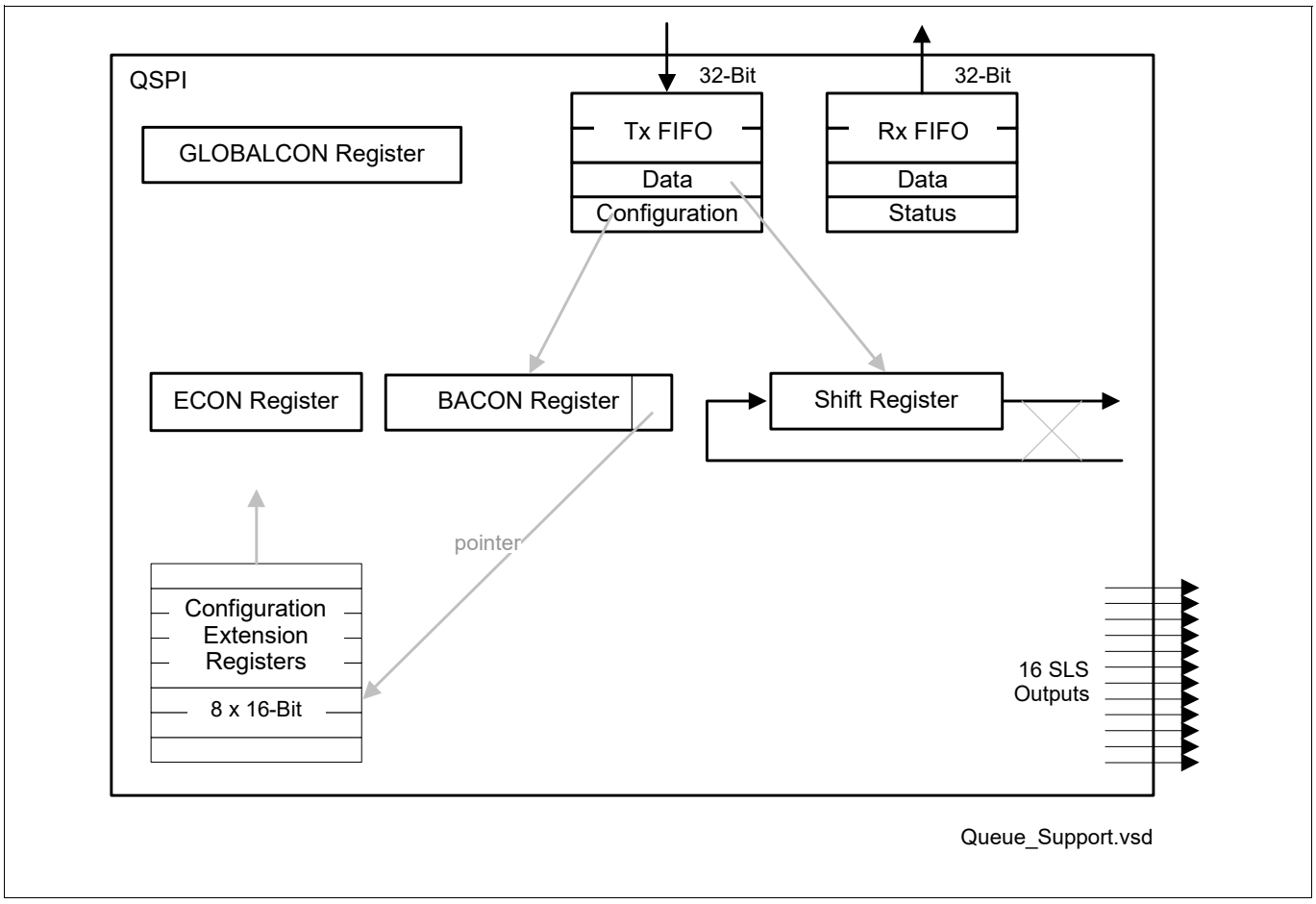


Figure 459 Queue Support Overview

37.2.5 Architecture Overview

The **Figure 460** shows the main blocks of the QSPI module. The TxFIFO and RxFIFO provide the user interface. The Shift Register and the “Miscellaneous Logic” block build the state machine of the module. The “Configuration Extensions” block provides comprehensive capabilities for configuring the QSPI frames.

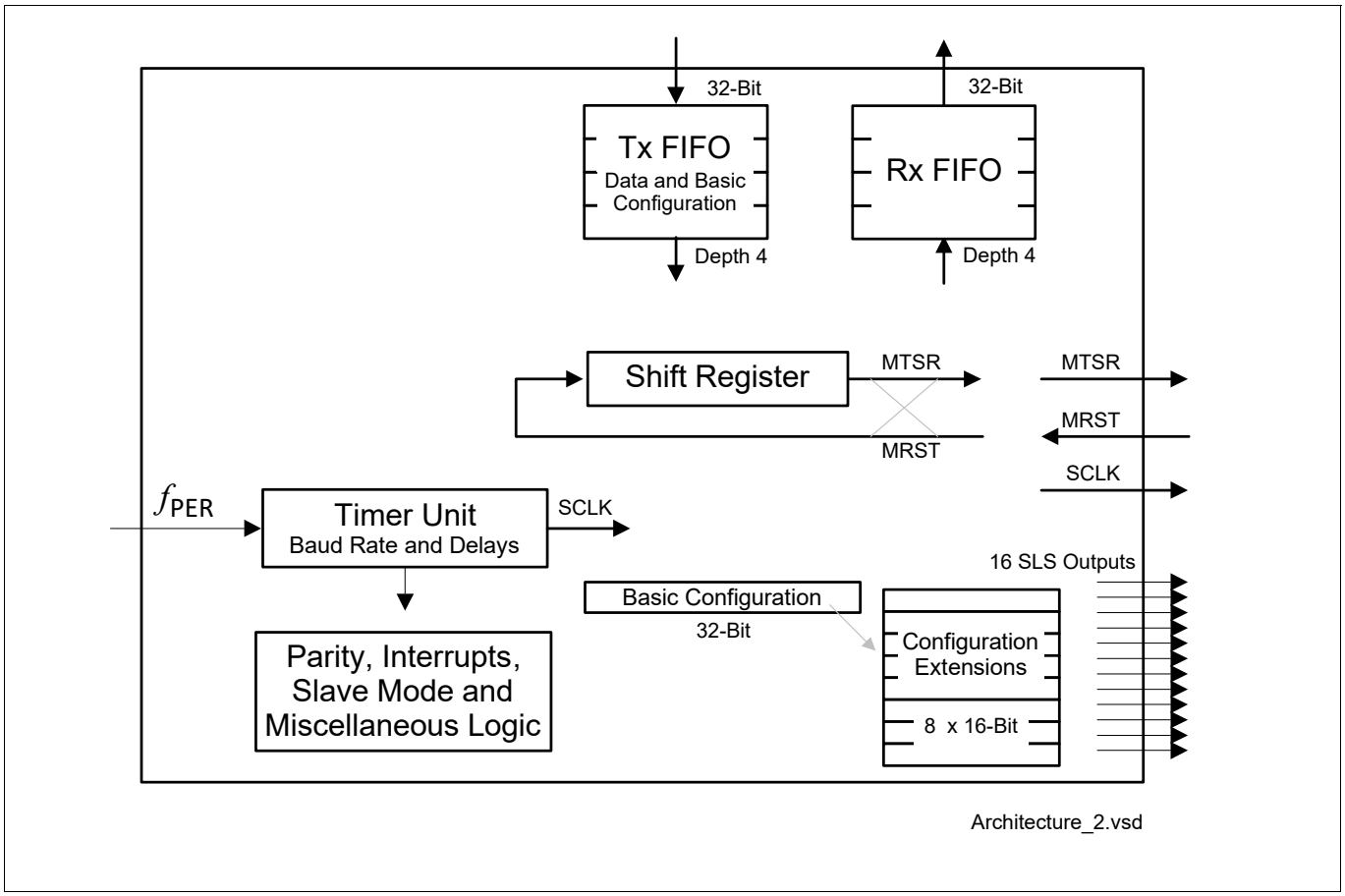


Figure 460 QSPI - Architecture Overview

37.2.6 Three Wire Connection

The term “Three Wire Connection” is used in this context to describe a connection without slave select signal. This connection relies solely on counting bits for determining the end of the current frame and resetting the shift register state machine, instead of using the slave select signal for this purpose. This way of communication is less robust than communication with slave select, but it saves a pin on both master and slave device.

The name “three wire connection” is true only in case of full-duplex connection, where exactly three signals (clock, data-in, data-out) are needed. In case of half-duplex and simplex connections only two signals (clock, data-in/data-out common data line) are necessary.

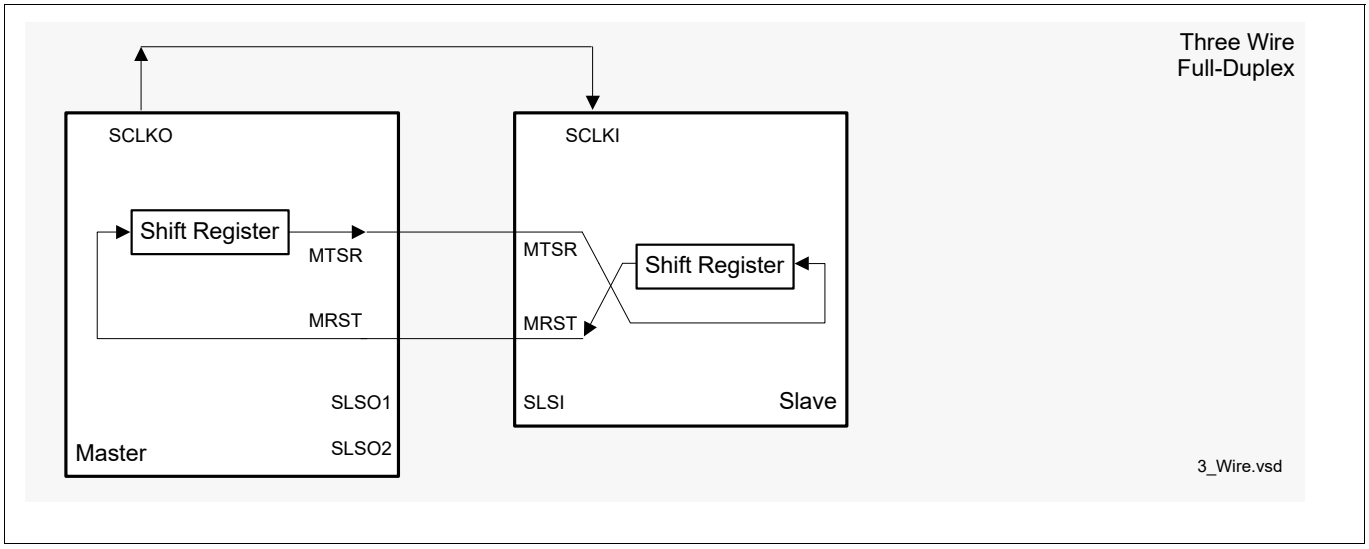


Figure 461 Three Wire Connection

37.3 Functional Description QSPI

37.3.1 Frequency Domains

The state machine of the QSPI is placed in a separate frequency domain operating with a frequency f_{PER} which is generated by the peripheral PLL. This clock is completely asynchronous to f_{SPB}.

In the Chapter Clock System, in the CCU (Clock Control Unit) section, this f_{PER} is referenced to as f_{QSPI} for overview purposes.

The baud rate is determined by the f_{PER} frequency divided by an integer equal or greater than four.

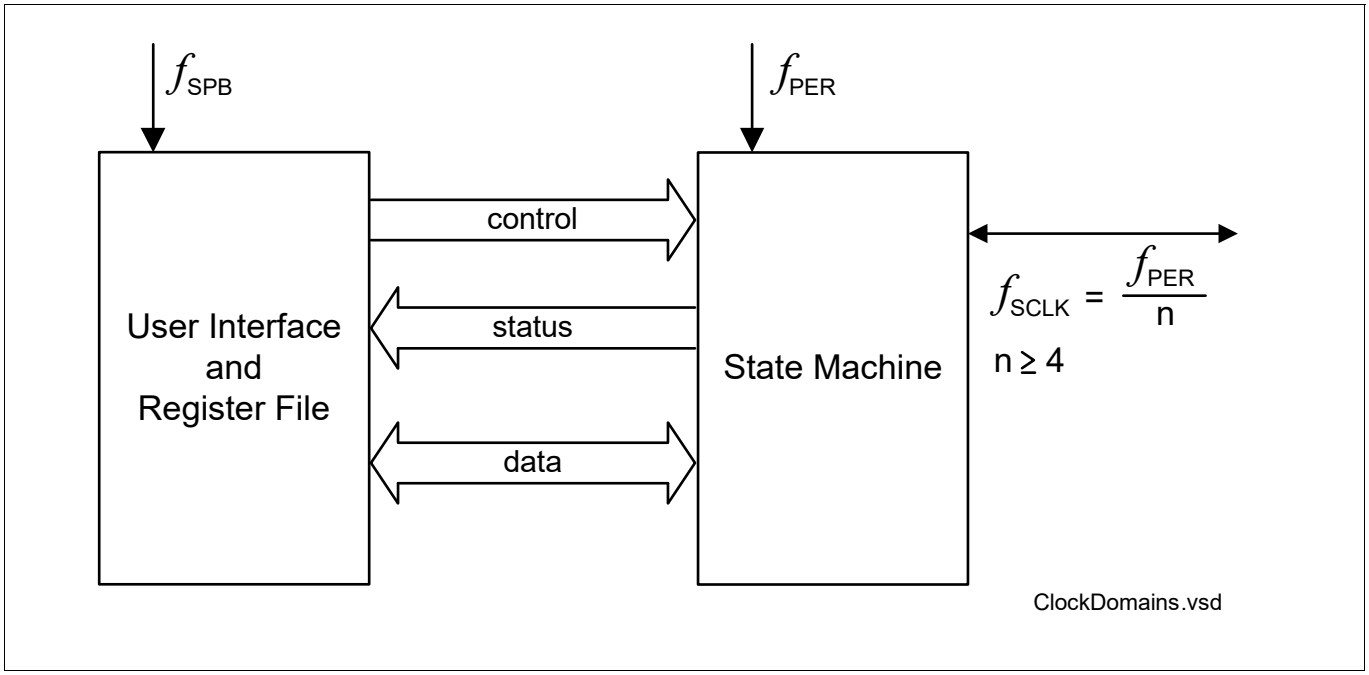


Figure 462 QSPI - Frequency Domains

37.3.2 Master Mode State Machine

In master mode, the QSPI module generates the timings, the serial clock, and the slave select signals.

37.3.2.1 Phases of one Communication Cycle

This section describes the possibilities available for configuring the length of the phases of the QSPI communication: timing delays, data length, duty cycle and data sampling.

A QSPI frame starts with activating a slave select signal SLSO (at transition from idle to leading delay phase), and ends with its deactivation (at transition from trailing delay to wait or idle phase). It is a sequence of five phases: idle delay, leading delay, data phase, trailing delay and an optional wait phase. The idle phase is subdivided in two phases of equal length: idle A and idle B.

Figure 463 shows a full and a compressed view of a QSPI frame and its phases. The full view shows all four signals needed for a QSPI connection. The compressed view represents the phases in a single row, in a way suitable to discuss their properties.

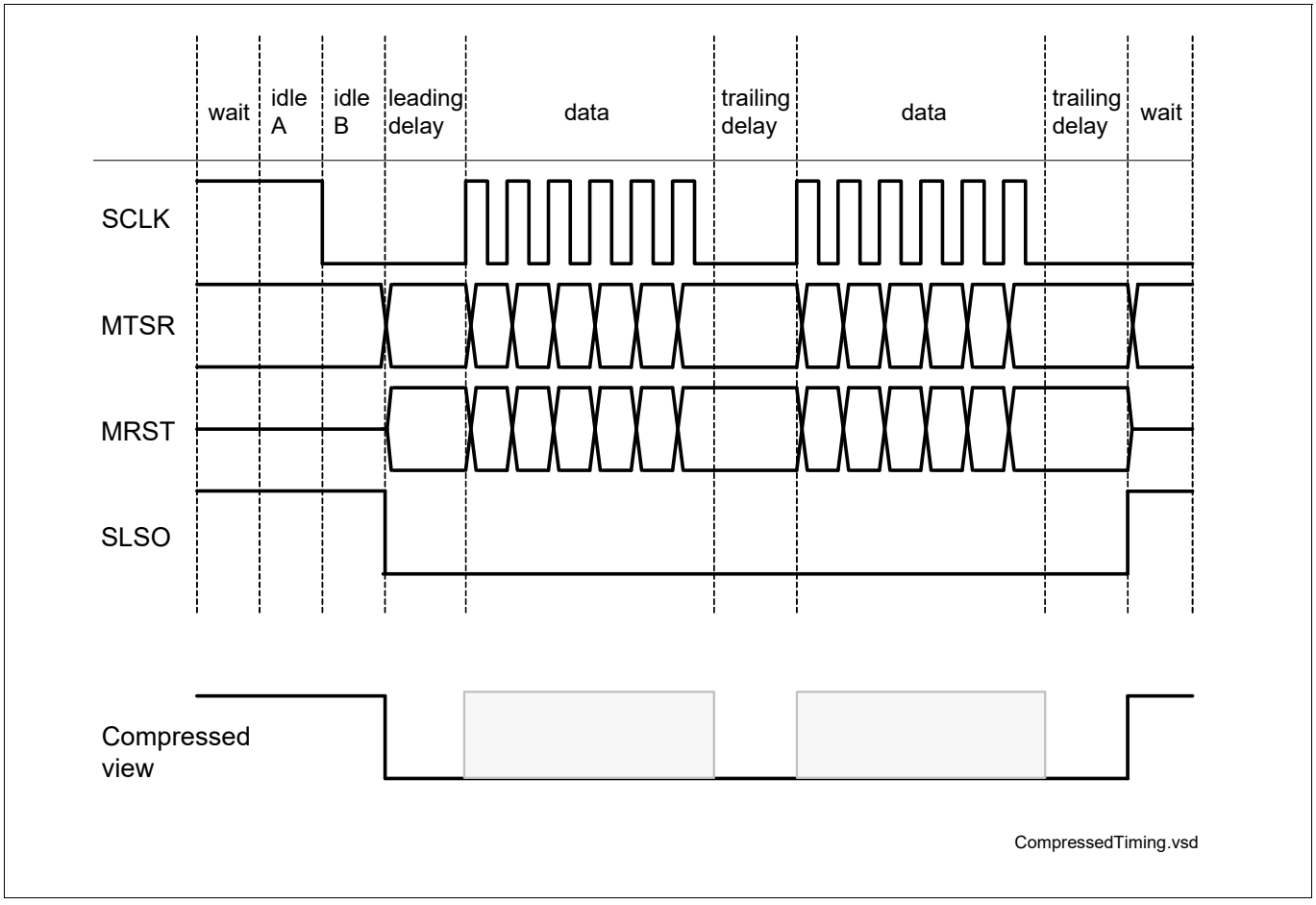


Figure 463 Phases of a QSPI Frame (Example for Data Length of 5 Bits)

The flexible timing control of the QSPI allows to program each phase in a very wide time range, with sufficient precision. Some examples of QSPI frames, in compressed view, with different lengths of the phases, are shown in **Figure 464**.

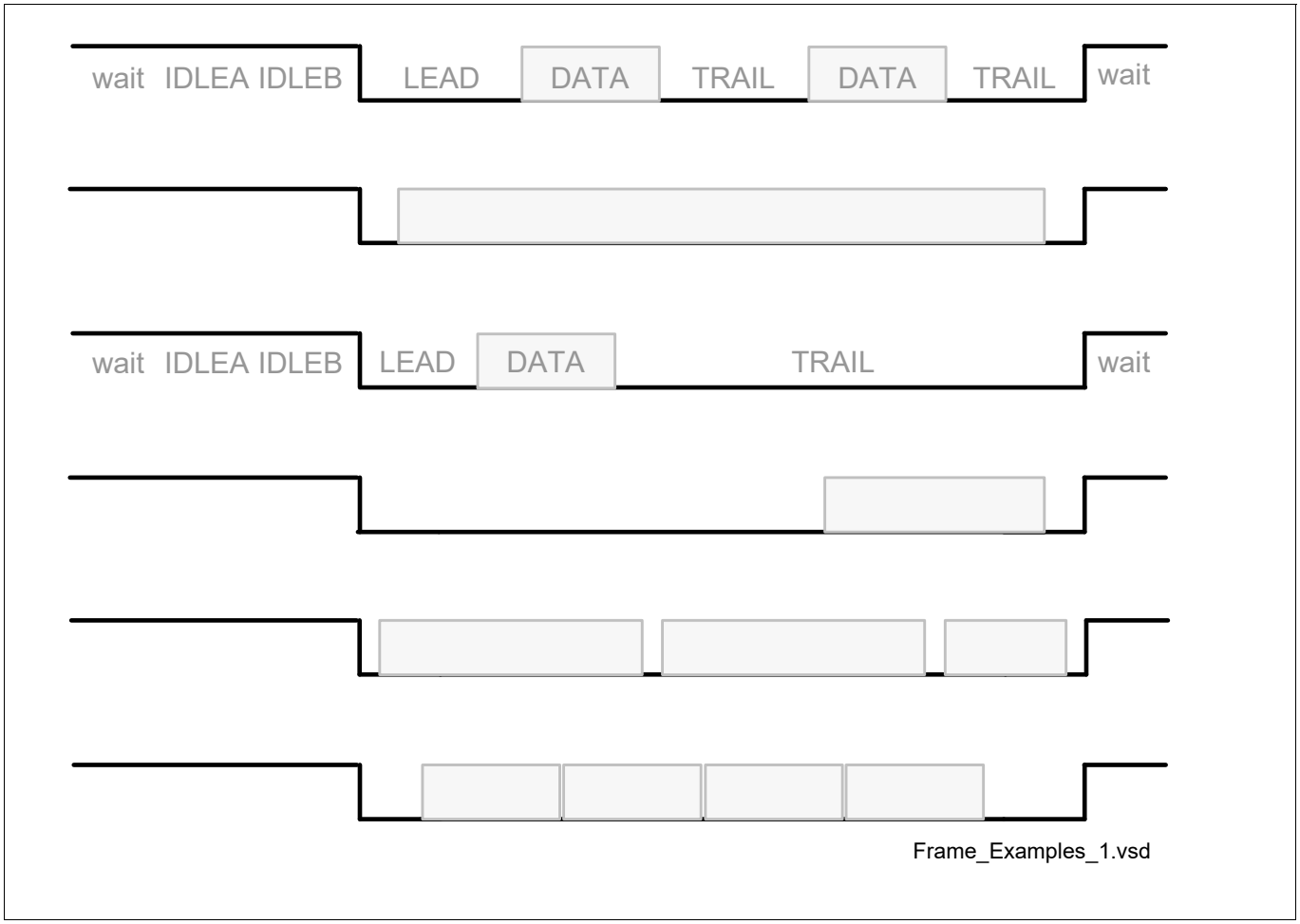


Figure 464 Examples of QSPI Frames in Compressed View

The length of the phases is programmed using the corresponding bit fields in the register **BACON**. There are four main bit fields defining the length of the phases in t_Q units and several modifier bit fields defining their range and granularity:

- IDLE
- LEAD
- DATA
- TRAIL

The WAIT phase is simply a loop waiting for a write to the Tx FIFO, without predefined duration, and consequently without a defining bit field.

The IDLE bit field defines two sub-phases with the same length IDLEA and IDLEB. At the transition between them the polarity of the serial clock signal for the next slave is set / changed.

In addition, the flexible timing control allows to program the duty cycle and the sampling point properties of the serial clock. This allows to improve to a certain extent the achievable baud rate taking into account the clock asymmetries and the loop-delay.

The **ECONz (z=0-7)** bit fields A, B and C define the length of the corresponding bit segments. The sampling point is always at the transition between phases B and C, and the clock duty cycle is defined by the phases A and B+C.

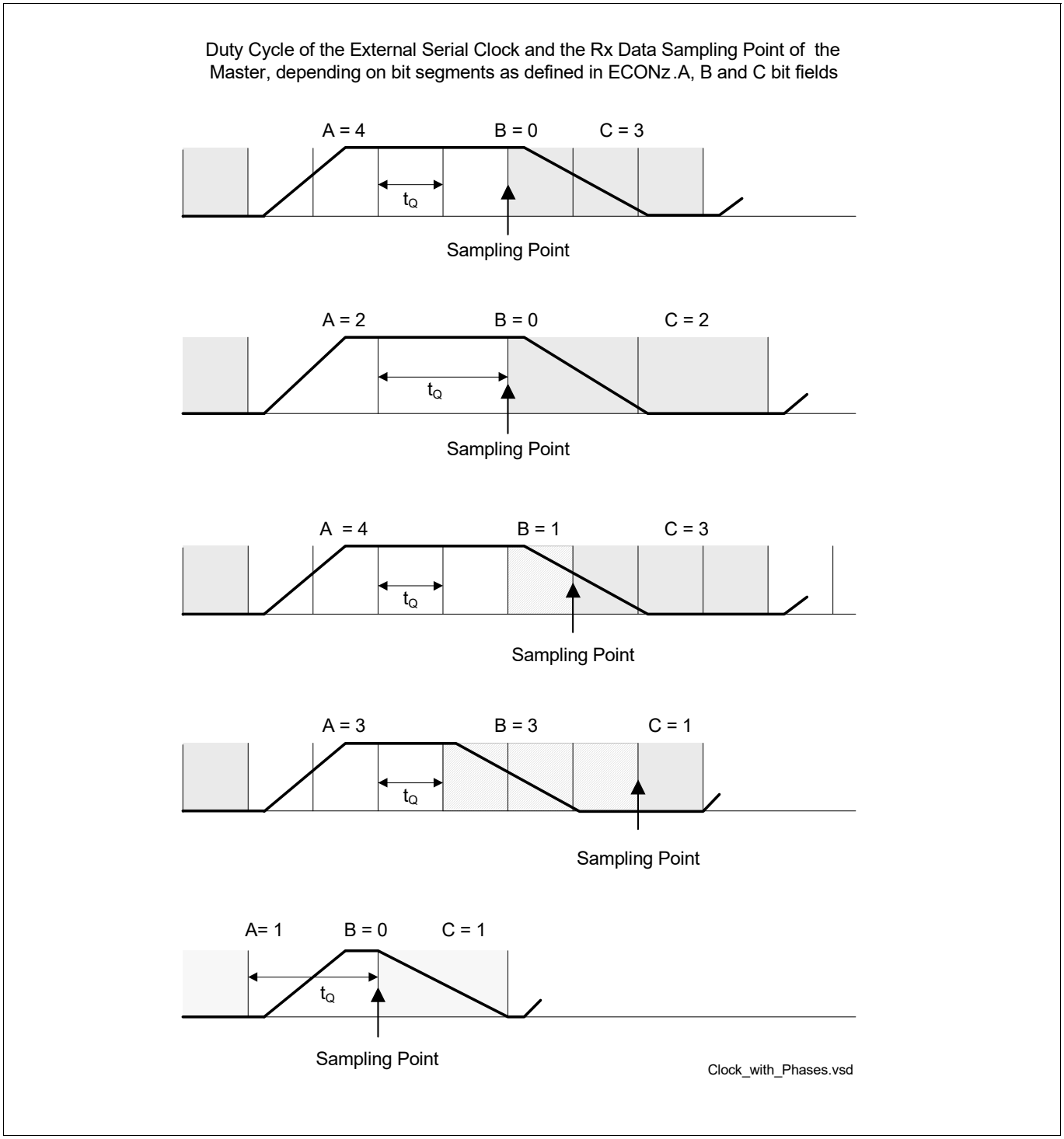


Figure 465 SCLKO Duty Cycle and the Rx Data Sampling Point Relative to SCLKO, in Master Mode

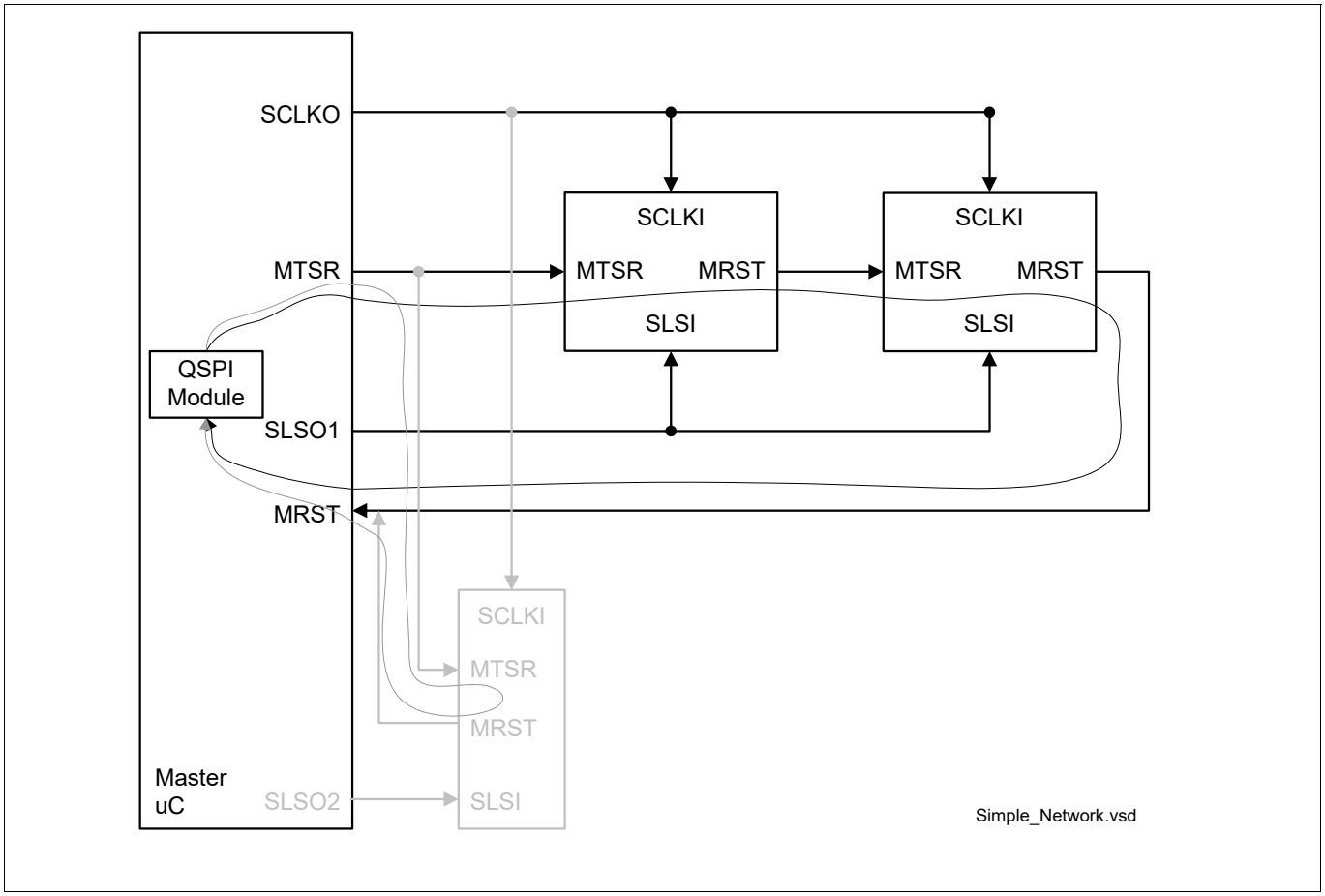


Figure 466 Closed-loop delays in a simple network consisting of one daisy-chain and one point to point connection in parallel

The purpose of the duty cycle control is:

- To compensate the influence of the path and pad asymmetry on the duty cycle
- To compensate the considerable asymmetry of the edges in case of half-duplex connection implemented as open-drain driver with pull-up resistor
- To adjust the duty cycle optimally to the timing properties of the master and slave, and to the properties of the connection (distance and capacitive load)
- To help achieve optimal baud rate, especially at higher baud rates which need an odd division factor (for example 5 = 3 high time + 2 low time or 5 = 2 high time + 3 low time)

The purpose of the sampling point control is to allow reliable reception of the data transmitted by the slave, at as high as possible baud rates.

In master receive direction the slave transmitted data reaches the master considerably late relative to the shift edge of the serial clock. This is caused by the loop delay consisting of:

- The propagation delay of the shift clock edge from the master to the slave
- The response delay of the slave
- The propagation delay of the data from the slave to the master

Last but not least, the flexible timing control allows to configure the phase and the polarity of the shift clock.

The clock polarity is configured by bit field **ECONz (z=0-7)**.CPOL. This bit field sets the clock idle polarity to low or high.

The clock phase is configured by bit field **ECONz (z=0-7)**.CPH. This bit field sets the initial clock delay to 0 or Bit Segment A, as defined by the bit field **ECONz (z=0-7)**.A.

The CPOL and CPH settings control the master mode. The slave mode timing behavior is fixed to CPOL=0 and CPH=1, and these values must be programmed in the **ECONz (z=0-7)** register pointed by **BACON**.CS. The **ECONz (z=0-7)**.A, B and C do not influence the sampling point in the slave mode, the sampling is at the falling edge of the SCLKI.

The **Figure 467** shows the QSPI timings for clock phase CPH = 0.

ECONz (z=0-7).CPH = 0 is used when communicating with a slave that delivers a valid value of the first bit immediately after being selected with an SLS signal. In such a case the first clock edge in a frame is used to latch the first bit. The second edge delivers the second bit value, and so on. The last edge in a frame delivers a bit with don't care value.

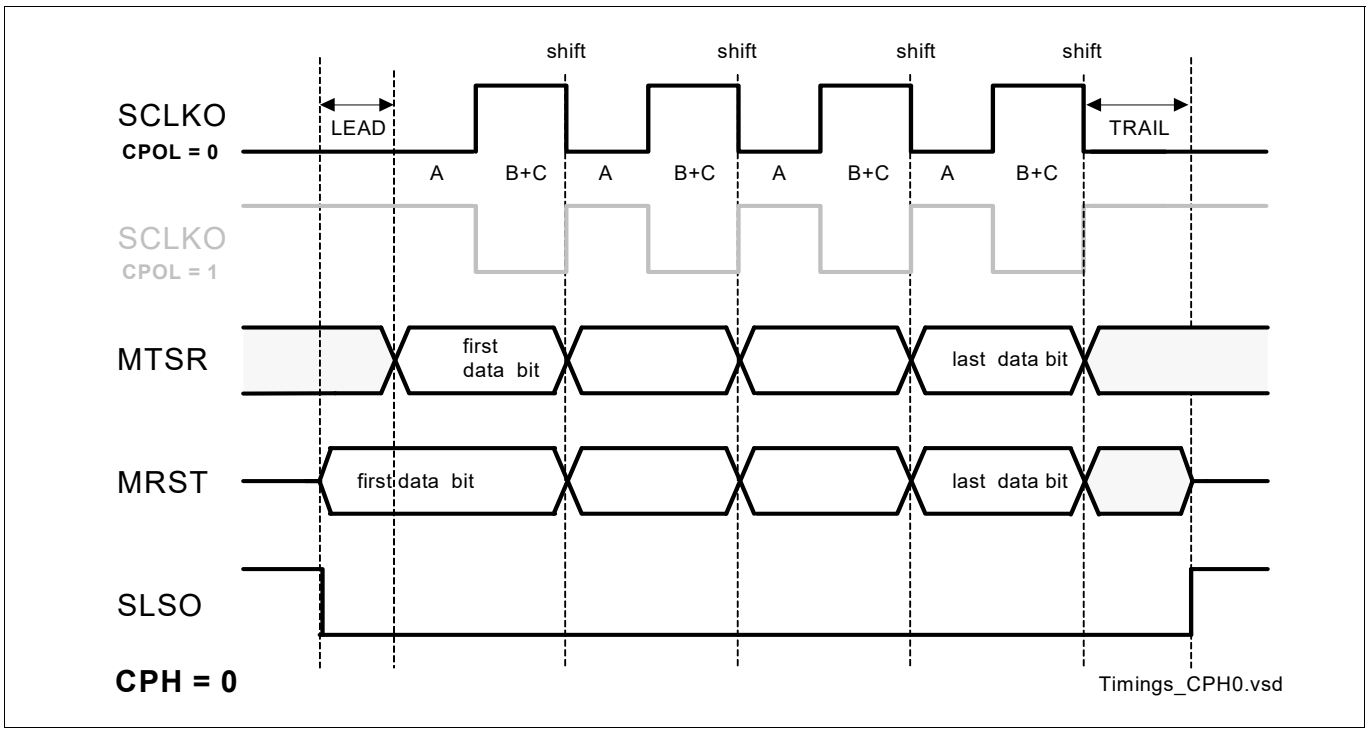


Figure 467 Timing of a transfer with CPH = 0 (Example for Data Length of 4 Bits)

The trailing delay starts after the last shift clock period of a data block and is followed either by the deactivating edge of SLSO, or a new data block in continuous mode.

Attention: ***If CPH = 0, then the total delay between the SLSO activating edge and the first edge of the serial clock SCLKO is LEAD + ECONz (z=0-7).A***

Figure 468 shows the QSPI timings for clock phase CPH = 1.

ECONz (z=0-7).CPH = 1 is used when communicating with a slave that delivers a bit with a random or don't care value immediately after being selected with the SLSO signal. In such a case the first clock edge in a frame is used by the slave to shift out the first valid data bit. The second edge latches this bit value, and so on. The last edge in a frame is used to latch the last valid data bit, which normally remains driven until the end of the frame.

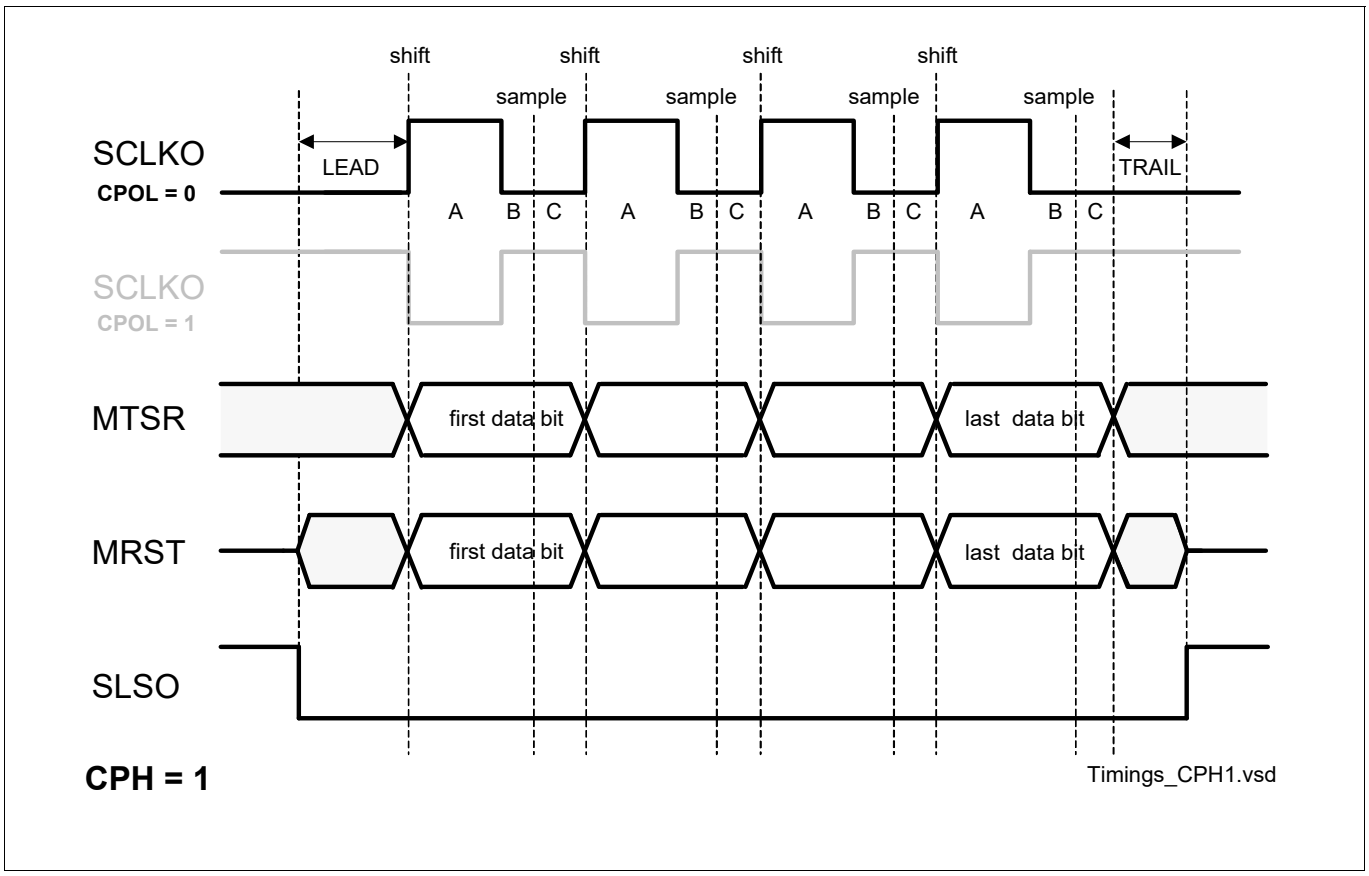


Figure 468 Timing of a transfer with CPH = 1 (Example for Data Length of 4 Bits)

The leading delay lasts between the active edge of the SLSO and the first shift clock edge.

Attention: ***If CPH = 1, then the total delay between the last edge of the SCLKO and the SLSO deactivating edge is TRAIL + ECONz (z=0-7).B + ECONz (z=0-7).C***

Attention: ***When 3 wire mode is used, and the polarity of CPH or CPOL is changing, a dummy frame is required. The pad enable to be after the dummy frame. The values of the clock and data signals are not defined before the first data is transmitted.***

37.3.2.2 Configuration Extensions

This section describes the possibilities available for configuring the phases of the QSPI communication: timing delays, data lengths, their ranges and granularity.

The QSPI module controls 16 communication channels, which are individually programmable. Each channel is associated to one slave select output and to the corresponding timing and other properties (baud rate, delays, data width, parity...).

The 16 slave select channels of the QSPI module are subdivided in two groups of 8 slave select channels:

- Channels 0 to 7
- Channels 8 to 15, channel 8 having the same **ECONz (z=0-7)** settings as channel 0, channel 9 as channel 1, and so on.

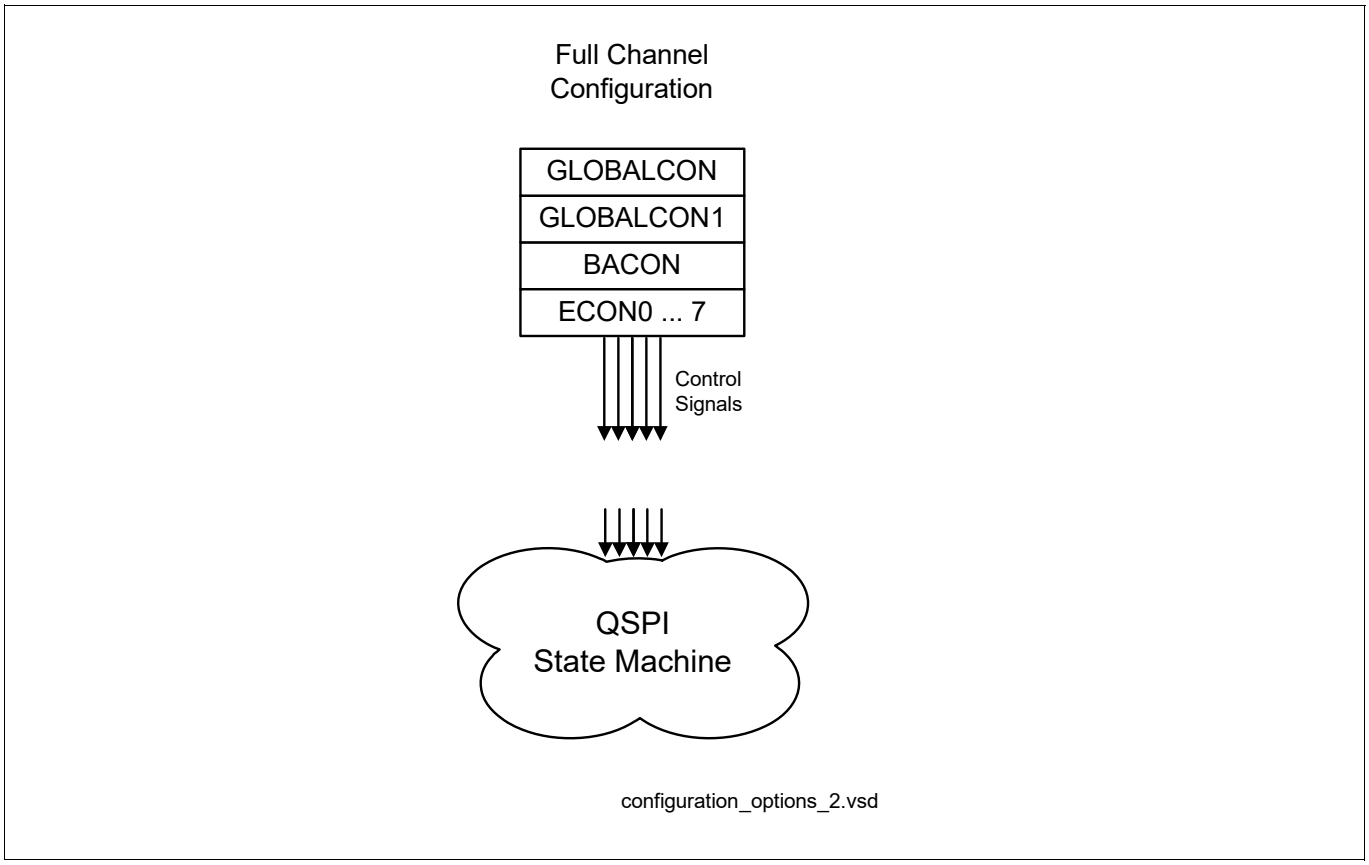


Figure 469 Flexible Configuration Concept

37.3.2.3 Details of the Baud Rate and Phase Duration Control

The QSPI phases correspond to states of a simple state machine defining a sequence of simple states and loops. Due to the strict sequential nature of the communication cycle one timer is enough to define the length of each phase. The length of each phase is defined in time quanta, and the length of a time quantum is defined by a time quantum timer. Each QSPI slave select channel has its own time quantum length, based on the module time quantum length: T_{PER} clock period -> one module time quantum -> one channel time quantum

Each QSPI module generates a module time quantum which is further downscale to a channel quantum. This affects both the delays and the baud rates. The assumption is that the pad strength setting, the capacitive load and the corresponding QSPI network layout properties do not allow extreme differences in the baud rates for the different slaves connected to one module. They all have similar speed range: high speed, or medium speed, or low speed. Mixing very high speed and very low speed slaves does not make much sense. Many slaves produce high accumulated capacitive load on the clock and data outputs, which influences the possible baud rates for all slaves. Variations in the baud rates of the slaves of one module in the range of 6:1 is possible by varying the bit segment phases in each channel. Additional two-bit counter per slave allows for additional scaling of 1, 2, 4, or 8, making a total bit-time variation of 48:1 between the channels in one module possible.

Each QSPI slave select channel has its own set of phase lengths, depending on $T_{PER} = 1 / f_{PER}$

- $T_{SCLKz} = T_{PER}$ * **GLOBALCON**.TQ * **ECONz (z=0-7)**.Q * (A + B + C)
- $T_{LEAD} = T_{PER}$ * **BACON**.LPRE * **BACON**.LEAD
- $T_{TRAIL} = T_{PER}$ * **BACON**.TPRE * **BACON**.TRAIL
- $T_{IDLEA,B} = T_{PER}$ * **BACON**.IPRE * **BACON**.IDLE
- $T_{STROBE} = T_{PER}$ * **GLOBALCON**.TQ * **ECONz (z=0-7)**.Q * **GLOBALCON**.STROBE

*Note: The equations above contain the representative decimal values of the bit-fields in corresponding units, not their actual binary bit-value. For the formulas using the actual binary values of the bit-fields, see **Figure 471** and **Figure 472**.*

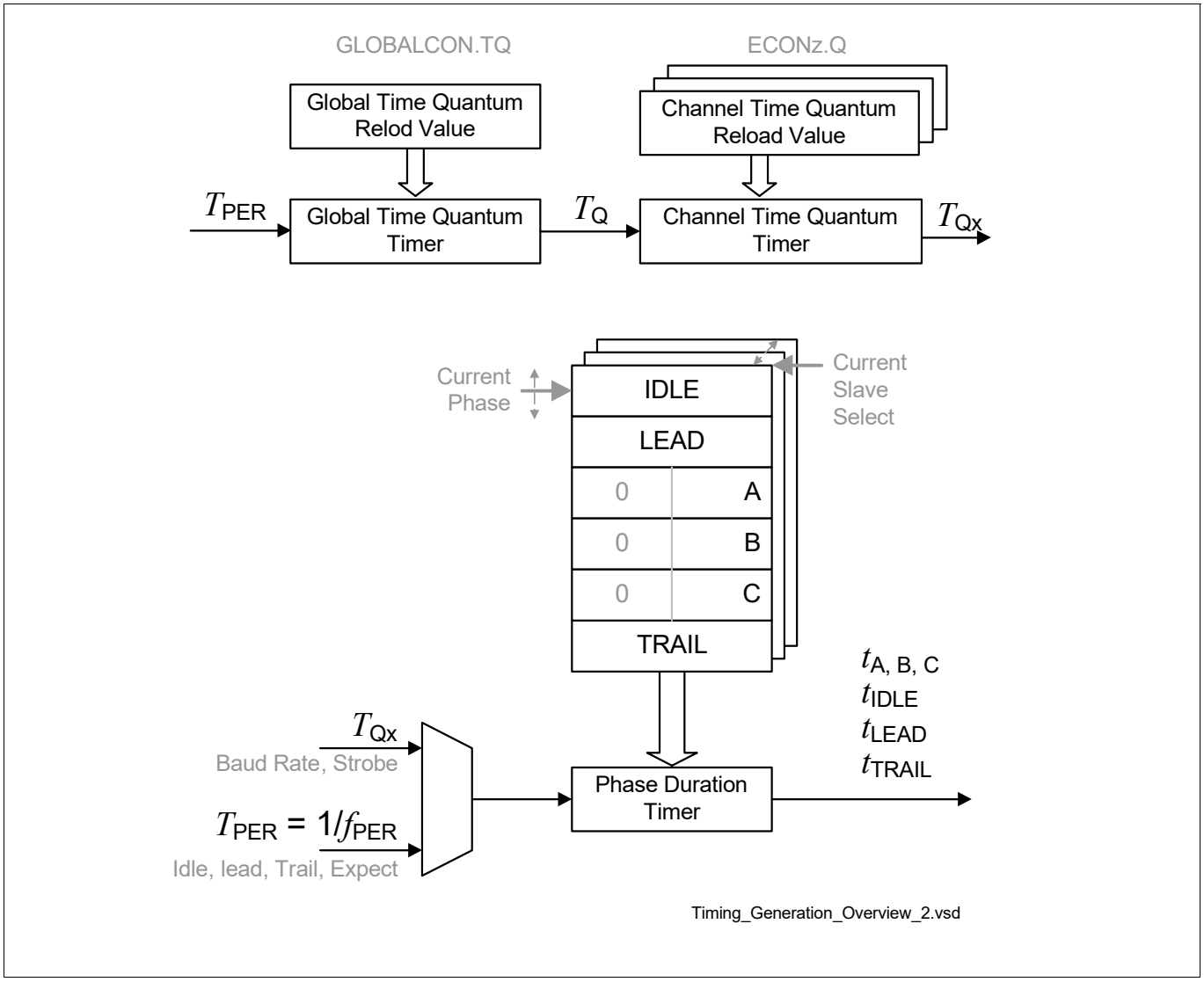


Figure 470 Phase Duration Control, Overview

37.3.2.4 Calculation of the Baud Rates and the Delays

This section gives another alternative view to the calculation of the baud rate and the duration of the delays.

The baud rate generation chain for a channel starts with a common TQ divider. The resulting global time quantum is used by the dedicated Q and A,B,C dividers to generate the baud rates per channel (slave select). See **Figure 471**.

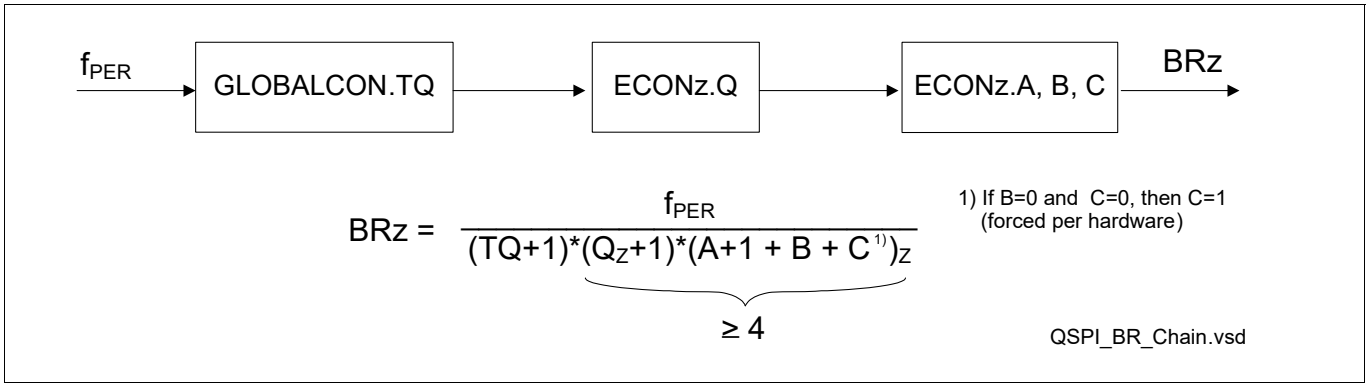


Figure 471 Baud Rate Calculation

Attention: The following condition must be satisfied: (Q+1)*(A+1 +B +C) >= 4. Or alternatively formulated: bit length must be at least 4*T_Q.

Each delay of an active channel (slave select) has a separate and independent divider chain starting with a power-of-four prescaler and an n-divider. See **Figure 472**.

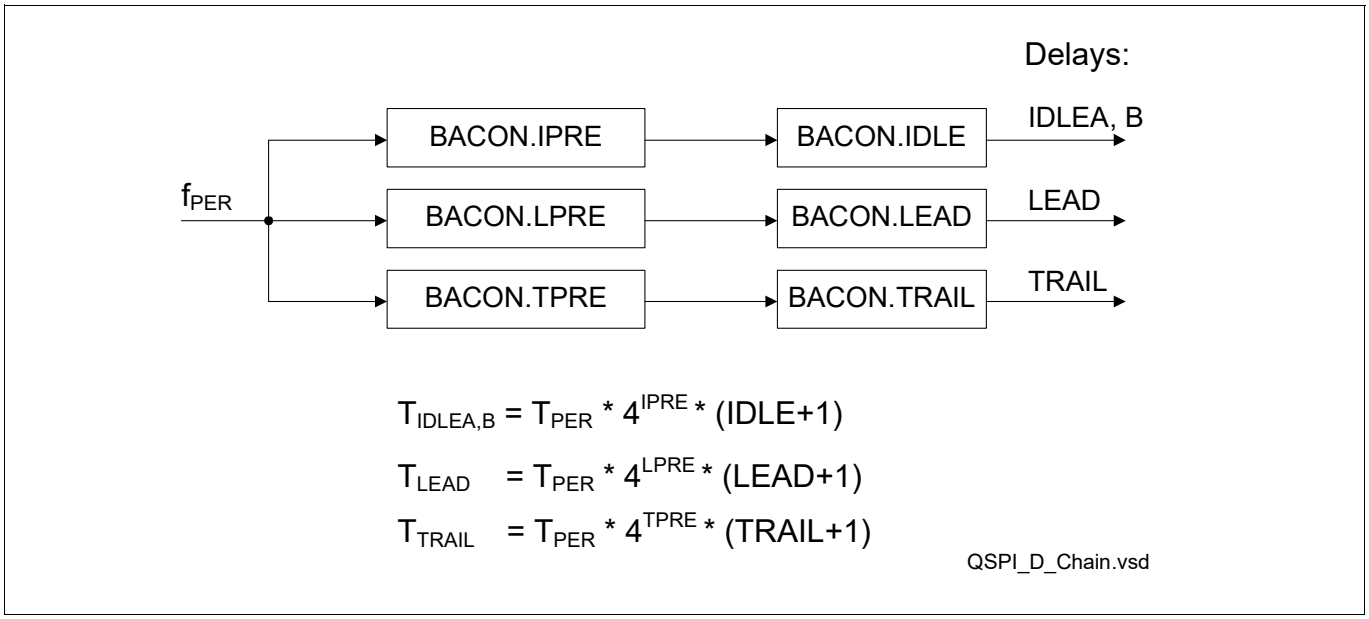


Figure 472 Calculation of the delays

37.3.2.5 State Diagram of Standard Communication Cycle

Figure 473 shows the top view of the standard communication cycle of the QSPI state machine in a hierarchical way. The “Data” state consists of a loop of several “Bit” states, where the number of repetitions is defined with the data width bit field. Each “Bit” state consists of a simple sequence of “A”, “B”, and “C” states.

A complete (or standard) communication cycle containing all phases, starting with activating the slave select and ending with its deactivation, is executed only if the bit **BACON**.LAST = 1. Otherwise, the slave select remains active and the module waits for more FIFO data entries.

In Master Mode, the Phase Transition Interrupt events, PTI1 and PTI2, each signal one of the phase transitions which are shown in **Figure 473** . The events are flagged via **STATUS**.PT1F and **STATUS**.PT2F. The values of **GLOBALCON1**.PT1 and **GLOBALCON1**.PT2 determine which of the transitions cause the events. The events will also cause interrupts if **GLOBALCON1**.PT1EN and **GLOBALCON1**.PT2EN are set to 1. The event flags are cleared via the programming of the **FLAGSCLEAR** register.

For the Move Counter Mode, see **MC**.

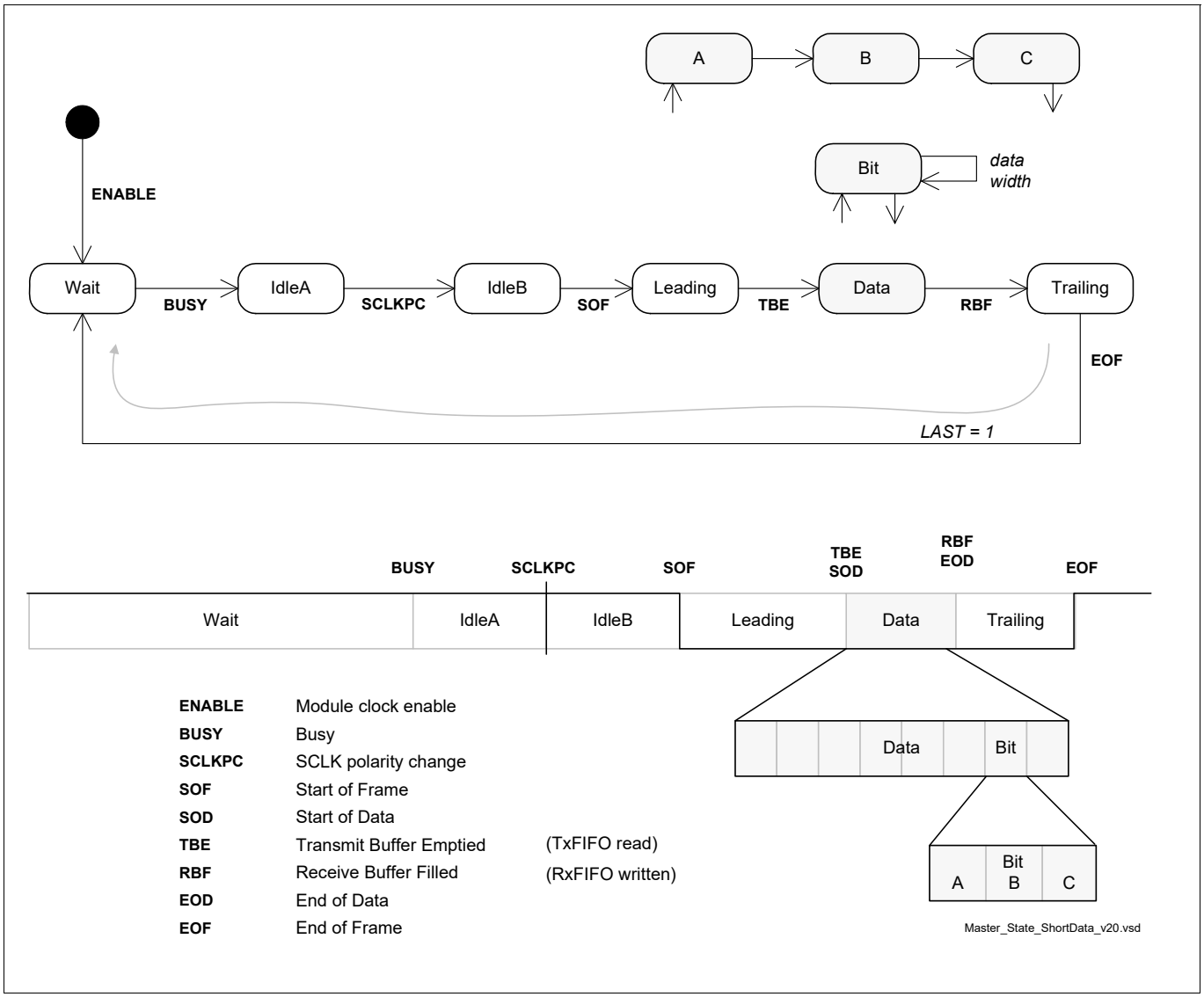


Figure 473 Standard Communication Cycle of the State Machine (Short Data)

(>>**Interrupts**)

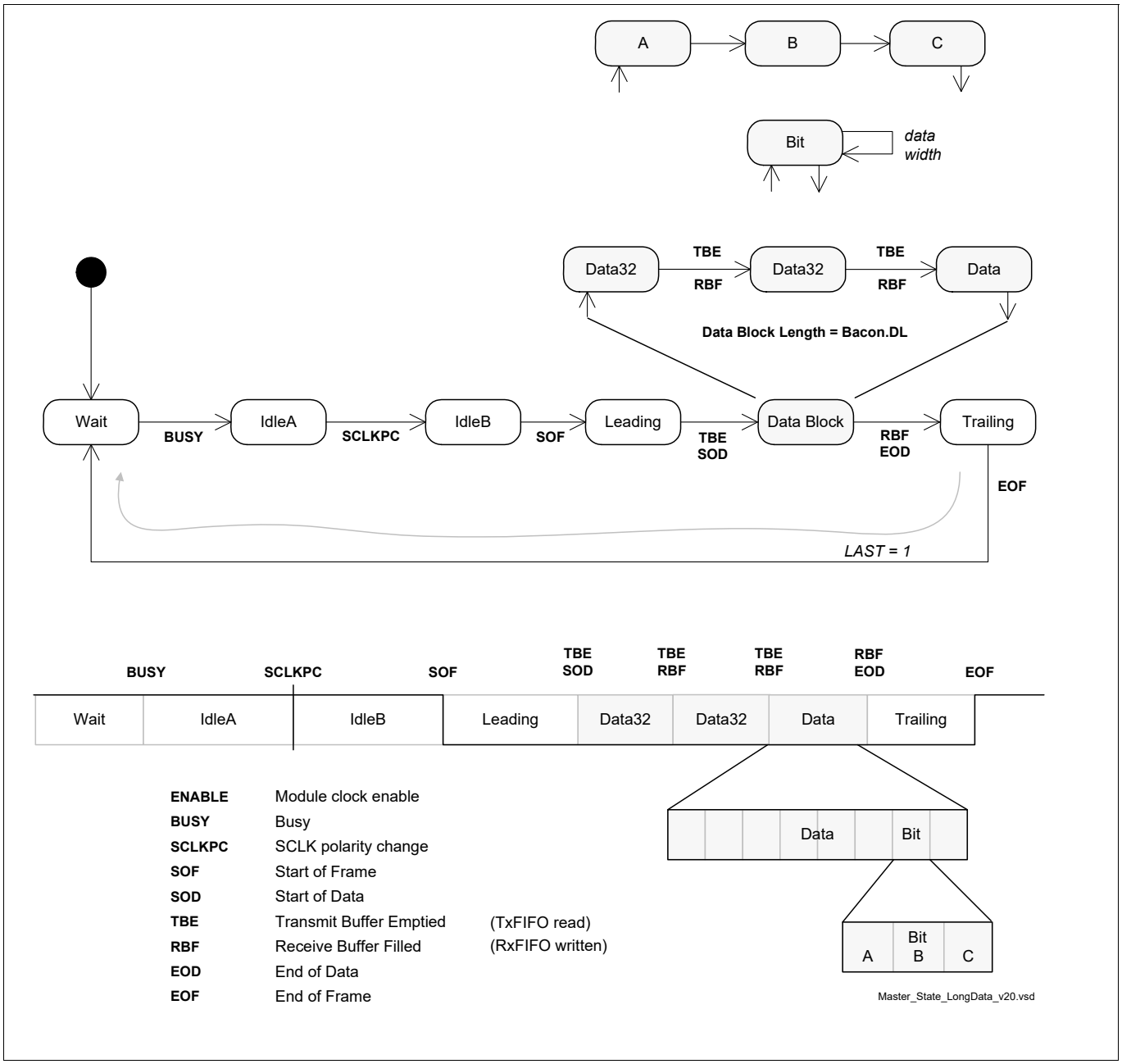


Figure 474 Standard Communication Cycle of the State Machine (Long Data)

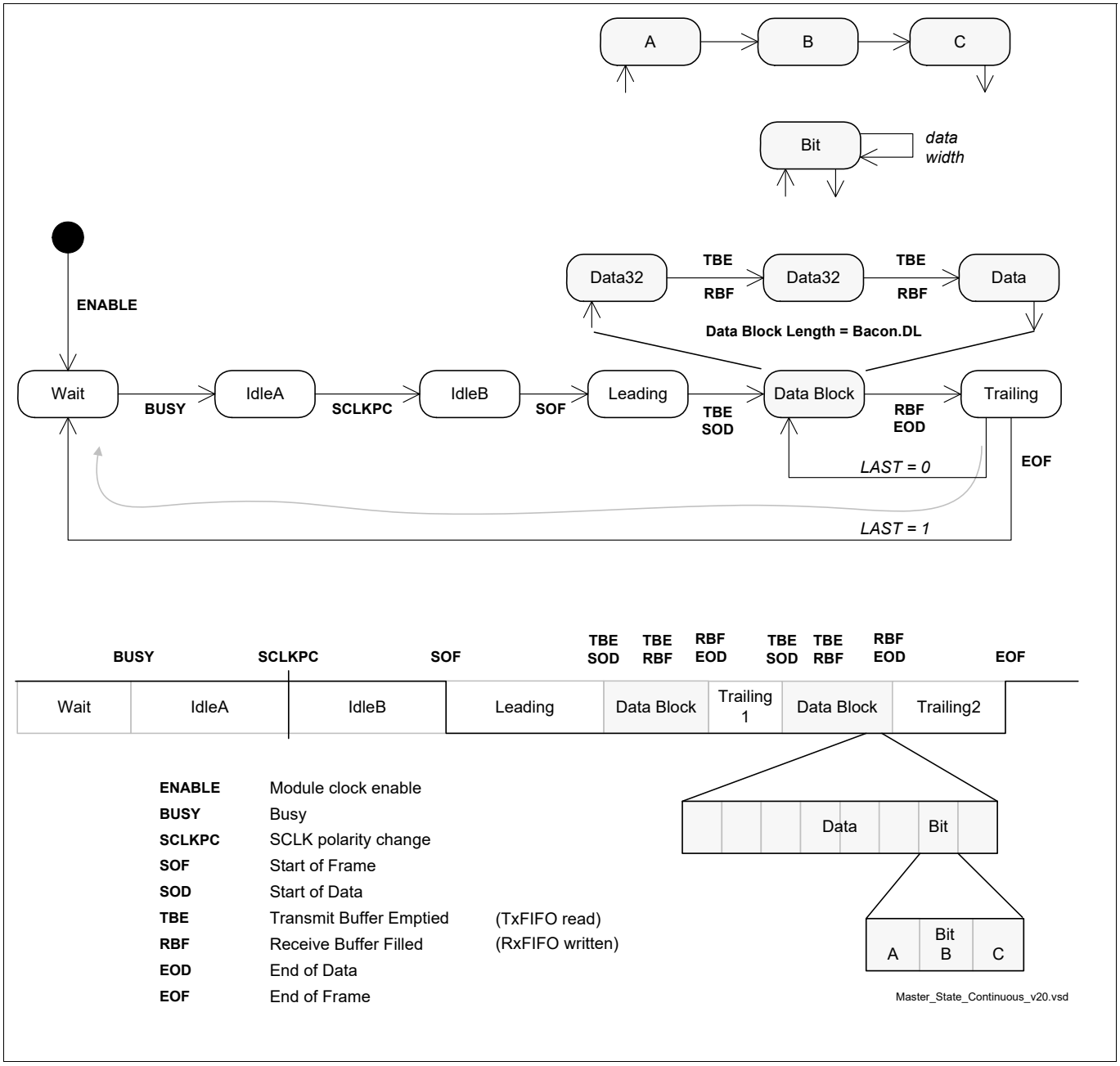


Figure 475 Standard Communication Cycle of the State Machine (Continuous)

"Data Block" in **Figure 475** refers to either short data or long data block. When concatenating data blocks in continuous mode, each block can be configured independently as short or long, according to its **BACON** configuration.

37.3.2.6 Expect Phase

Figure 476 shows the EXPECT state in the QSPI state machine diagram. This state can be entered in the long data mode and in the continuous mode, where one frame consists of more than one TXFIFO entries. Its purpose is to provide monitoring of the latencies on the FPI bus. In this case, after one FIFO entry has been shifted out and the module expects more bits to shift, according to the **BACON**.DL, a time-out counting starts. If the expected FIFO entry does not come in time, a time-out occurs.

The duration of the EXPECT state is zero if at the end of the current DATA state the new data is already available in the FIFO.

The duration of the EXPECT state can be anywhere between zero and the programmed upper limit, if the next data is written into the TXFIFO any time within the allowed time window. In this case the current frame continues with the SLSO signal remaining active.

The duration of the EXPECT state is exactly maximum if the data did not come in time. Then the current frame is stopped, but the SLSO is not deactivated. A TO (Time - Out) interrupt is raised. The state machine continues the EXPECT state in a loop, waiting for the software intervention, generating an interrupt signal after each expect time interval.

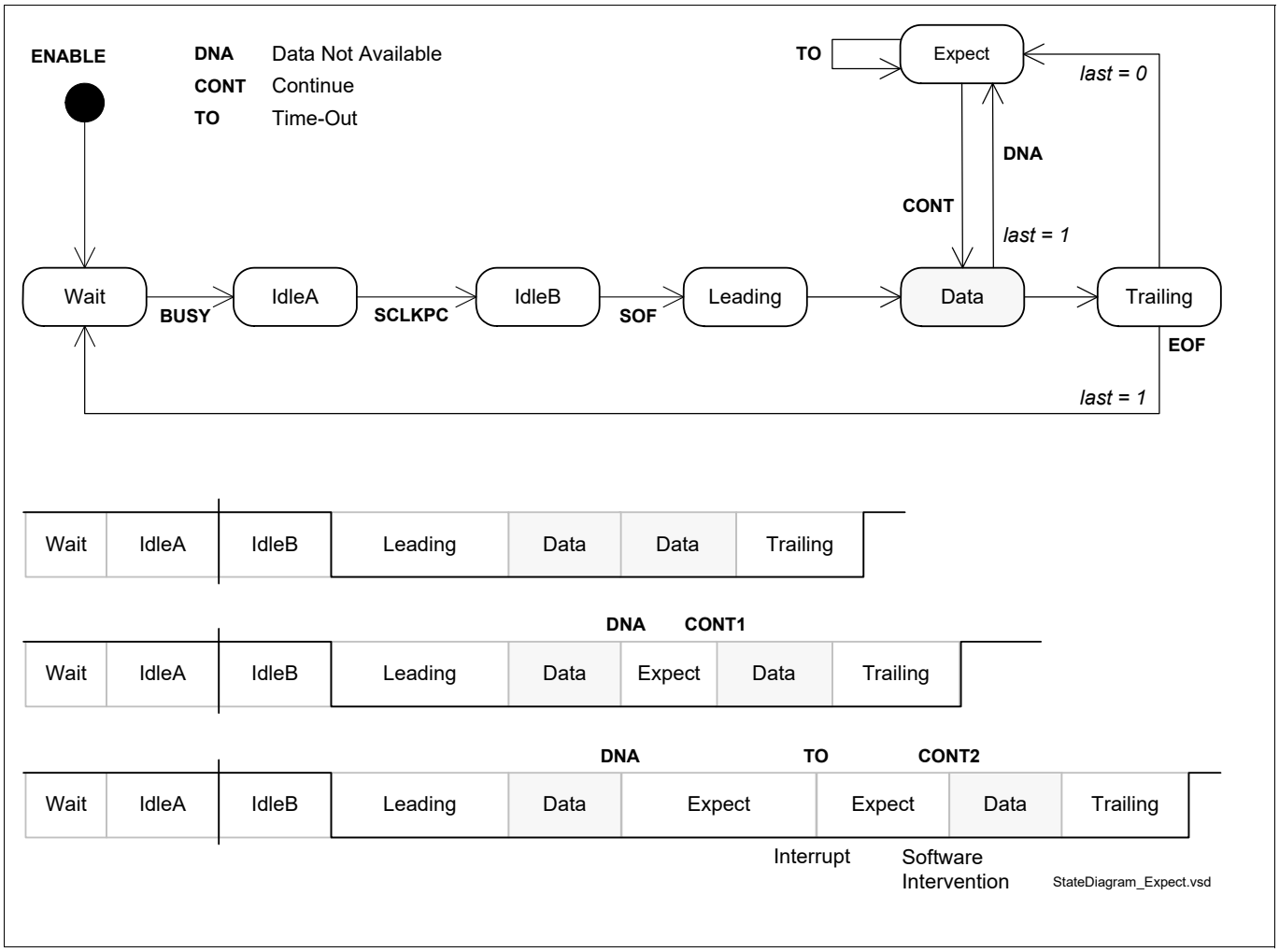


Figure 476 EXPECT Cycles of the QSPI State Machine

37.3.2.7 External Slave Select Expansion

In this mode the bit field **BACON**.CS drives directly the SLSO1 to SLSO4 signals, not demultiplexed by the QSPI, but chip-externally. **SSOC**.AOL bits can invert the SLSO0...4 signal levels individually; **SSOC**.OEN bits enable the signals to the output pins. External Slave Select Expansion Mode (Delayed Mode for SLSO0) is activated with the bit **GLOBALCON**.DEL0.

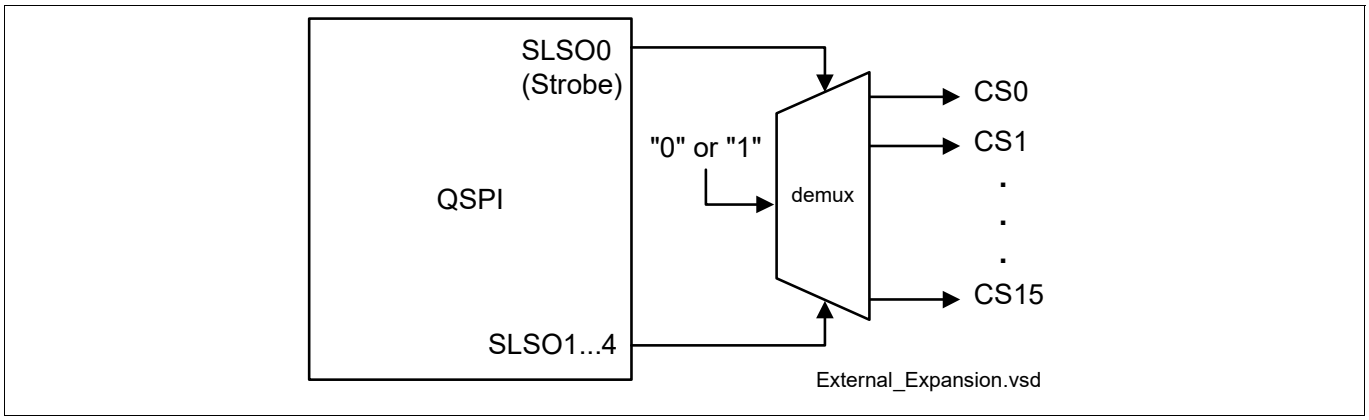


Figure 477 Demultiplexing the Slave Select Signals Externally

In order to ensure glitch free selection, a strobe signal is provided, driven at SLSO0 pin. This signal is delayed relative to the SLSO1...4 signals for LS (Lead_Strobe) and TS (Trail_Strobe) delays, equal in duration and configurable in the range of 1 to 16* f_{PER}. The duration of the LS and TS delays is set in the **GLOBALCON**.STROBE.

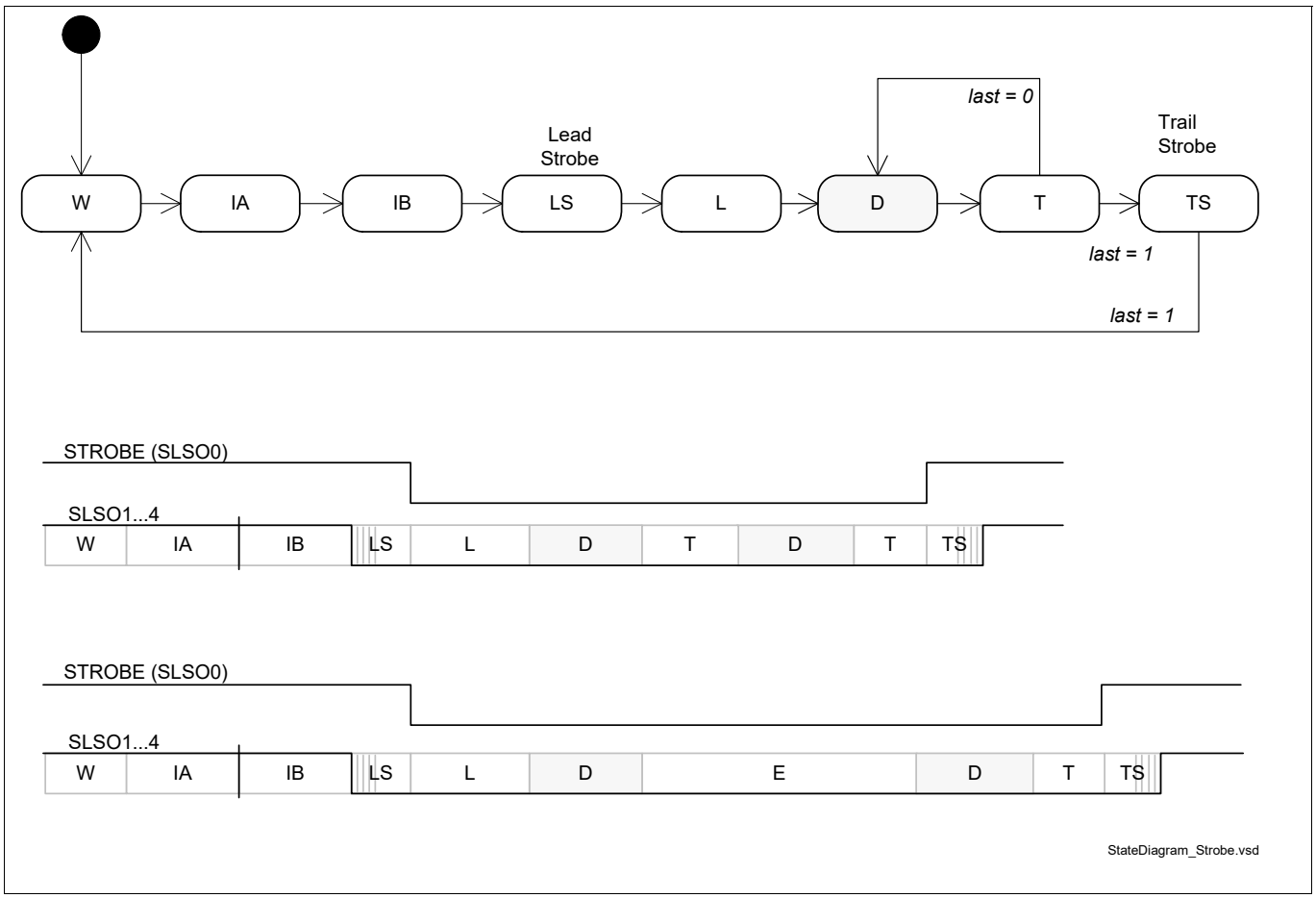


Figure 478 State Machine and SLS Signals in Strobed Mode

37.3.3 Slave Mode

The slave mode is entered by setting the bit field **GLOBALCON**.MS to value 1x. Before entering the slave mode, the related parameters must be configured, including pin selection for SCLKI, SLSI and MTSR, pin input level, and pull-up/pull-down configuration. The configuration should be done before data transmission starts. Reconfiguration is only allowed when the module is in a wait state. Subsequently state machine reset should be performed and afterwards the slave mode should be entered.

In slave mode data is being pushed into or out of the module through the serial pins by an external master. The RXFIFO must be regularly emptied by the system in order to avoid data loss, and the TXFIFO must be regularly fed by the system to avoid delivering all “1” data in case of empty TXFIFO.

If the slave mode relies on a slave select signal to mark the start and, in parallel to the bit counting, the end of a frame, then deactivating the slave signal by the master automatically resets the shift register state machine to wait state.

If the slave mode relies solely on bit counting for determining the frame end, then any SCLKI clock period longer than two bit times is treated as an end of a frame and simultaneously as a baud rate error.

In slave mode, the module simply immediately responds to external clock edges. Therefore, the settings for the leading, trailing, idle delays, and duty cycle and sampling point are irrelevant. From all timing settings, only the data length and the baud rate divider setting are relevant. The baud rate setting is important because of its role of watchdog timer of the incoming serial clock.

Generally, the slave mode supports the same user interface as the master mode. This means the BACON-Data sequences remain the same. The SRF bit is ignored.

Limit the data length setting in the slave mode to the range of 2 to 32 bits. The byte setting **BACON**.BYTE is to be set to zero, that is, it always defines bits.

In case of receive only (simplex receive) the user software must reset the module in order to switch to transmit and receive mode (full duplex). Generally, mode reconfiguration (slave mode to master mode) requires module reset which resets the internal state machines and counters.

In case of slave transmit, the user software must write data to the TXFIFO before the first SCLKI edge of a frame arrives. If the TXFIFO is empty at this point of time, an underflow occurs, all “1” data is delivered, and underflow error interrupt is triggered if enabled. If the TXFIFO is written at the same time when the first SCLKI edge starts the shifting, the underflow occurs, interrupt will be triggered and data from the TXFIFO will be transmitted possibly corrupted - the first bit will be “1”.

In case of TXFIFO / RXFIFO underflow the fifos deliver all “1” data.

In case of TXFIFO / RXFIFO overflow, the last data is lost. In such case data or optionally status in the RXFIFO can be lost.

The bit counting in the slave mode operates according to the following rules:

- The incoming bits (or shift clock shift edges) are counted and when the number defined in the **BACON**.DL is reached, the bits are transferred in the RXFIFO
- If the SLSI input gets deactivated, the internal bit counter is reseted.

In slave mode, sleep requests may be enabled only when there is no pending transmit queue and no pending transmission and the TXFIFO is empty.

Phase Interrupts in the Slave Mode

PT1 interrupt signals the following events:

- BUSY -> option 000 from GLOBALCON1.PT1. Transmit data is present and is waiting for the shift clock.
- SOF -> option 010 from GLOBALCON1.PT1. Transmission of the first data bit started. It is not connected to SLSI active edge, but to first clock pulse received.

- TBE -> option 011 from GLOBALCON1.PT1. Data is taken from the TXFIFO by the QSPI shift engine. It will be transmitted with the first shift clock pulse.
- RBF -> option 100 from GLOBALCON1.PT1. Received data is written to the RXFIFO.
- EOF -> option 101 from GLOBALCON1.PT1. The last data bit has been received.
- DNA -> option 110 from GLOBALCON1.PT1. Transmit data not available at the first shift clock edge reception.

PT2 interrupt signals the following events:

- EOF -> option 101 from GLOBALCON1.PT2. SLSI deactivated (rising edge on the SLSI pin)

37.3.3.1 Shift Clock Phase and Polarity in Slave Mode

In slave mode the shift clock phase and polarity are fixed to CPH=1 and CPOL=0 (have to be programmed accordingly in the **ECONz (z=0-7)** register selected by **BACON**.CS) and slave select polarity is fixed to low active. The baud rate that the slave expects is defined in the **ECONz (z=0-7)** register pointed at by the **BACON**.CS.

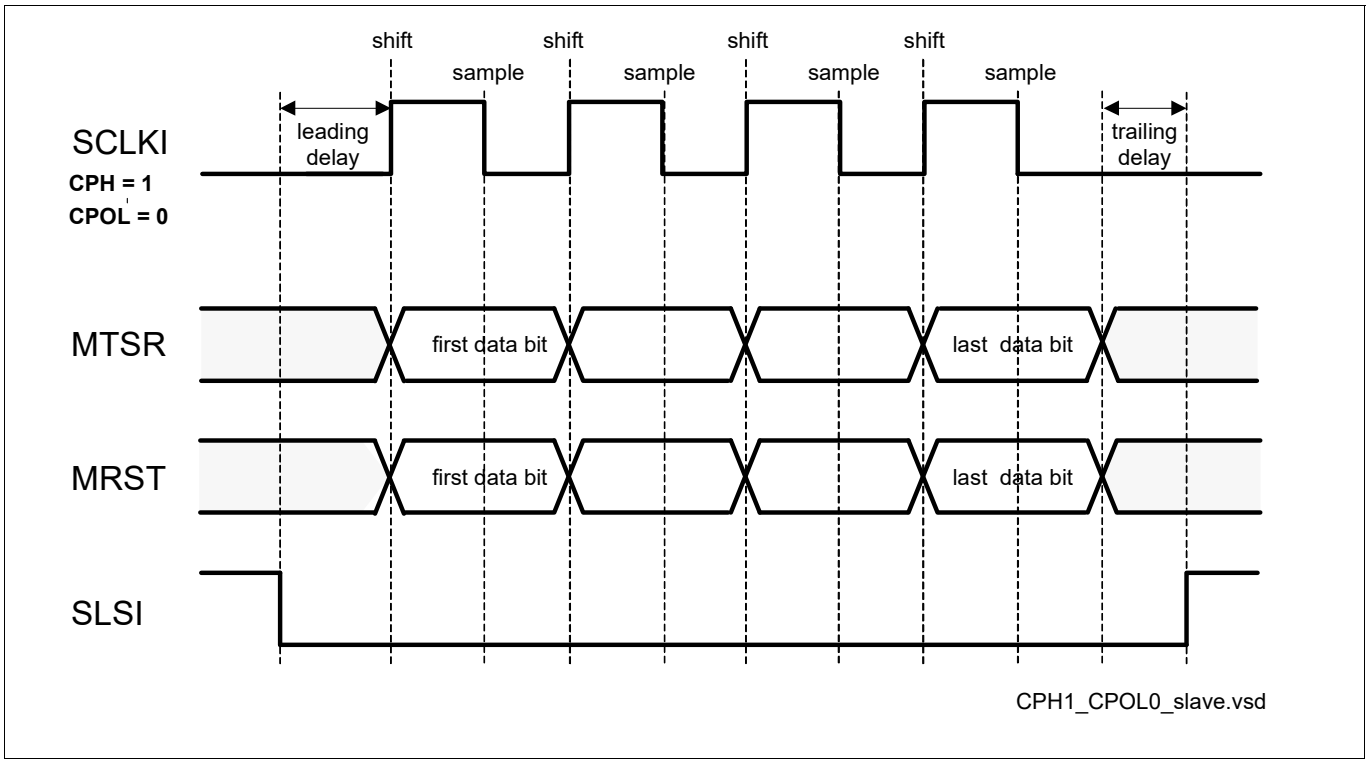


Figure 479 Slave transfer, CPH = 1, CPOL = 0

Attention: Slave re-configuration must only be performed in a gap between the frames. However, as a general recommendation, slave mode configuration should be done once only and before the transition starts.

37.3.3.2 Shift Clock Monitoring

The QSPI module provides two mechanisms for monitoring the shift clock input signal in slave mode. These mechanisms monitor:

- if the receiving shift clock frequency (baud rate) lies within a certain range and
- if there are spikes on the shift clock input.

In case such disturbances are detected, the reaction is always the same:

- the corresponding error flags in **STATUS**.ERRORFLAGS are set, which can be cleared with **FLAGSCLEAR**.ERRORCLEARS
- the common error interrupt is raised if enabled in **GLOBALCON1**.ERRORENS and
- the bit **GLOBALCON**.EN is automatically cleared, if this feature is enabled by the bit **GLOBALCON**.AREN. The user software can activate a reset afterwards

In case of too high baud rate error, baud rate detection and spike detection both react, where spike detection is more effective. Spike detection reacts immediately, baud rate detection requires several bits with higher baud rate. Baud rate detection uses a window two nominal bit times wide and counts the real bit times: double baud rate error requires at least four real shift clock pulses to be detected, triple baud rate error requires six and so on.

Baud rate detection does not react to all spikes smaller than one kernel clock period.In case of too low baud rate, baud rate error detection reacts immediately, spike detection ignores this case.

37.3.3.2.1 Baud Rate Error Detection

If the shift clock has less than half or more than double the expected baud rate, then:

- the corresponding flag is set in **STATUS**.ERRORFLAGS
- error interrupt is raised if enabled in **GLOBALCON1**.ERRORENS
- automatic clear of **GLOBALCON**.EN follows, if enabled by **GLOBALCON**.AREN.

37.3.3.2.2 Spike Detection

Short spikes on the clock line due to ground bouncing or clock ringing due to over/undershoots would cause one or more extra bits to be written to the front end receive FIFO of the slave. This extra bit, if no measures are taken by the software, would cause all the subsequent received frames to be corrupted (shifted).

The spike detection mechanism operates up to 50 MBaud and checks if a bouncing spike has occurred on the clock line near to the correct receiving clock edges.

If a spike is detected, the software should use **GLOBALCON**.RESET bit field, and should clear the corresponding status flags. Alternatively a complete module reset may be performed via registers **KRST0**/**KRST1**.

The spike detection is based on two mechanisms:

- detection of two close writes to the front end receive FIFO by detecting filling level of two
- using an inverted watchdog set to ca. 75% to 80% SCLKI clock period, which is the optimal range.

The inverted watchdog is a timer which starts to count triggered on the write event to the front-end receive FIFO of the slave. During the time the watchdog counts, no edge is allowed to occur. An edge in the forbidden time raises a spike error event.

If a spike occurs in the last 25% of the SCLKI period, when the watchdog has stopped counting, the watchdog will be triggered to count by the spike. The next regular edge will violate the forbidden time and raise a spike event.

The watchdog counts **ECONz (z=0-7)**.B + **ECONz (z=0-7)**.C time quantas. The following constraint applies: 50% $*T_{SCLKI}$<**ECONz (z=0-7)**.B + **ECONz (z=0-7)**.C <100%$*T_{SCLKI}$.

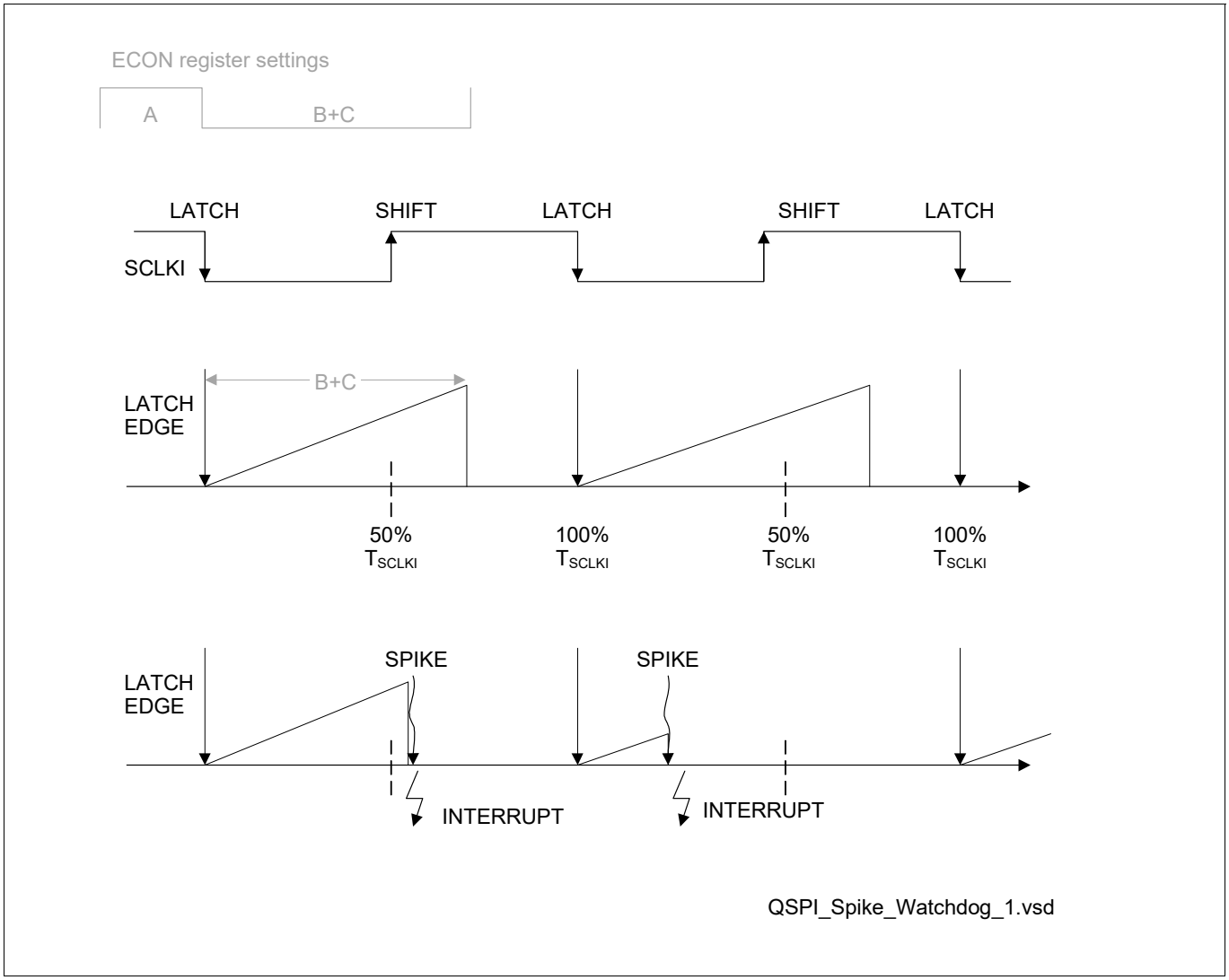


Figure 480 Spike Watchdog

37.3.3.2.3 Shift Clock Monitor Flags

The spike detection mechanism sets the flag **STATUS1**.SPD, if it has detected a spike. The baud rate error detection mechanism sets the flag **STATUS1**.BRD, if it has detected excessive baud rate deviation.

Writing 1 to SPD and BRD sets the bit and causes an interrupt, if enabled. Writing 0 has no effect.

Both signals are ORed. This combined signal is shown in **STATUS**.ERRORFLAGS and raises an interrupt, if enabled in **GLOBALCON1**.ERRORENS. Automatic clear of **GLOBALCON**.EN follows, if enabled by **GLOBALCON**.AREN.

If the **STATUS**.ERRORFLAGS[2] bit is cleared via **FLAGSCLEAR**.ERRORCLEARS[2], both flags SPD and BRD are automatically cleared.

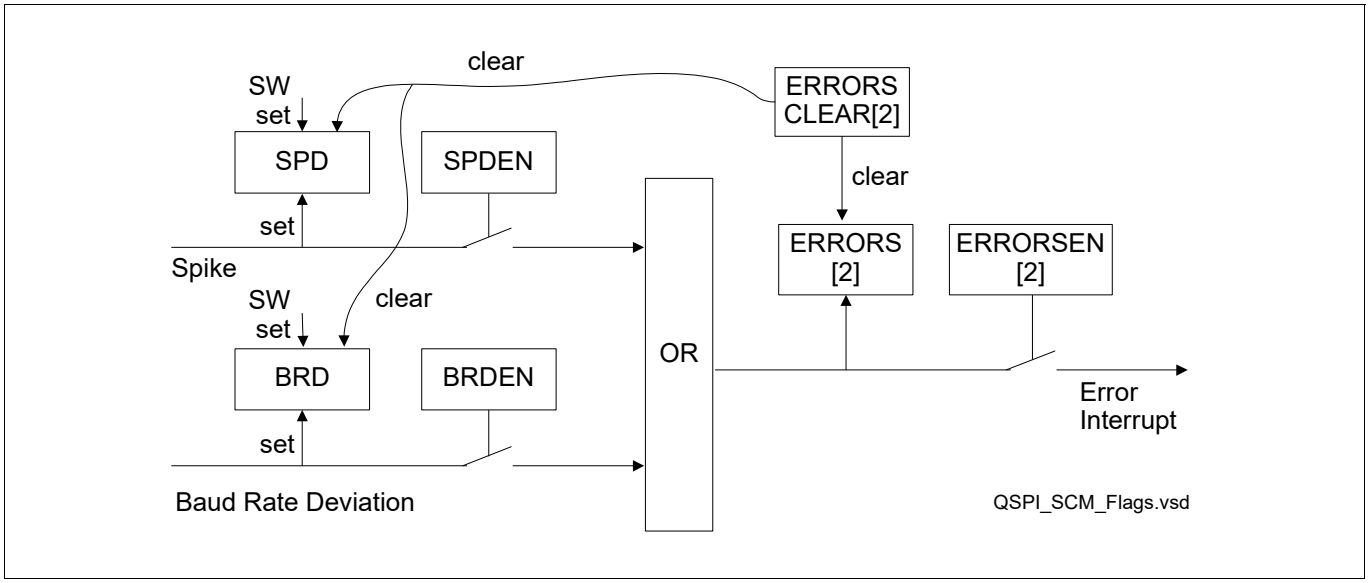


Figure 481 Shift Clock Monitor Flags

37.3.3.3 Parity

Parity settings for transmit and receive are defined by the corresponding bit fields in the **BACON** register.

The parity bit is concatenated to the data bits at transmission time. That means that a frame containing 16 bit data is 17 bits long (data + 1 parity). The transmitted parity bit can be read from the **STATUS**.TPV bit.

The parity bit is removed from the received bits at receive time automatically. The received parity bit can be read from the **STATUS**.RPV bit.

Attention: Parity is available only in short mode, that is, only for data of up to 32 bits length.

Attention: The parity bit is concatenated to the data at the LSB location. If the bit field BACON.MSB = 0 it is shifted out first, else it is shifted out last.

37.3.4 Operation Modes

Generally, the QSPI module transmits a RAM queue of an arbitrary length, fed by a DMA module, and makes pauses between the queues. The QSPI module does not know anything about the queue length or the current position within the queue or the future schedule of the messages that may come.

If the QSPI module gets a request for switching off the clock or pausing the module, it disables all service request lines.

The QSPI module provides two options for switching off the clocks or pausing the module: hard suspend (as soon as possible regarding for example the length of interrupt request pulses), and soft suspend (transition to the PAUSE state after reaching predefined state). The requests can be triggered by hardware signals (disable, suspend, sleep) or by software write to **GLOBALCON**.EN bit.

Disable, Pause and Run

After being enabled by clearing the **CLC**.DISR bit, the QSPI enters the PAUSE state. In this state, the QSPI module can be initialized, the TXFIFO pre-filled, and then, by setting the **GLOBALCON**.EN bit, put into a RUN state.

The software requests PAUSE state of the running module by clearing the **GLOBALCON**.EN bit. The software can detect that the QSPI module has reached the PAUSE state by polling for the **GLOBALCON**.EN=0 and **STATUS**.PHASE=0 (indicating WAIT state, master mode only).

In PAUSE state, the user interface of the module is active. The registers and the FIFOs can be read and written. However, both the receive and transmit state machines are inactive, in master as well as in the slave mode. This allows for reconfiguration of the module without affecting the serial bus.

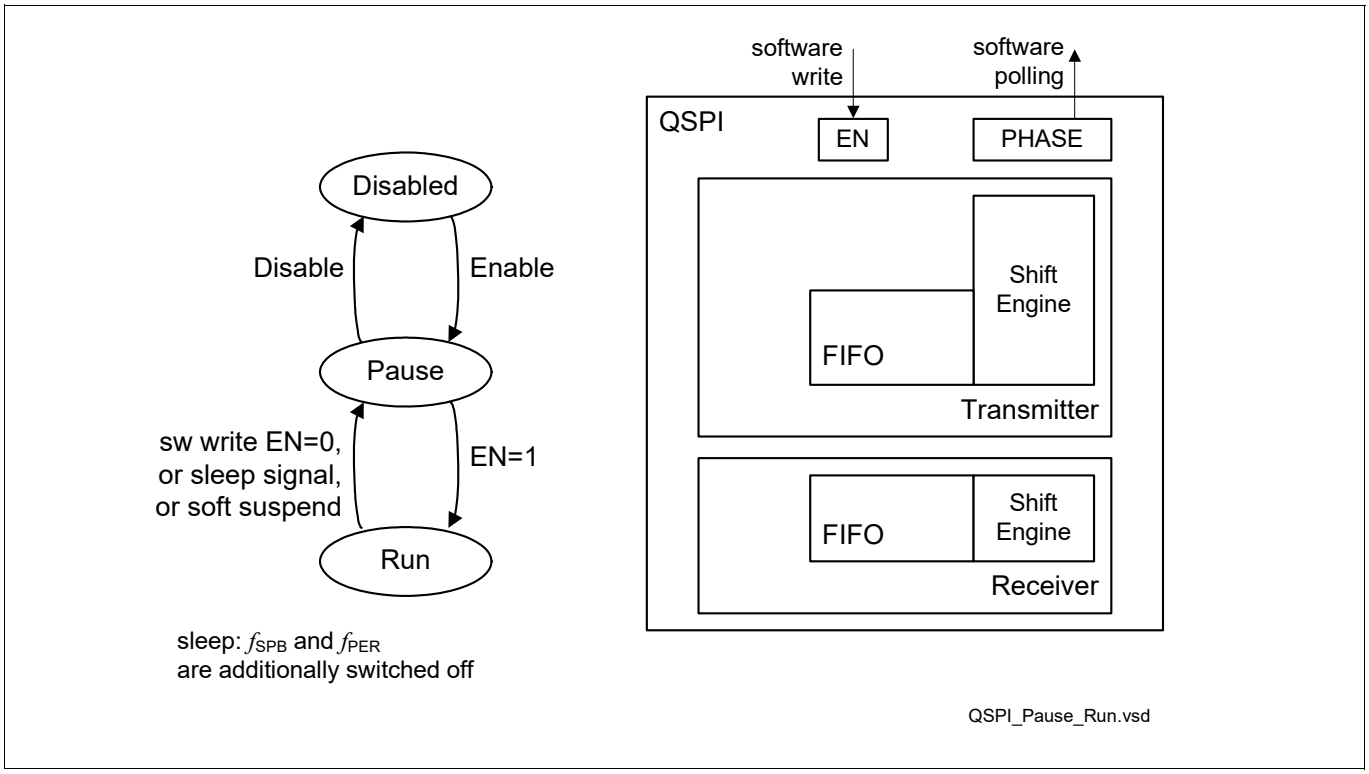


Figure 482 Pause and Run Overview

The concept of the PAUSE state is reused for handling the soft OCDS suspend and Sleep requests. In contrast to active Pause state and OCDS suspend, the sleep state has the special property that (due to the low power requirement) both FPI and QSPI clocks are switched off.

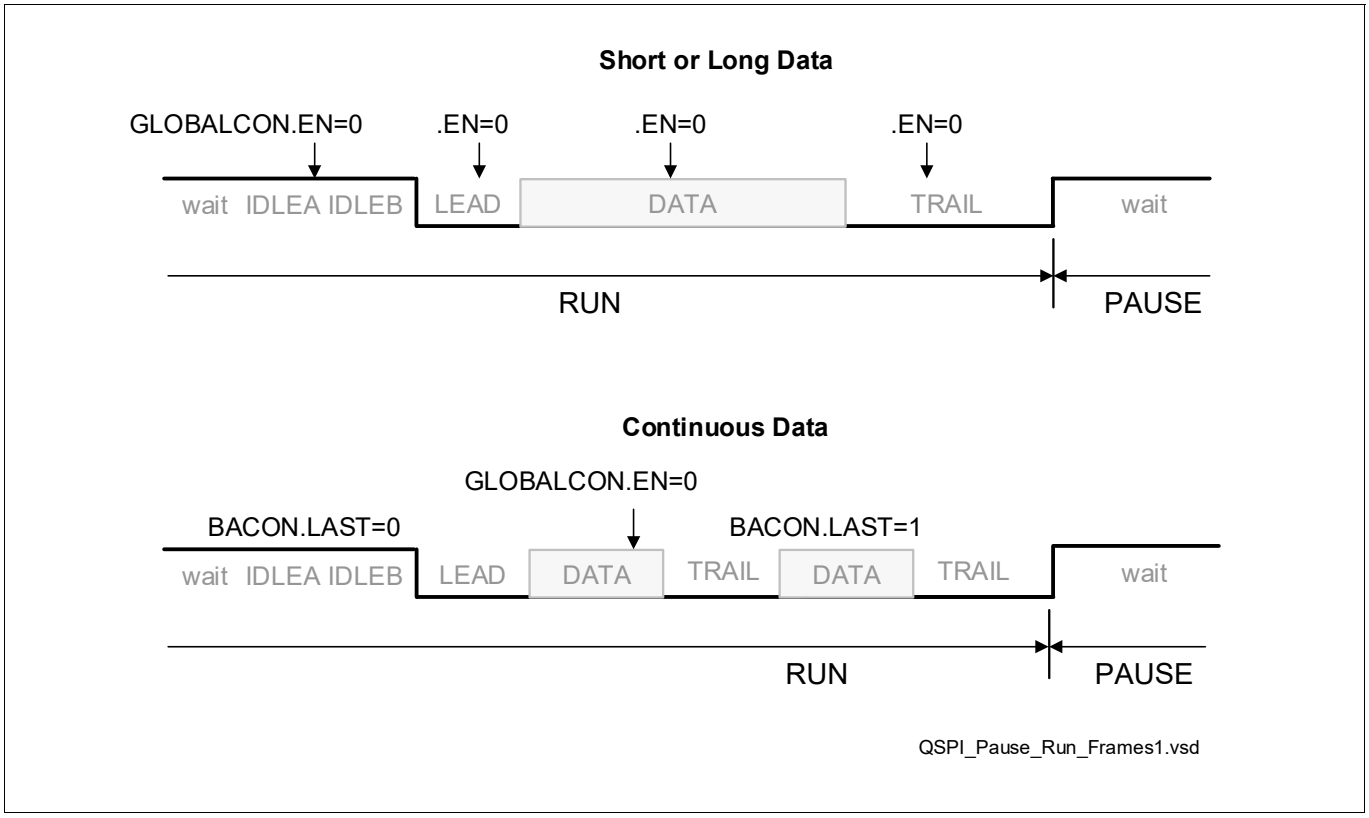


Figure 483 Entering the Pause State With GLOBALCON.EN=0

If a running module receives a request to PAUSE, it waits for the end of the last TRAIL phase of a frame.

For all modes (Short, Long and Continuous), the PAUSE state is indicated by **GLOBALCON**.EN=0 and **STATUS**.PHASE=0, master mode only.

37.3.4.1 OCDS Suspend

The OCDS hard suspend request, for debugging purposes, simply freezes the QSPI module in the current state. No signal is changed including the service request lines, which remain frozen in the current state.

The QSPI responds to an OCDS soft suspend request by entering the PAUSE state immediately after the subsequent end of a trail phase, and then sending acknowledge (see **Figure 484**). The behavior is the same as for entering the **Sleep Mode**.

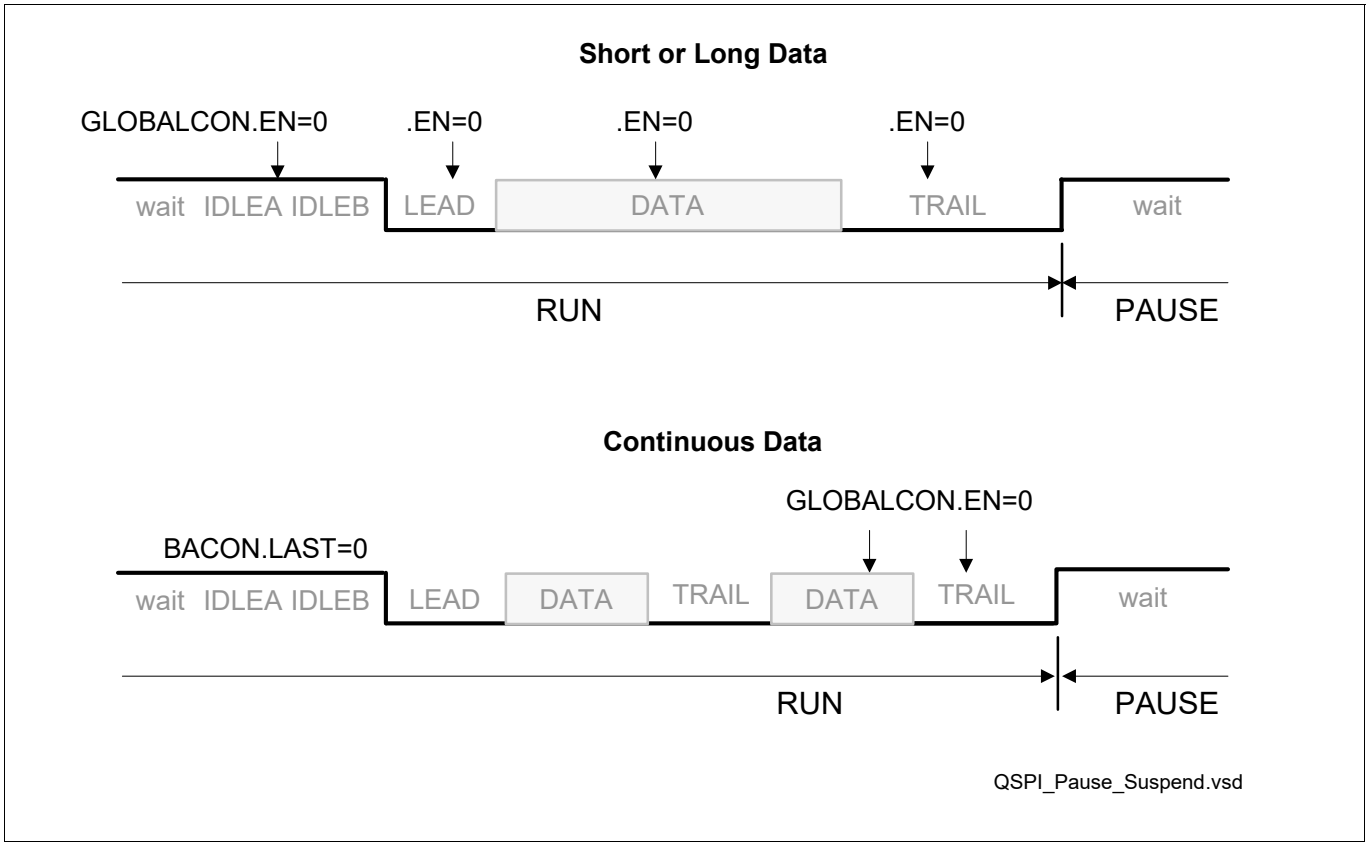


Figure 484 Entering the Sleep, Soft Suspend, and Disable State

Reading the FIFO entries by the Cerberus (detected by the FPI master tag) does not modify the FIFO content nor the read and write pointers.

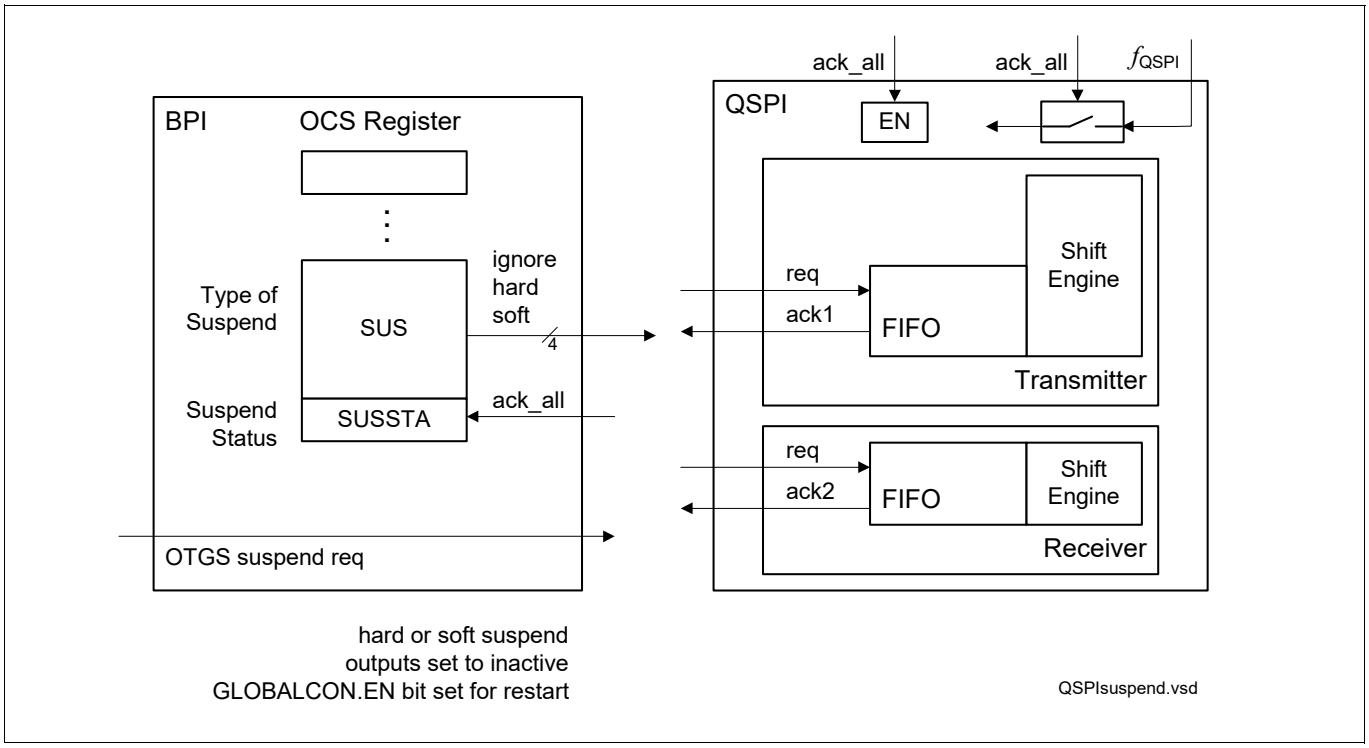


Figure 485 Suspend Overview

Please note the following:

- In Hard Suspend Mode:
  - the QSPI kernel clock is switched off immediately
  - Writing to registers is possible but will enable the kernel clock for a few cycles. Attention: these clock pulses in Hard Suspend Mode can have unintended side effects like signals becoming and staying active. This can affect also other modules, so a QSPI kernel reset might not be sufficient to bring the system into a defined state
  - Read access to a register does not let through any clock cycles, but is done in combinatorial way. Therefore, RXFIFO pointer does not move
- In Soft Suspend Mode:
  - the QSPI kernel clock is switched on, the module is only paused
  - Each read and write can cause state transition in some state machine in the module. Therefore RXFIFO reacts to the read accesses.

37.3.4.2 Sleep Mode

Sleep signal is a system wide hardware signal requesting from all modules to go to low power saving state as soon as possible. For the QSPI module, going to sleep mode can be disabled or enabled by setting or clearing the **CLC**.EDIS bit. The module enters the sleep as soon as possible and remains in this state as long as the hardware sleep signal is active. Upon deactivation of the sleep signal, the QSPI wakes in PAUSE state (**GLOBALCON**.EN bit is cleared by the QSPI module) and waits for the software to enable the **GLOBALCON**.EN bit.

If **CLC**.EDIS bit is cleared, the sleep mode is enabled and the module enters the sleep state:

in master mode

- after ending the TRAIL delay if a transmission is in progress
- or immediately if the module is in WAIT state

in slave mode

- after the bit counter has counted the full predefined number of bits as defined in GLOBALCON.DL field
- immediately if the module is in wait state, no transmission ongoing and shift register empty and TXFIFO empty
- or immediately if the SLSI is inactive (the one selected by PISEL)

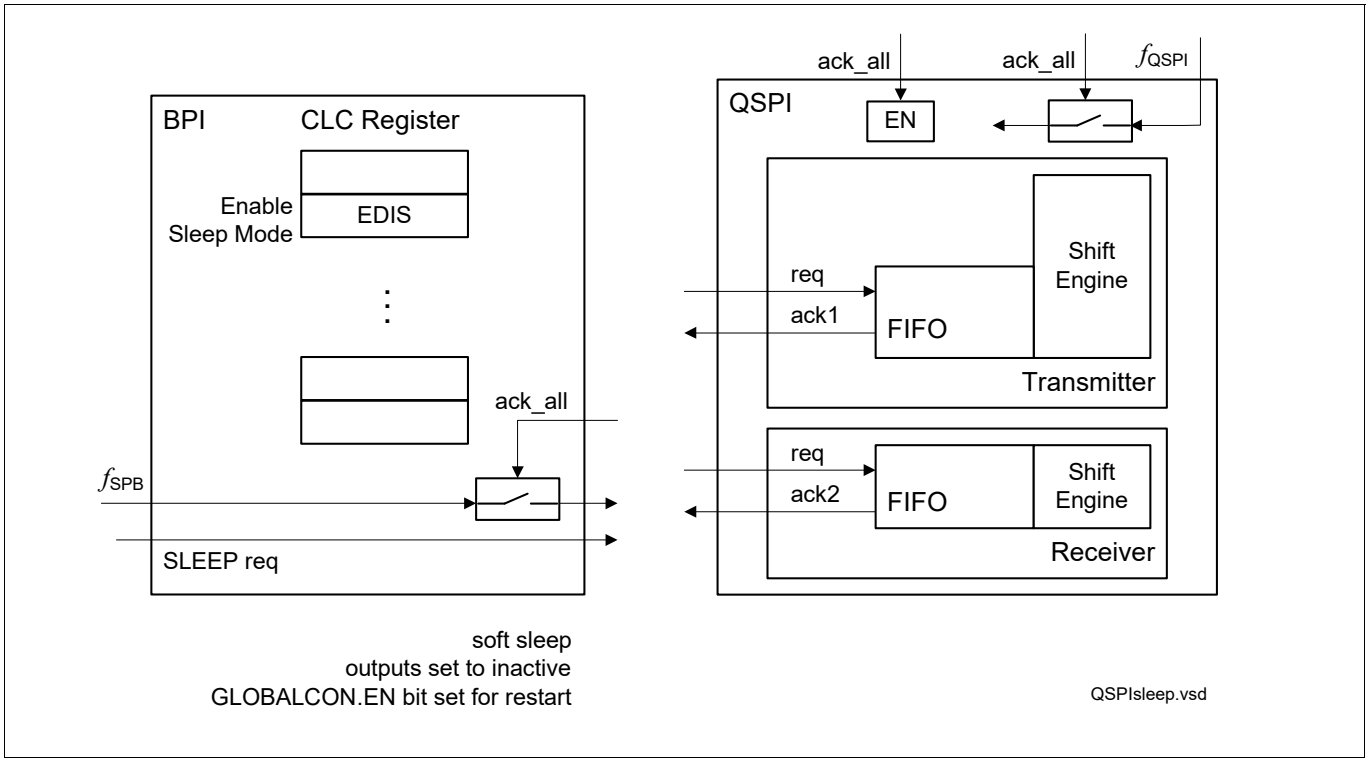


Figure 486 Sleep Overview

It is responsibility of the user software to have the sleep enabled in a safe time intervals, where no race conditions or pipeline effects between the QSPI module and the DMA can occur in case of sleep request.

Note: CLC disable request and sleep request block immediately kernel register write accesses. This includes also the TXFIFO access. Therefore if the module expects more data (like in long or continuous mode), it will enter the expect phase and remain there and never do an acknowledge and enter the required state. Periodic timeout interrupts will be generated. Therefore it is not recommended to request a sleep or disable during an ongoing long or continuous transfer.

37.3.4.3 Disabling the QSPI

The software can switch off the QSPI module by setting the **CLC**.DISR bit. It is responsibility of the user software to issue a disable request in a safe time interval, where no race conditions or pipeline effects between the QSPI module and the DMA can occur. This is no issue if the switch-off procedure is one-way, that is no continuation of operation without reinitialization/reset of the module is expected. **GLOBALCON**.EN bit is cleared.

The QSPI module itself behaves in the following way.

in master mode like entering the sleep mode:

- after ending the TRAIL delay if a transmission is in progress
- or immediately if the module is in WAIT state

in slave mode

- after the bit counter has counted the full predefined number of bits as defined in **BACON**.DL and BYTE bit fields, if a transmission is ongoing or
- immediately if the module is in wait state, no transmission ongoing and shift register empty and TXFIFO empty

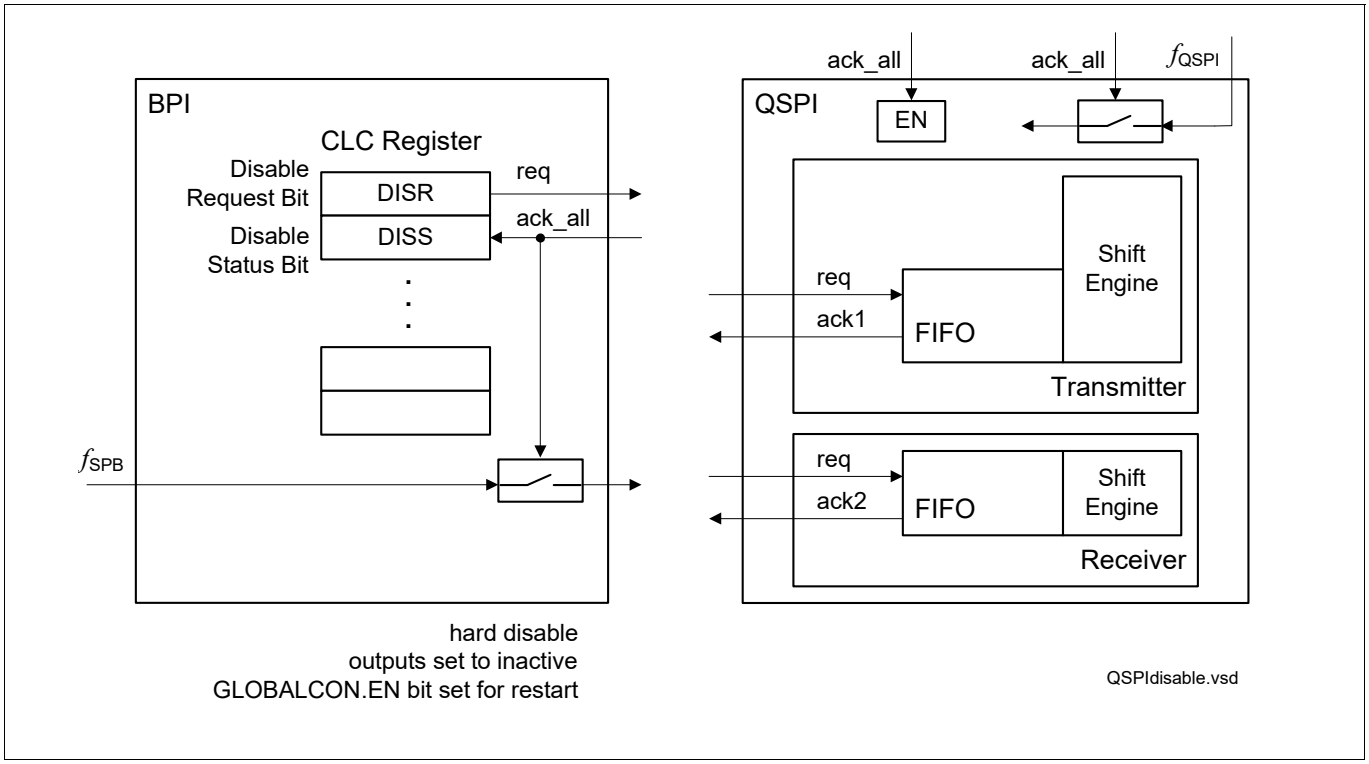


Figure 487 Disable Overview

37.3.5 User Interface

This section describes the possibilities available for transferring data between the on-chip RAM memories and the QSPI module using the transmit and receive FIFOs. The QSPI features one 4 x 32-bit Tx FIFO and one 4 x 32-bit Rx FIFO. The address ranges for writing data and configuration to the Tx FIFO and for reading data and status from the Rx FIFO consist of address locations with special properties:

- Tx FIFO
  - DATA_ENTRY - writes to any of these eight functionally identical locations are always interpreted as data
  - BACON_ENTRY - writes to this location are always interpreted as configuration
  - MIX_ENTRY - writes to this location are interpreted as data or configuration, based on a set of rules
- Rx FIFO
  - RX_EXIT - reads from this location deliver either data and optionally status, based on a set of rules.

Use either MIX_ENTRY or the combination DATA_ENTRY / BACON_ENTRY for sending a frame. Do not combine DATA_ENTRY/BACON_ENTRY and MIX_ENTRY for sending a frame.

The following list gives an overview of the operating modes of QSPI and usage of the TxFIFO entry addresses:

- Short Mode: MIX_ENTRY, DATA_ENTRY / BACON_ENTRY
- Long Mode: MIX_ENTRY, DATA_ENTRY / BACON_ENTRY
- Continuous Mode: MIX_ENTRY, DATA_ENTRY / BACON_ENTRY
- Single Configuration
  Multiple Frames: MIX_ENTRY
- XXL Mode: DATA_ENTRY / BACON_ENTRY
- Move Counter Mode: DATA_ENTRY / BACON_ENTRY

37.3.5.1 Transmit and Receive FIFOs

The Tx_FIFO is filled with both configuration and data entries, which are distributed to the **BACON** and the shift register at the exit of the Tx_FIFO. The TxFIFO keeps track if an entry is configuration or data. The control and the data elements are automatically distributed to the **BACON** and shift register. If no data is available, the new BACON is pending.

If a frame is finished and there is data in the FIFO

- in short mode, a new frame starts automatically with the same configuration as the previous one.
- in long mode, the extra data are ignored.

TXFIFO error conditions:

- FPI bus write to a full TXFIFO generates an overflow interrupt. The write is ignored.
- In case of slave mode, (**GLOBALCON**.MS = 1X), in case of an underflow, only “1” are delivered.

RXFIFO error conditions:

- FPI bus read from an empty RXFIFO generates an underflow interrupt, and delivers only “1” bits.
- Hardware attempt to write a full RXFIFO with data or status generates an overflow interrupt. The write attempt is ignored by the RXFIFO.
- If the **GLOBALCON**.SRF bit is set, the communication is stopped in case of full FIFO in master mode. In slave mode, the bit is ignored, no stop can be done.
- If the FPI bus frequency is very slow, and the baud rate very high, it can happen that some data is lost on the way between the shift register and a not full RXFIFO. In such a case, an overflow interrupt is generated, although the RXFIFO is not full.

The Rx_FIFO can be filled with both status and received data entries, depending on **GLOBALCON**.SI.

Note: If a 8-bit or 16-bit write is executed to the TXFIFO entry addresses, the data is mapped 1:1 to its location within the 32-bit wide TXFIFO entry, and the rest is padded with zeros.

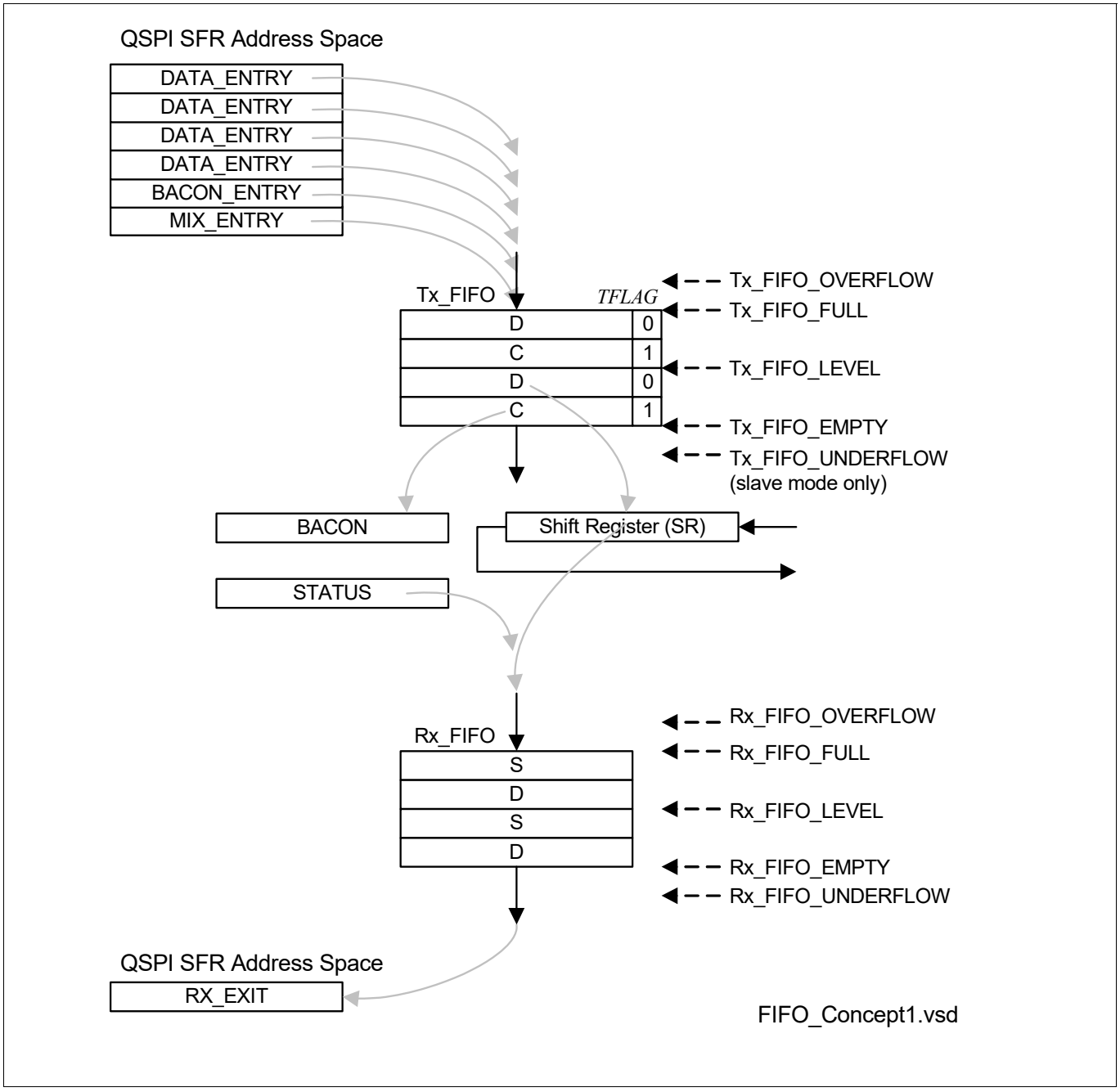


Figure 488 Architecture of the Tx and Rx FIFOs

(>>**Interrupts**)

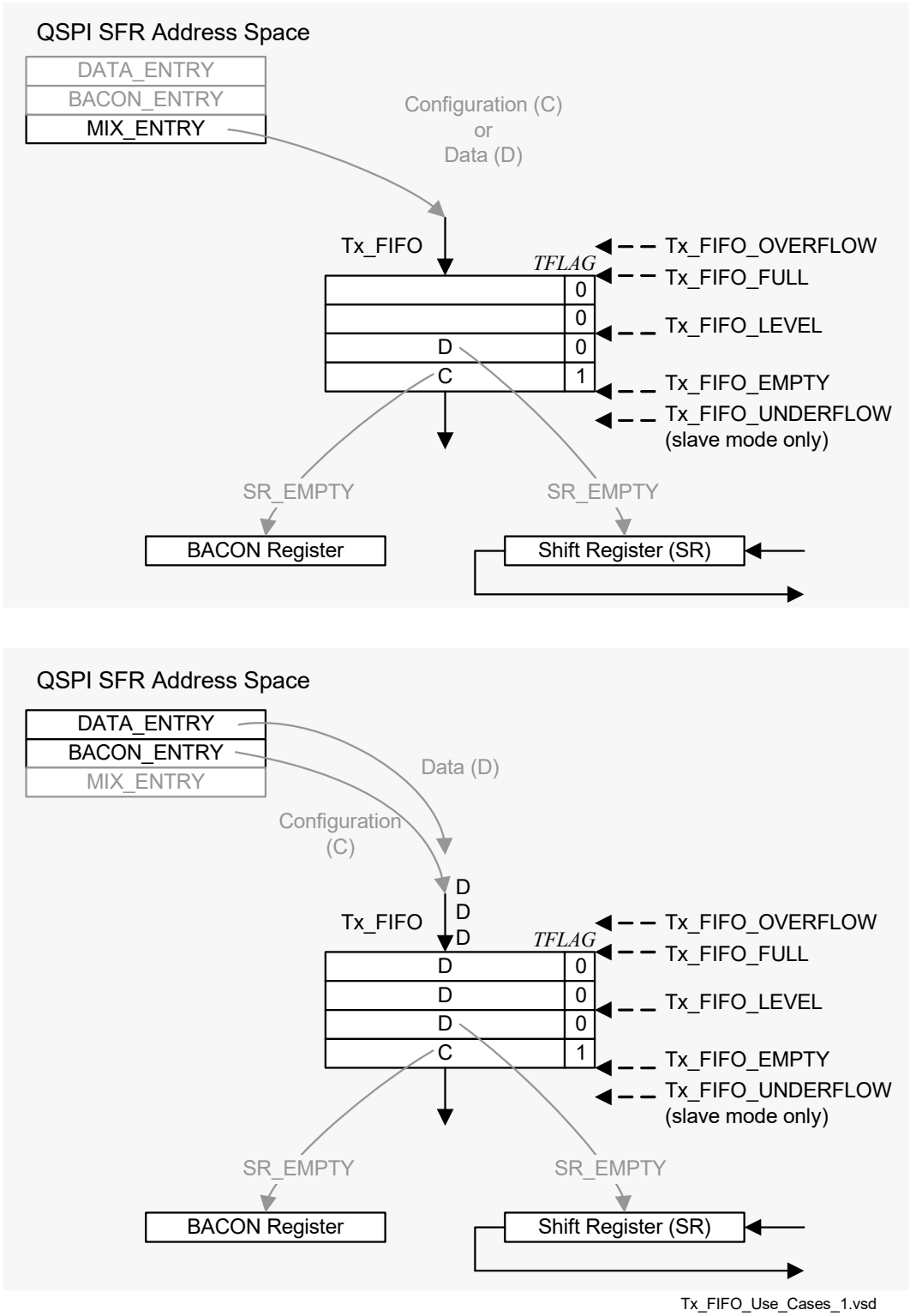


Figure 489 Using Tx FIFO Mix and Bacon / Data Entries

37.3.5.1.1 Short Data Mode

In Short Data Mode, the QSPI module transmits single data with a length of 2 to 32 bits in one frame. This mode is defined by **BACON**.LAST = 1 and **BACON**.BYTE = 0.

The transfer cycle is a sequence of the following phases: W?_I_L_D_T. The symbol “?” means that the phase is optional (duration of 0 allowed).

For example, a Tx_FIFO event can be used to trigger one DMA transfer (consisting of two 32-bit moves) to transfer both the BACON and DATA entries from an on-chip RAM memory to the QSPI module.

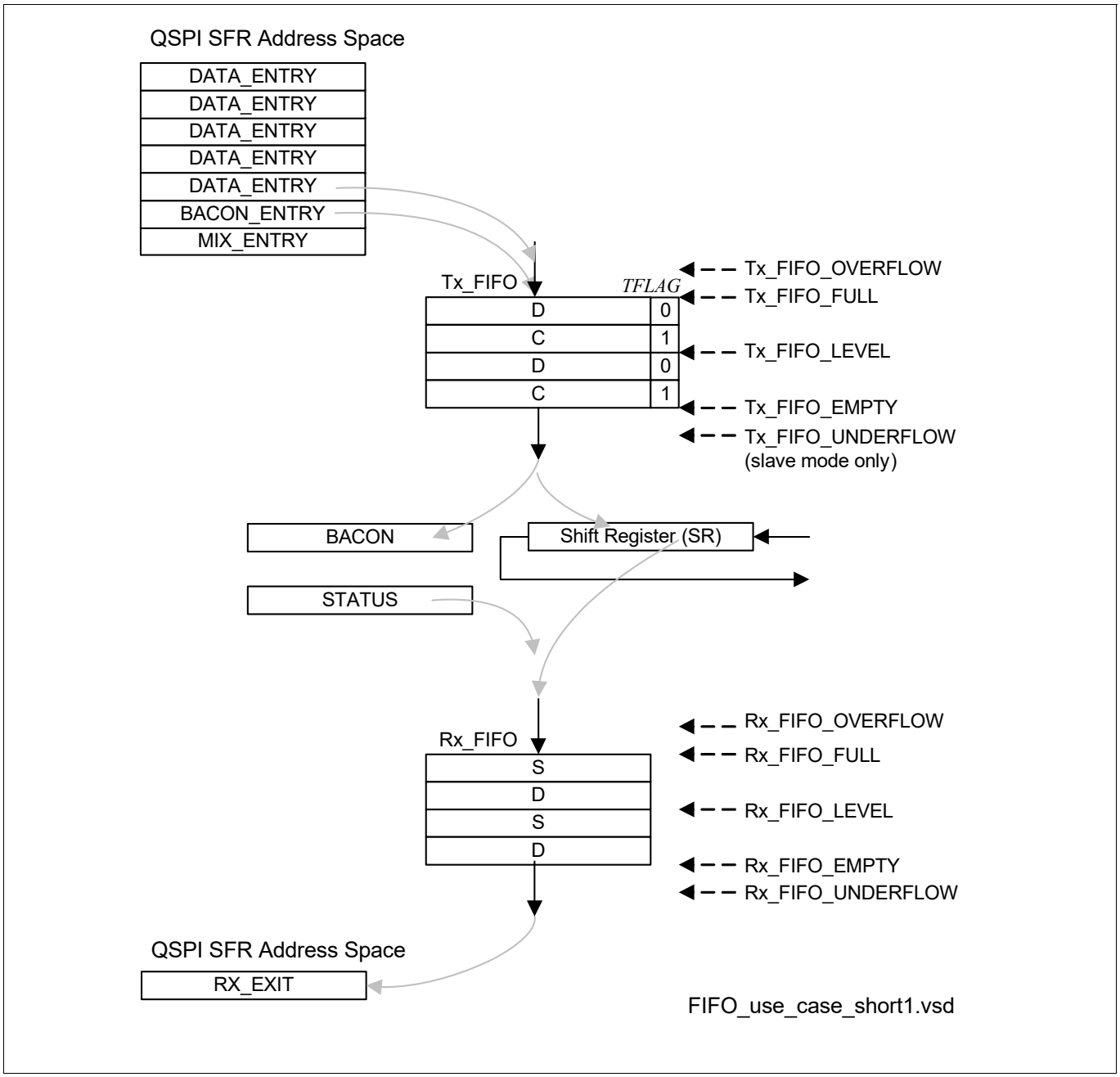


Figure 490 User Interface in Short Data Mode

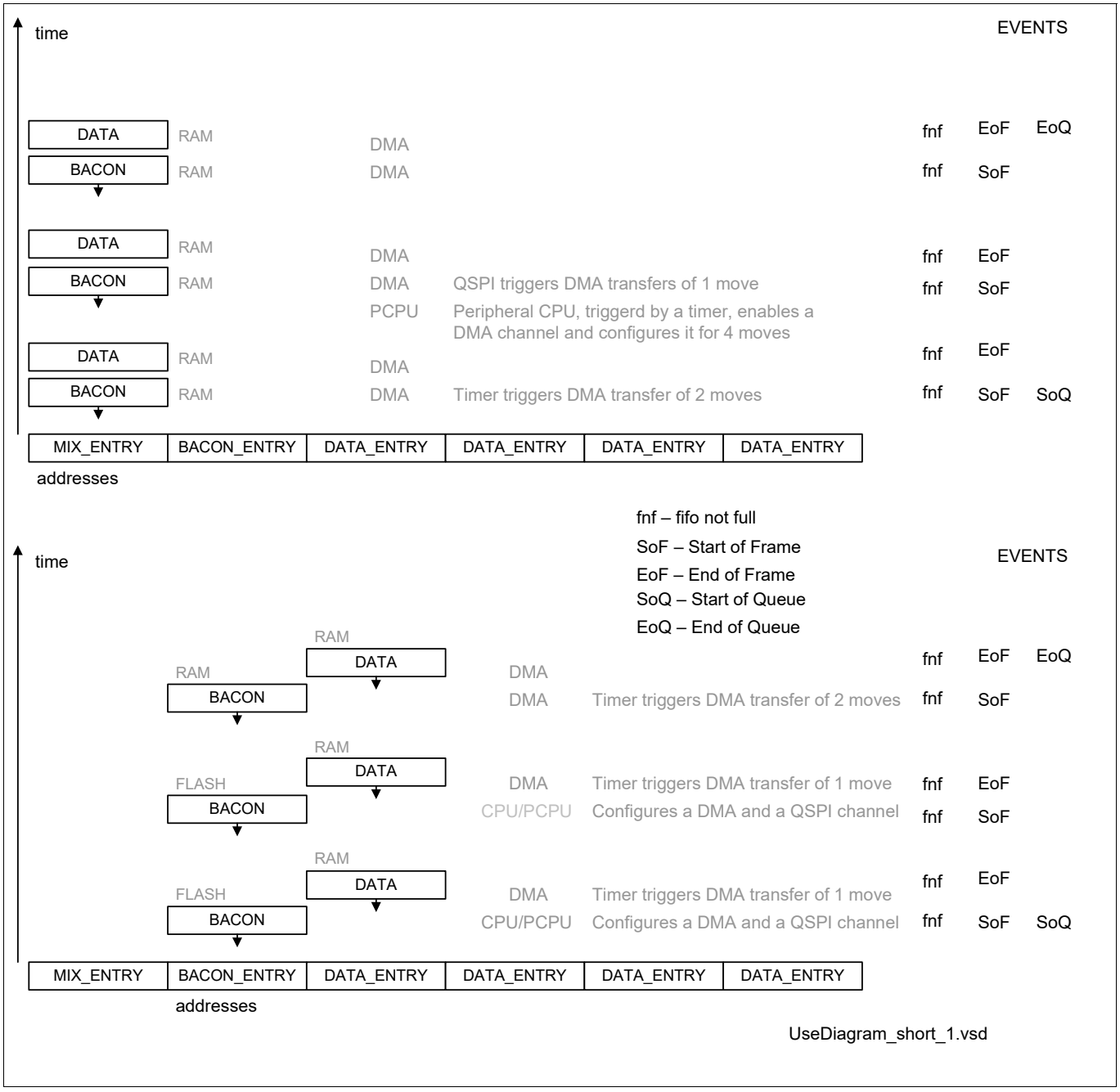


Figure 491 Use Diagram of the User Interface in Short Data Mode

37.3.5.1.2 Long Data Mode

In Long Data Mode, the QSPI module transmits bursts of up to 32 bytes (256 bits) in one frame. This mode is defined by programming **BACON**.LAST = 1 and **BACON**.BYTE = 1.

One transfer cycle in long data mode is a sequence of the following phases: W?_I_L_D_T. The W_I_L_D_T_I_L_D_T sequence is possible. The symbol "?" indicates that the phase is optional (duration of 0 allowed). Each "D" indicates a data length as defined with **BACON**.DL and **BACON**.BYTE, that is, up to 32 bytes. One "D" is made of a number of 32-bit TXFIFO entries "d" and one rest at the end "b", which can be written as D=ddddddb.

The Long Data Mode can be extended by using the **XXLCON** register.

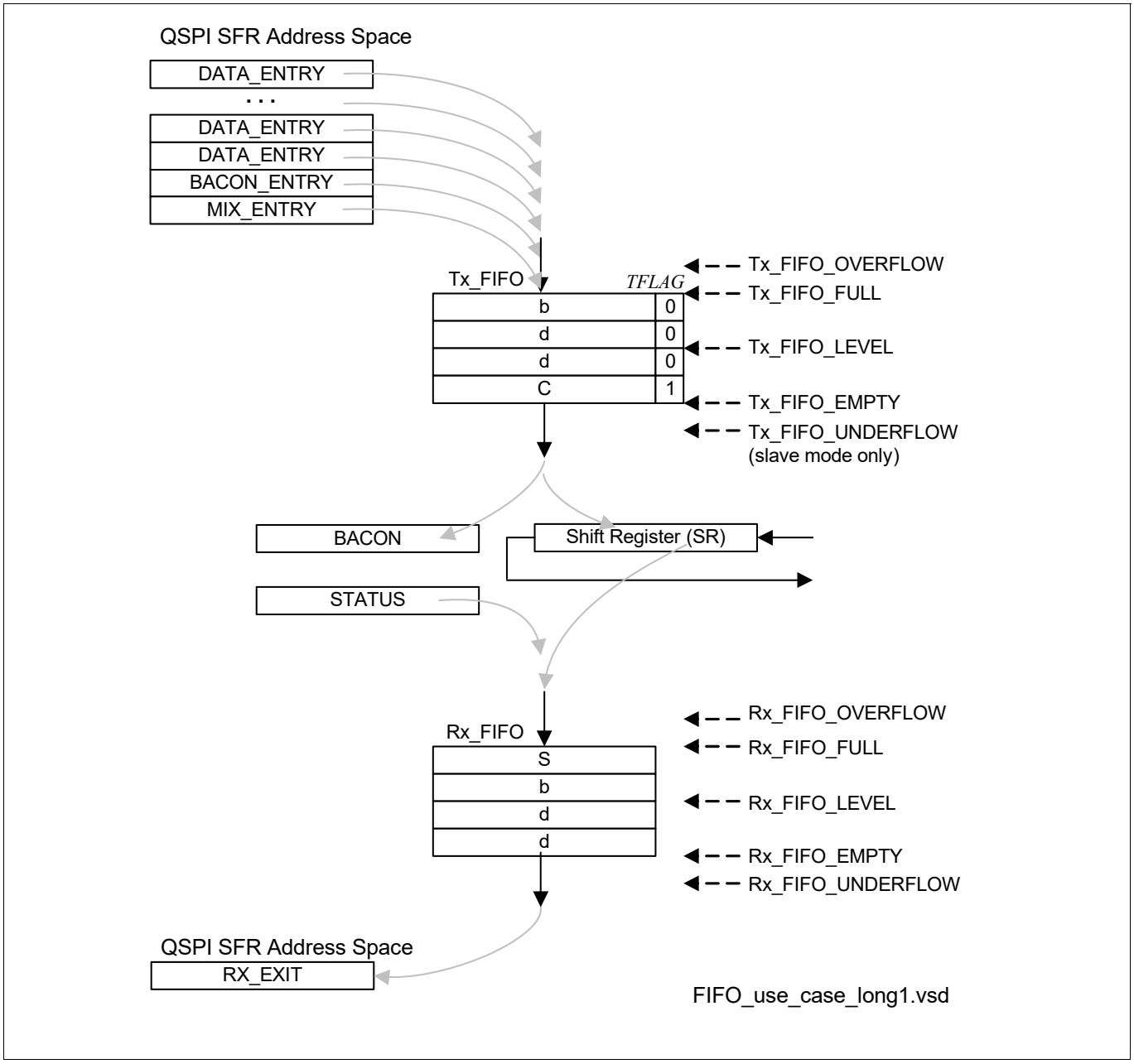


Figure 492 User Interface in Long Data Mode

In a DMA transfer example, a Tx_FIFO_EMPTY event can be used to trigger one DMA transfer (consisting of, for example, four 32-bit moves) to transfer the BACON and three DATA entries from RAM memory to the QSPI module.

The three data words could contain 80 bits (10 bytes) of data. The three data words could also contain 40 bits (5 bytes), 64 bits (8 bytes) plus one dummy word. In this way short bursts of up to 96 bits can be transmitted.

In such 4 words DMA transfer example, the Tx_FIFO event can be used to transfer two 32-bit words per transfer.

If a configuration word is written when a data word is expected, the configuration word is ignored, and an Unexpected Configuration Error is raised.

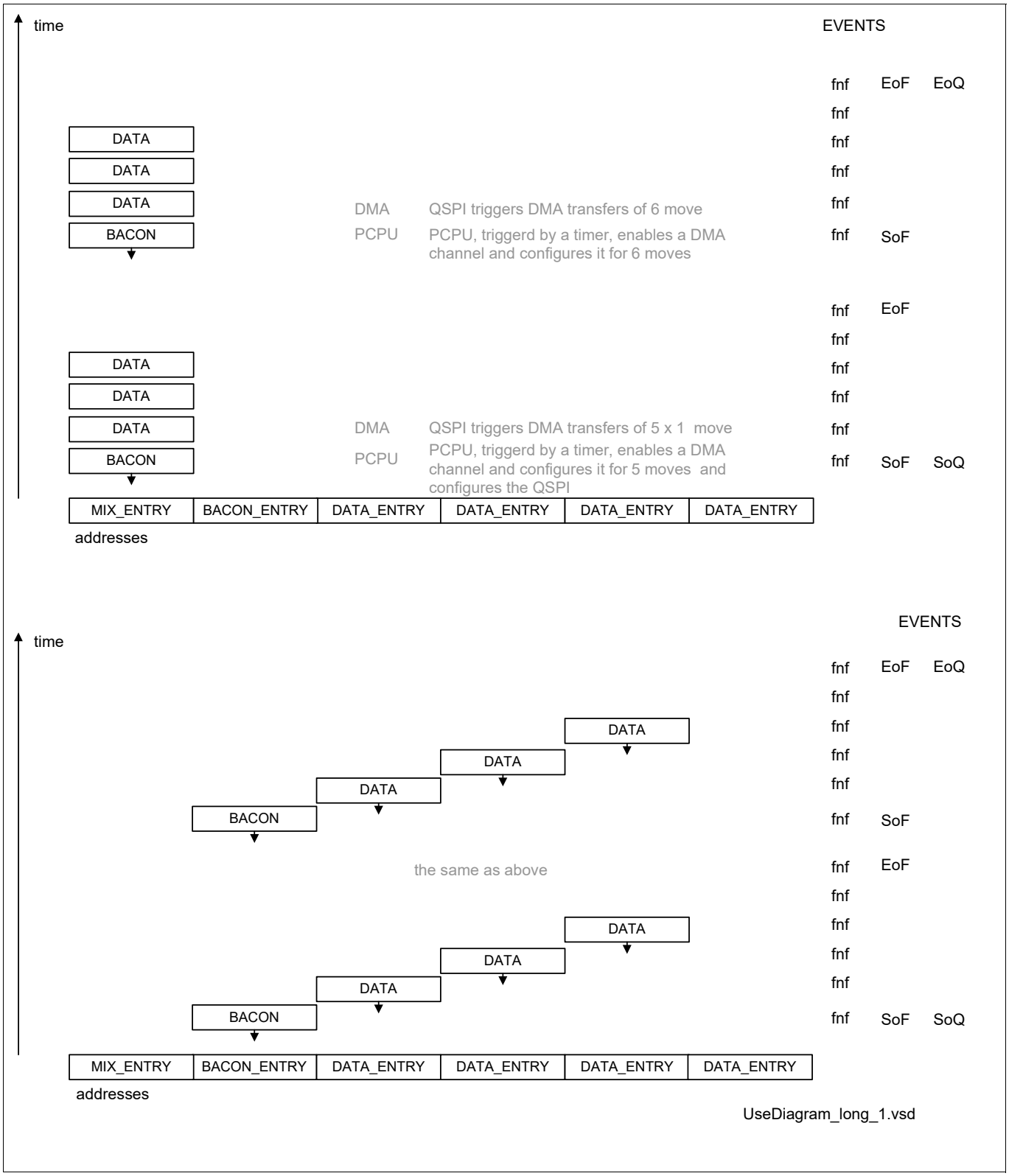


Figure 493 Use Diagram of the User Interface in Long Data Mode

37.3.5.1.3 Continuous Data Mode

In Continuous Data Mode, the QSPI transmits a stream of data with an arbitrary length and an active slave select signal. It can be used with both short and long data.

This mode starts by programming a control word containing **BACON**.LAST = 0 and the first data. The communication continues with the subsequent data entries written. The mode ends with a control word containing **BACON**.LAST = 1 and the last data. If LAST = 0, then TRAIL is a delay between data blocks. The W_I_L_D_T_D_T_D_T sequence appears. Each “D” represents data block as defined with **BACON**.DL and **BACON**.BYTE (up to 256 bits). If LAST = 1 then TRAIL is trailing delay, see **Figure 464**.

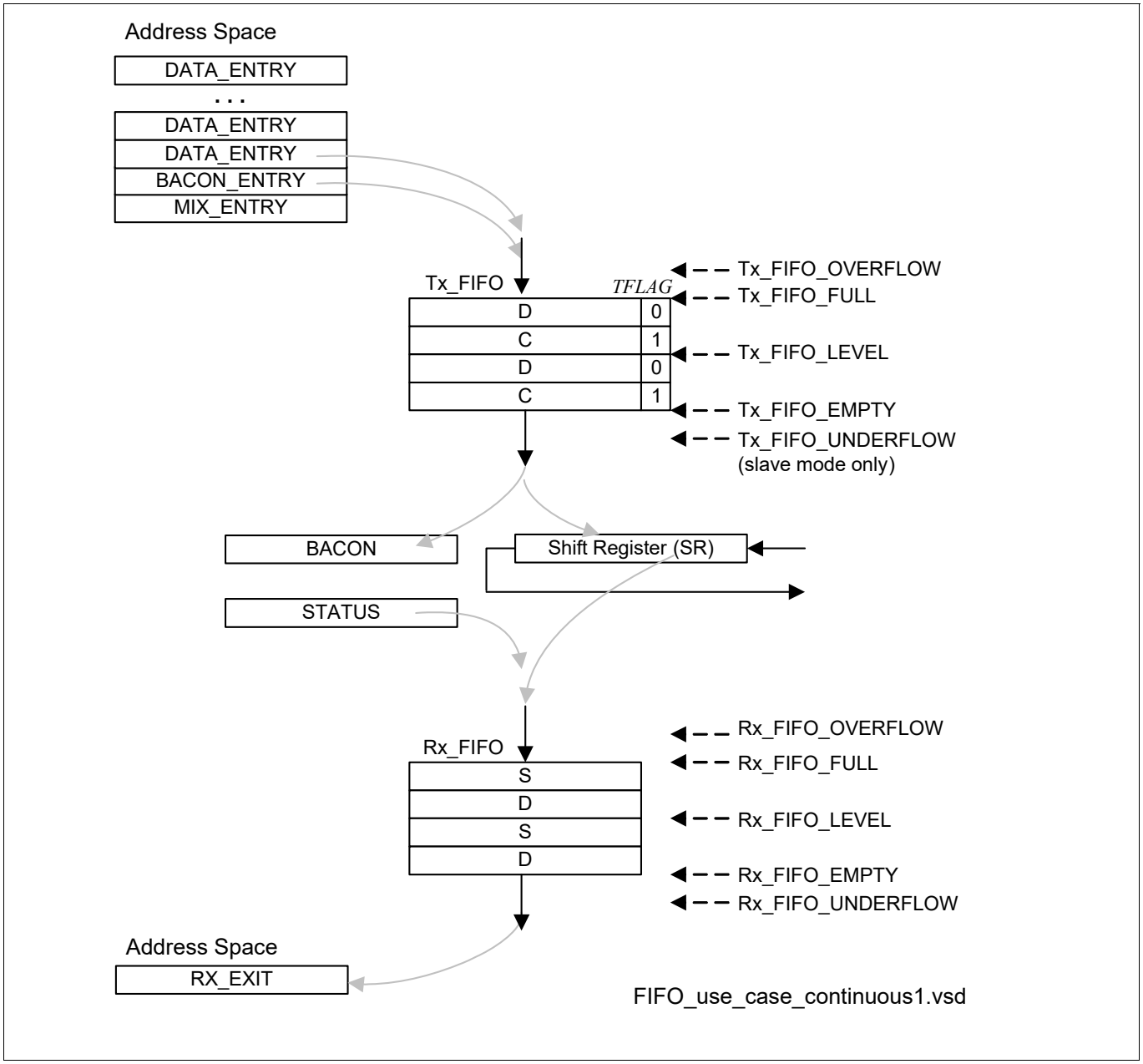


Figure 494 User Interface in Continuous Data Mode

The **Figure 495** shows an example of continuous data mode using short data. Here, the CPU starts the stream by writing BACON to the BACON_ENTRY address, and afterwards a DMA transfers the whole stream except the last BACON and DATA.

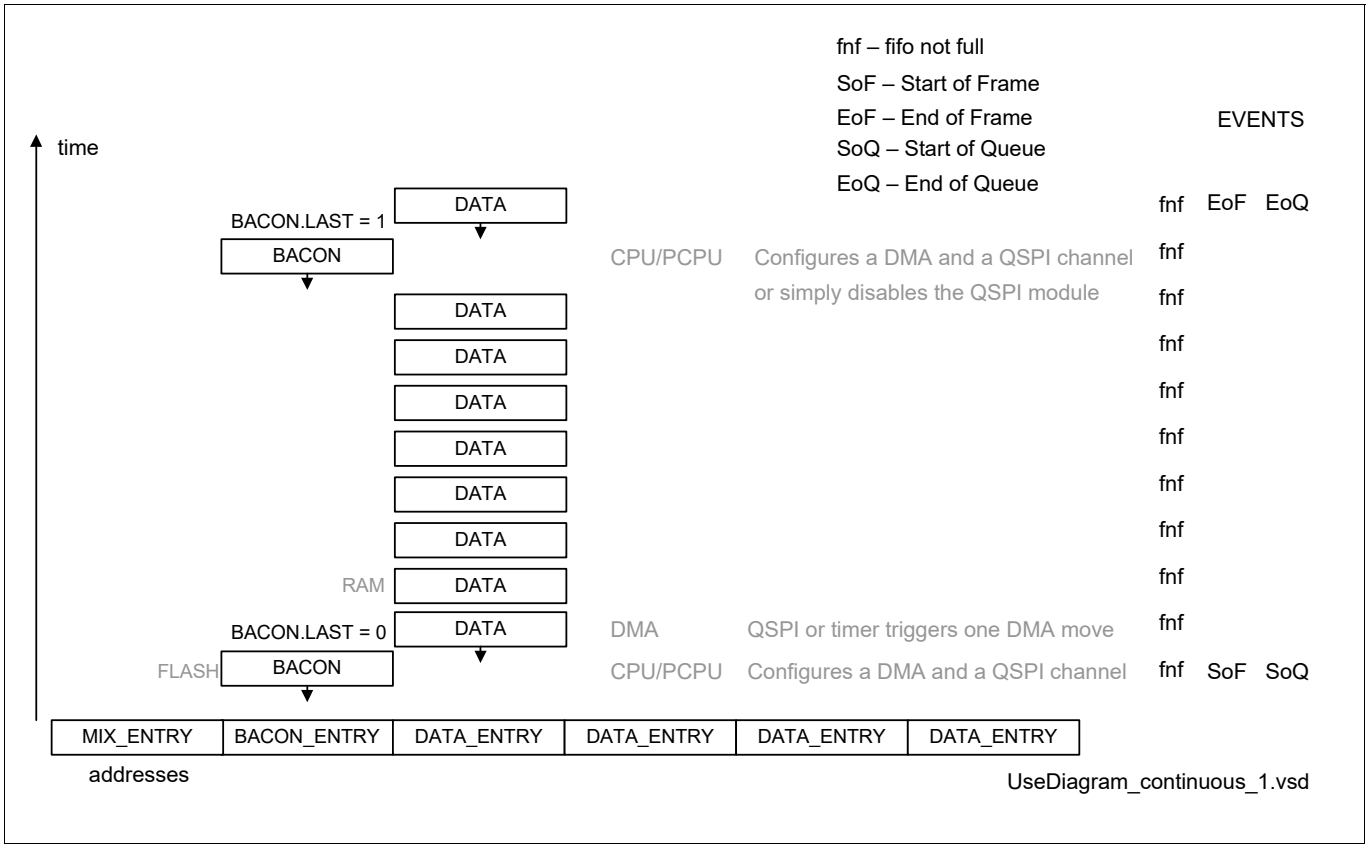


Figure 495 Use Diagram of Continuous Data Mode with Short Data

The **Figure 496** shows an example of continuous data mode using long data. Here, the first BACON must be repeated after each data block, but on the other hand, the whole transfer can be done using only the MIX entry and one single DMA channel, without CPU involvement. The loss of efficiency because of the unnecessary extra BACONs can be minimized by using long data blocks.

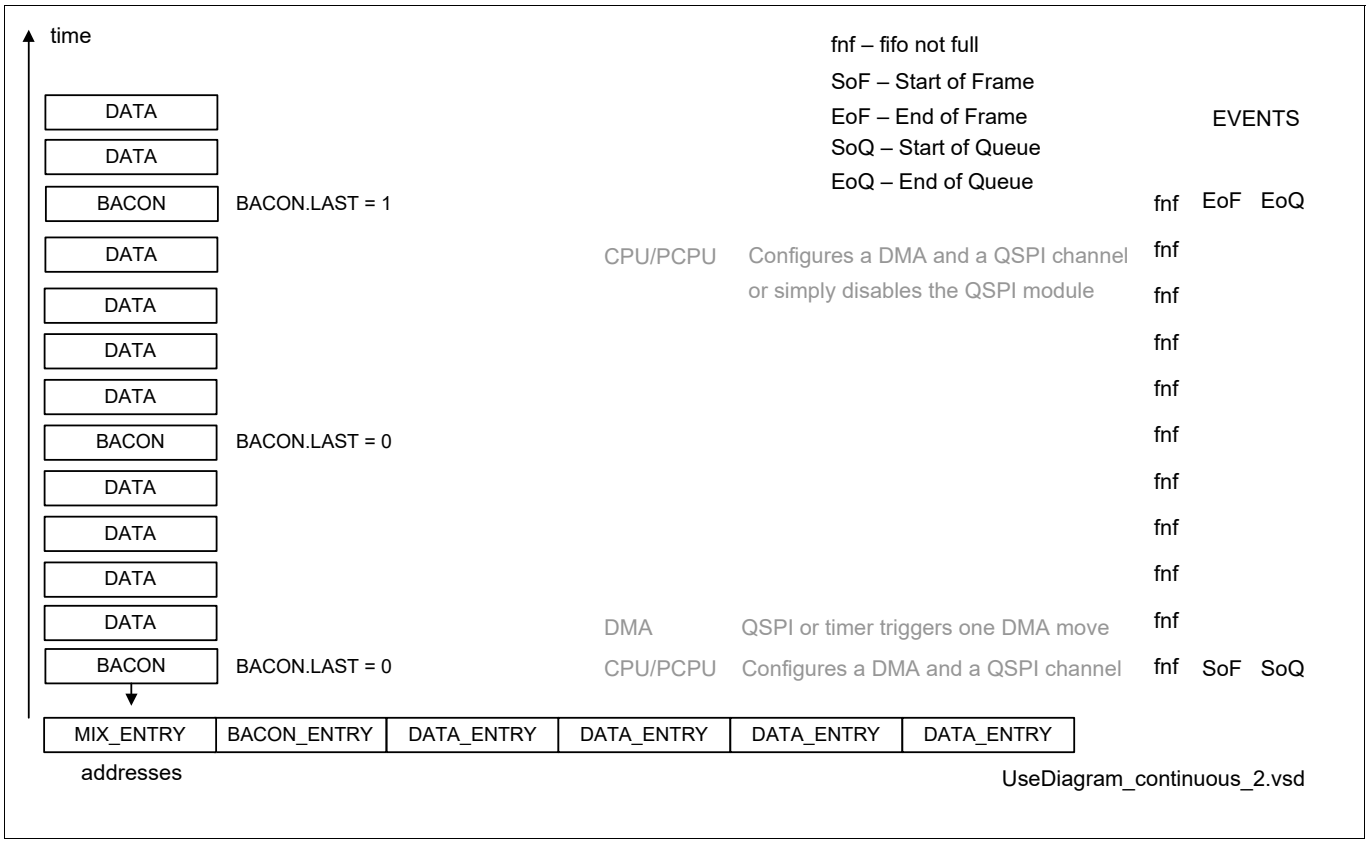


Figure 496 Use Diagram of Continuous Data Mode with Long Data

37.3.5.1.4 Single Configuration - Multiple Frames Behavior

If longer sequences of data with length of up to 16 bit have to be sent to single slave in separate frames, then the DMA move for the repeating configuration word can be saved. An example of a TxFIFO contents and the resulting message sequence is shown in **Figure 497**.

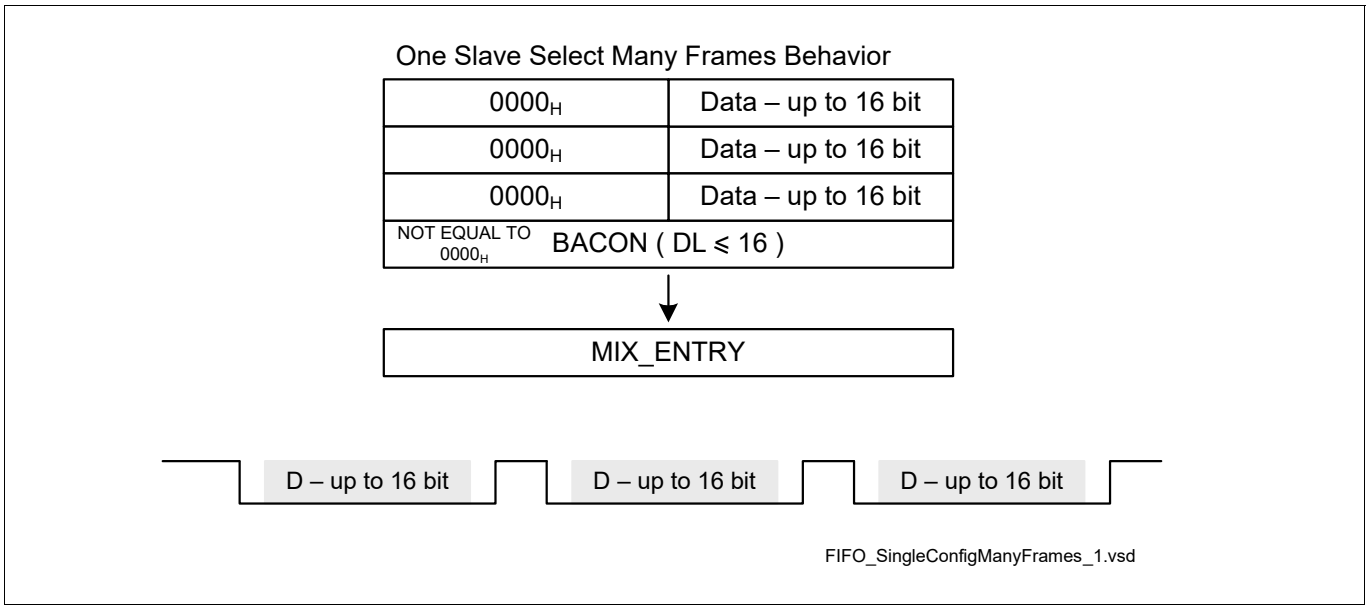


Figure 497 Multiple Frames to One Channel (Example of up to 16 Bit Data)

In order to support sending short multiple frames to single channel without having to send the same configuration each time, the TXFIFO follows this rule:

- If **BACON**.DL less or equal 15 (up to 16 bit data length)
- if MIX_ENTRY address is used
- then if the upper 16 bit of a 32-bit MIX_ENTRY are 0, the entry is considered data, else configuration. Note that there is no valid configuration word with upper 16 bits equal to 0.
- In such a case the TXFIFO marks the entry as data.

*Note: In case the **BACON**.LAST = 0, the slave select remains active between the frames. The behavior is equal to the continuous mode. The disadvantage of this method is that only up to 16 bits data can be transferred with one FPI bus move. The advantage is that continuous mode is possible using the mix entry.*

37.3.5.1.5 Big Endian Data Format

The bit-field **ECONz (z=0-7)**.BE activates the permutation of the data bytes written to the shift register and read from it to the big endian format and back.

ECONz (z=0-7).BE allows for switching between big and little endian per slave (SLSO signal).

See **Figure 498** and **Figure 499**.

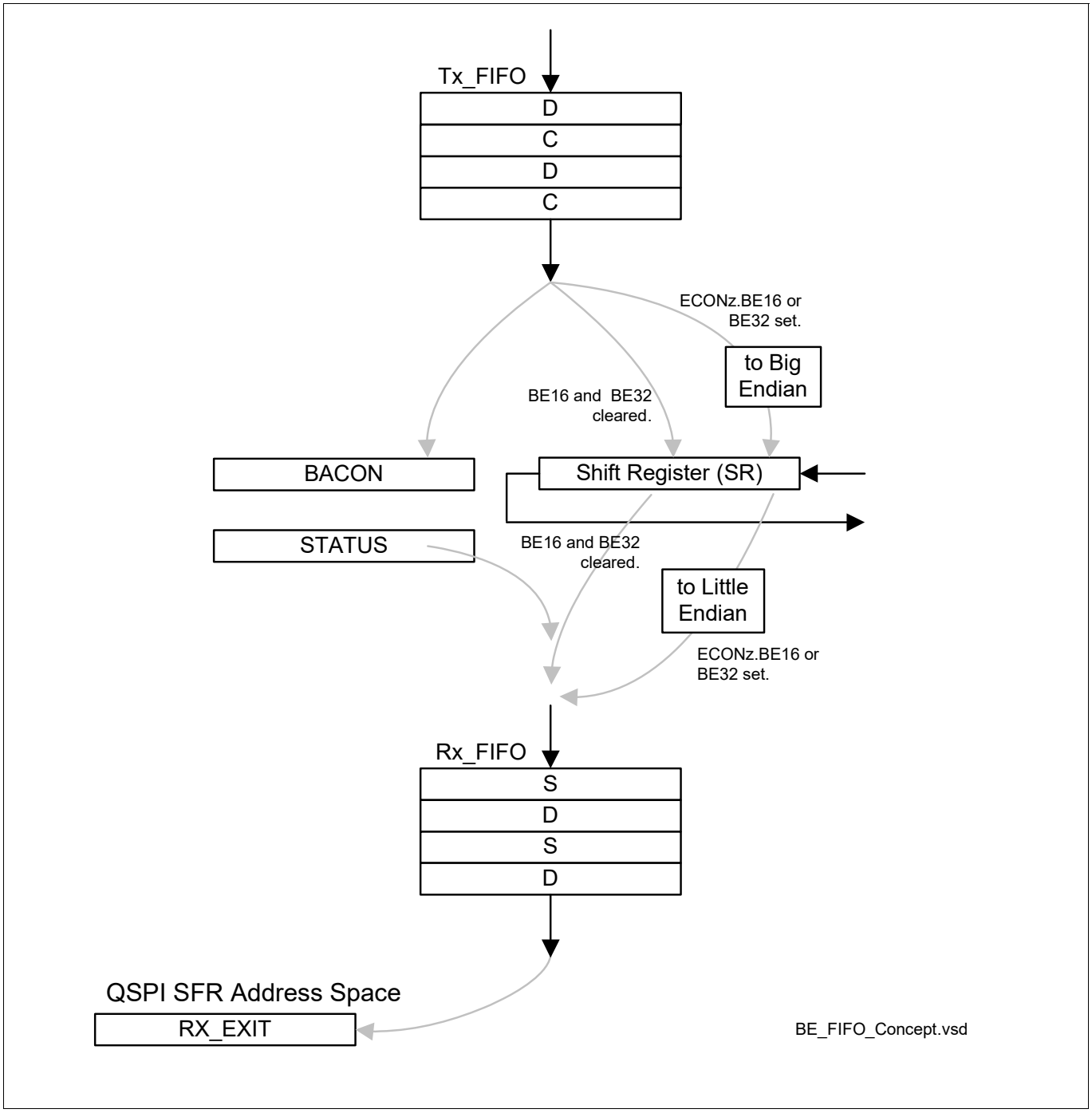


Figure 498 Big Endian Data Path

Converting data from little to big endian and other way around requires one and the same permutation of the data bytes, as shown in **Figure 498**, one permutation block is located before and one after the shift register, so that only data can be permutated, and not **BACON** or **STATUS**.

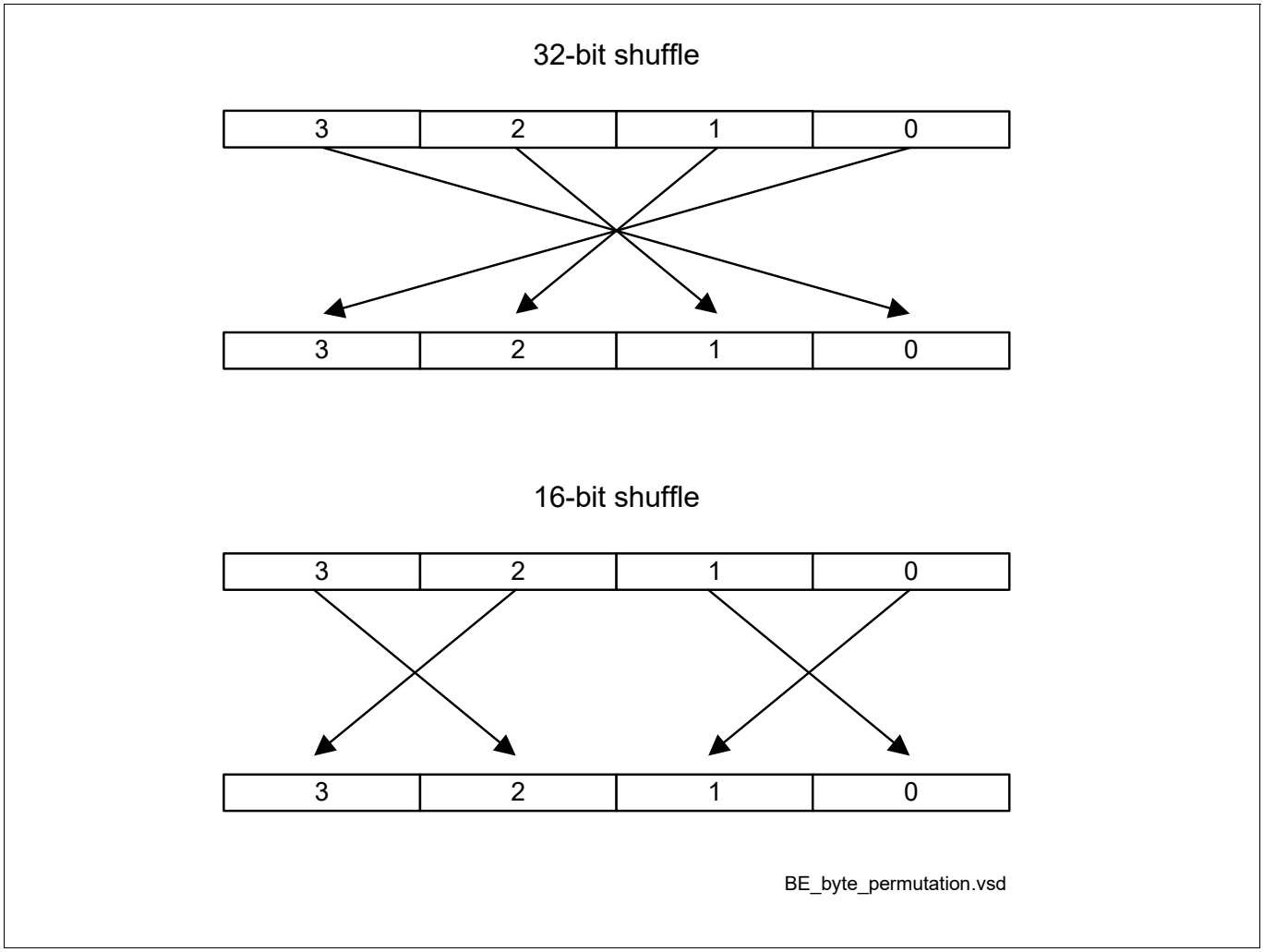


Figure 499 Endianness Byte Permutation

The endianness byte permutation should be used only for 16-bit and 32-bit data (and multiples of it), and not for other data lengths.

37.3.5.2 Loop-Back Mode

Loop-Back mode is a master mode test feature used for establishing a module internal connection between the transmit and the receive signal, the same as if the transmit and receive pins would be short connected. It is activated by setting the **GLOBALCON**.LB bit.

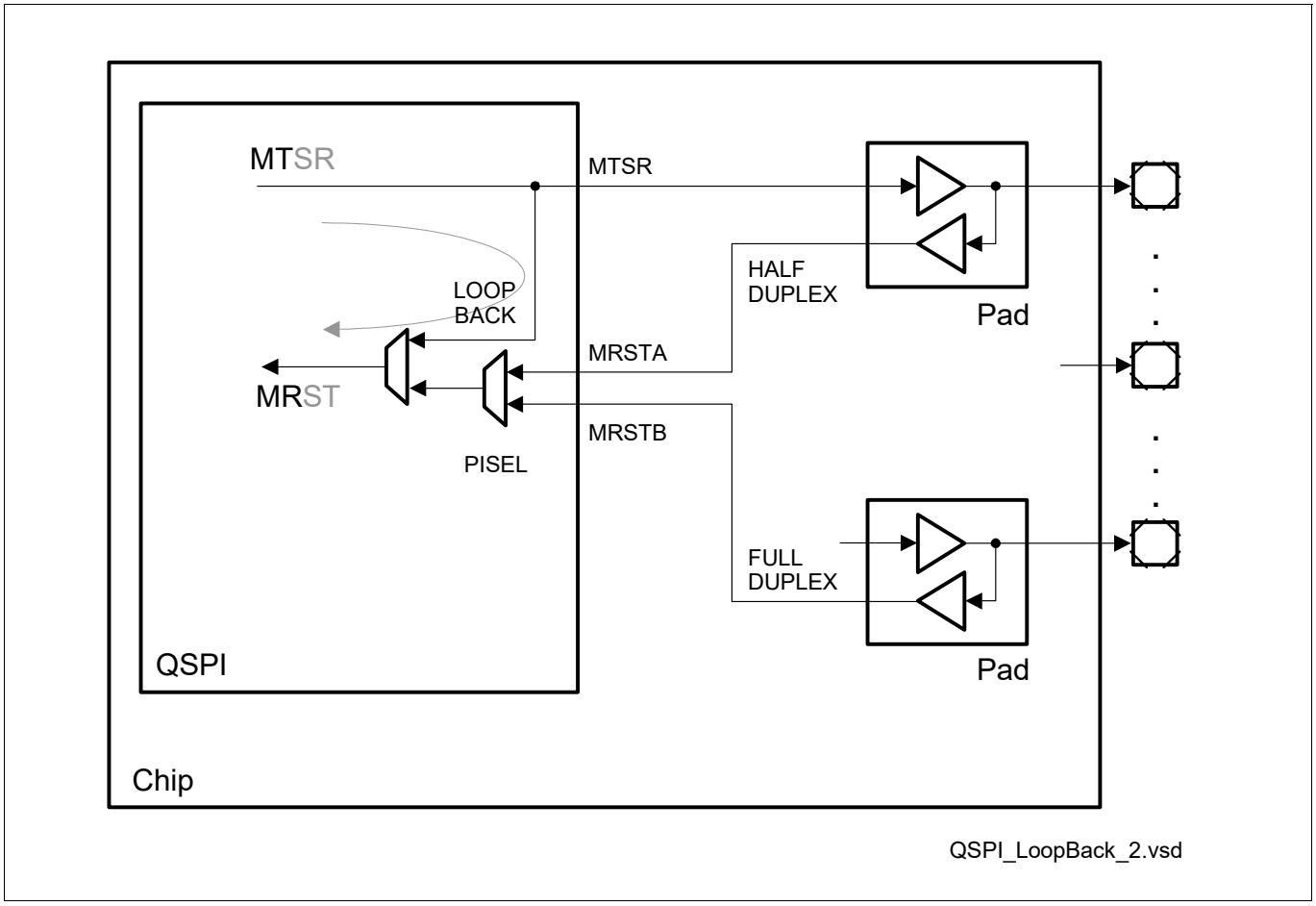


Figure 500 Loop-Back Mode

37.3.6 Interrupts

There are several types of interrupts:

- Interrupts related to the user interface
  - FIFO related interrupts
- Interrupts related to the shift engine (state machine)
  - error interrupts
  - phase transition interrupts

According to this classification, here follows the complete list of all the interrupts:

- FIFO related interrupts
  - TX - Transmit FIFO Interrupt, requests feeding of the TXFIFO, but does not signal the error events of overflow and underflow (see **Figure 488**)
  - RX - Receive FIFO Interrupt, requests emptying of the RxFIFO, but does not signal the error events of overflow and underflow (see **Figure 488**)
- ERR - Error Interrupt
  - signals every error condition
  - signals TX_FIFO_OVERFLOW, TX_FIFO_UNDERFLOW
  - signals RX_FIFO_OVERFLOW, RX_FIFO_UNDERFLOW
- PT - Phase transition interrupt, signalling two out of all phase transitions in the QSPI communication cycle (see **Figure 473** and **Figure 476**)
  - PT1 event
  - PT2 event
- U - User Interrupt
  - triggered by the PT1 event during the time **BACON**.UINT bit keeps this event enabled.
- MC - Move Counter Mode Interrupts
  - Interrupt before last - triggered when move counter was decremented to 1.
  - Interrupt after last - triggered when move counter was decremented to 0.

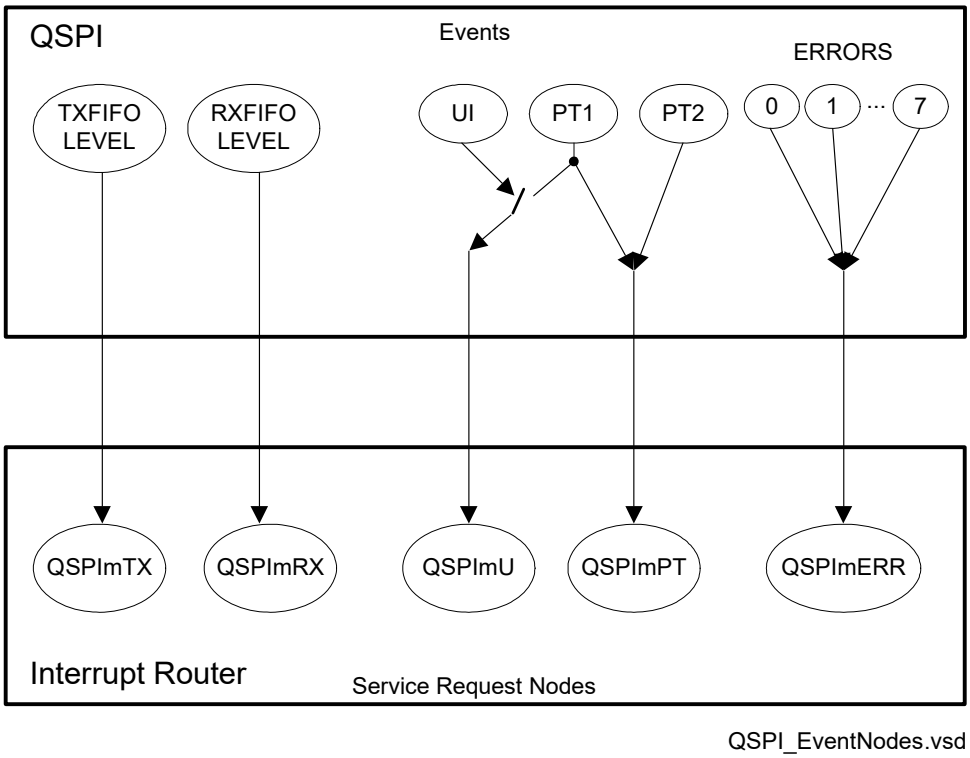


Figure 501 Events and Interrupts Overview

37.3.6.1 Slave Mode SLSI Interrupt

In slave mode, deactivating the SLSI signal (SLSI rising edge) triggers the PT interrupt, if PT2 enabled.

37.3.6.2 Interrupt Flags Behavior

A state of an event flag (low or high) does not influence the flow of its event signal. If an event occurs recurrently, and the event flag remains set (if not cleared by the software interrupt handler), the interrupt flag (which is cleared by hardware, by an ICU, when the interrupt wins an arbitration round) would receive recurrent triggers, and the flag would be constantly set all the time.

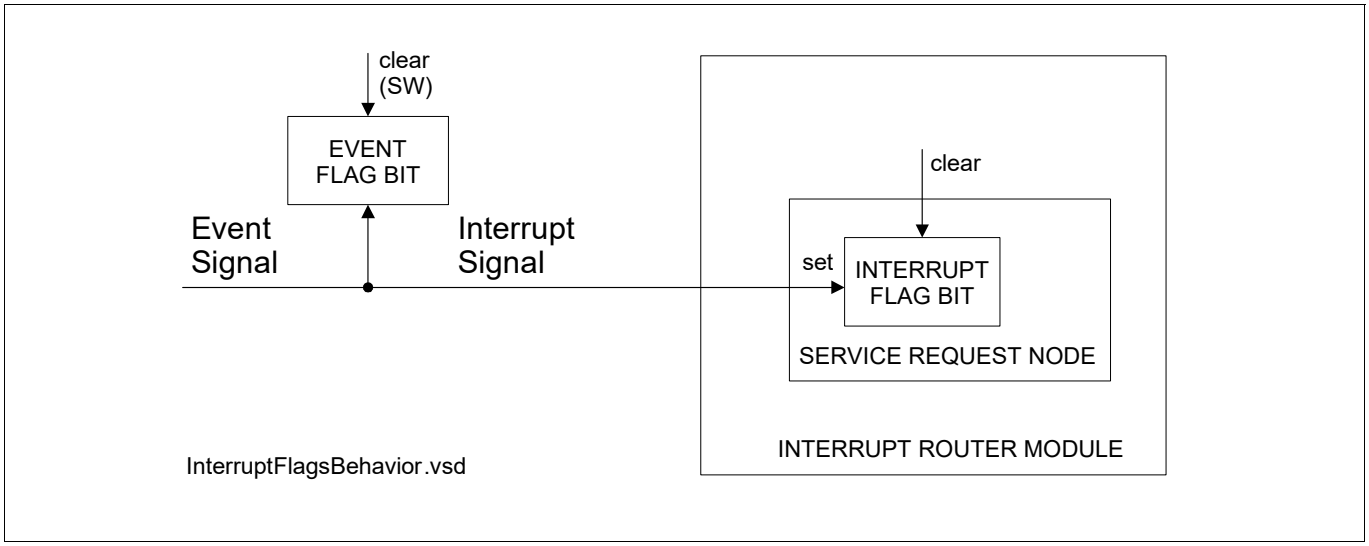


Figure 502 Events and Interrupts Overview

37.3.6.3 TXFIFO Interrupt Generation

The TXFIFO provides two interrupt generation modes, selected by the bit-field **GLOBALCON1**.TXFM:

- Single Move Mode
- Batch Move Mode
- Combined Mode

The RXFIFO and the TXFIFO can use the interrupt generation modes independently of each other.

Single Move Mode

The purpose of the Single Move Mode is to keep the TXFIFO as full as possible, refilling the TXFIFO by writing to it as soon as there is a free element.

An example use case could be using single move mode with a DMA performing single move per TXFIFO interrupt. In this mode the DMA keeps the TXFIFO full. A DMA request is triggered each time a write to the TXFIFO is performed, and there is at least one empty element available. If the write makes the TXFIFO full (no more empty elements), an interrupt is not generated in order to prevent overflow. It is generated later when the shift register (or BACON) reads one element from the TXFIFO, making it not full.

The **Figure 503**, shows the filling levels of the TXFIFO, and the events associated with each filling level triggering a TXFIFO refill interrupt.

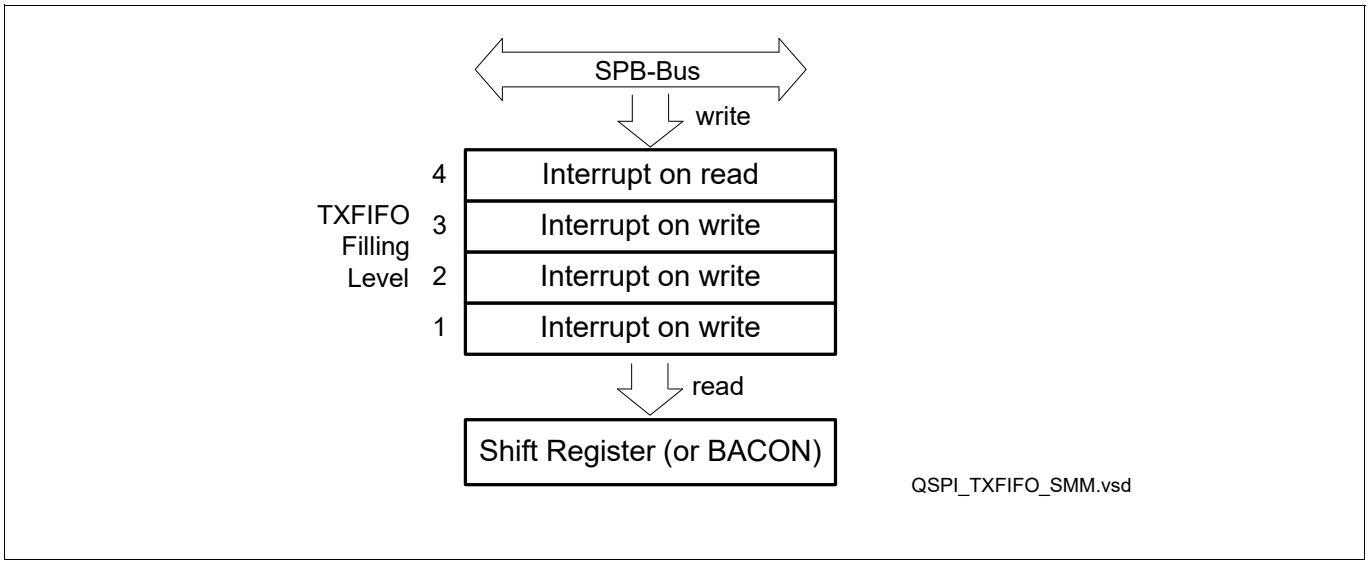


Figure 503 Interrupt Generation in the Single Move Mode

In order to initiate the very first chain of the refill interrupts after power-on reset, it is necessary that the software either performs one write to the TXFIFO, or sets the interrupt flag in the TXFIFO interrupt node. Afterwards, the (DMA) interrupt-chain is self sustaining until the whole transaction is over.

Attention: ***In Single Move Mode multiple software writes or block DMA moves would lead to multiple interrupts and (false) transaction lost events. Therefore they should be avoided - only single moves should be used.***

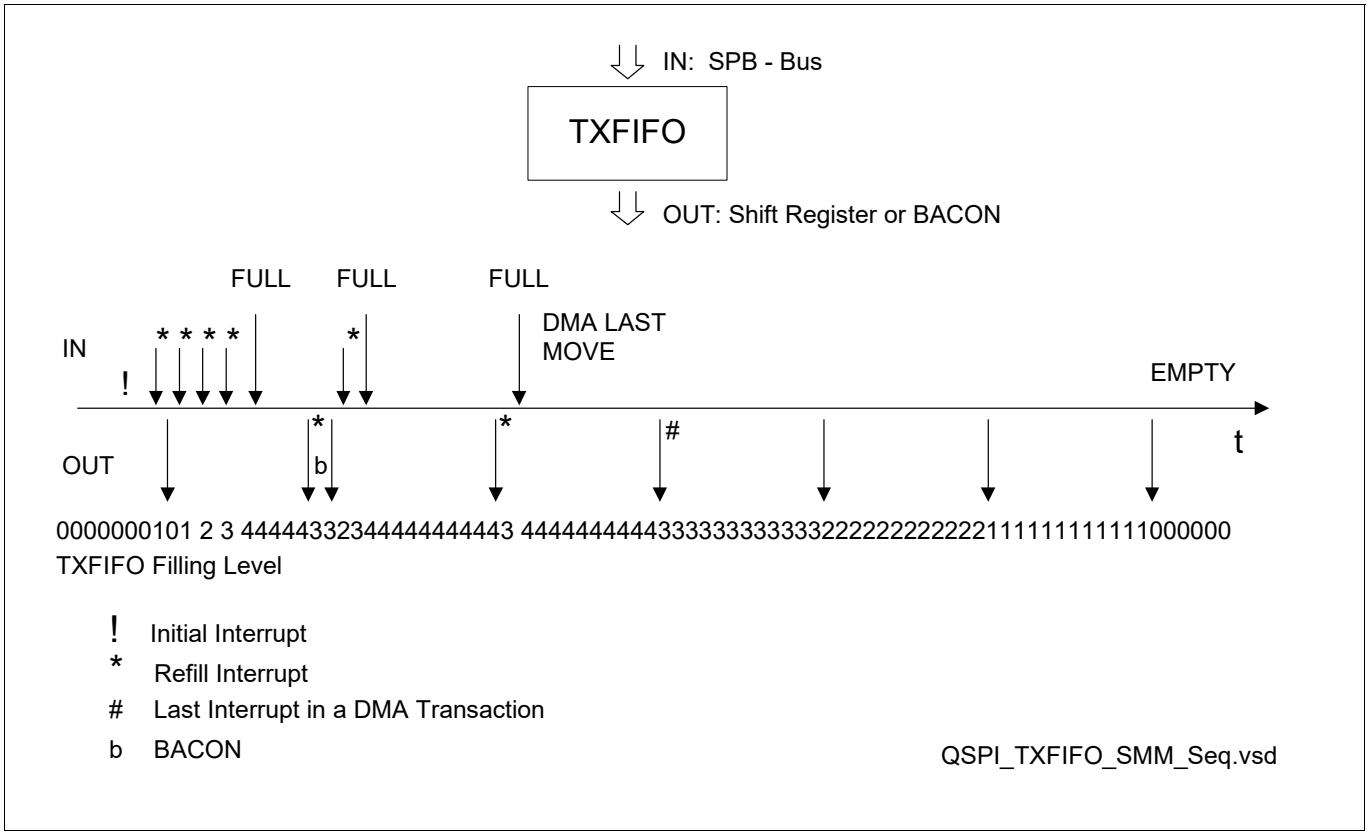


Figure 504 TXFIFO - Interrupt Operation in Single Move Mode

Batch Move Mode

The purpose of the Batch Move Mode is to reduce the number of interrupts by triggering an interrupt when more than one TXFIFO element is empty. For example, a CPU can be interrupted less frequently and it can perform more than one moves per interrupt.

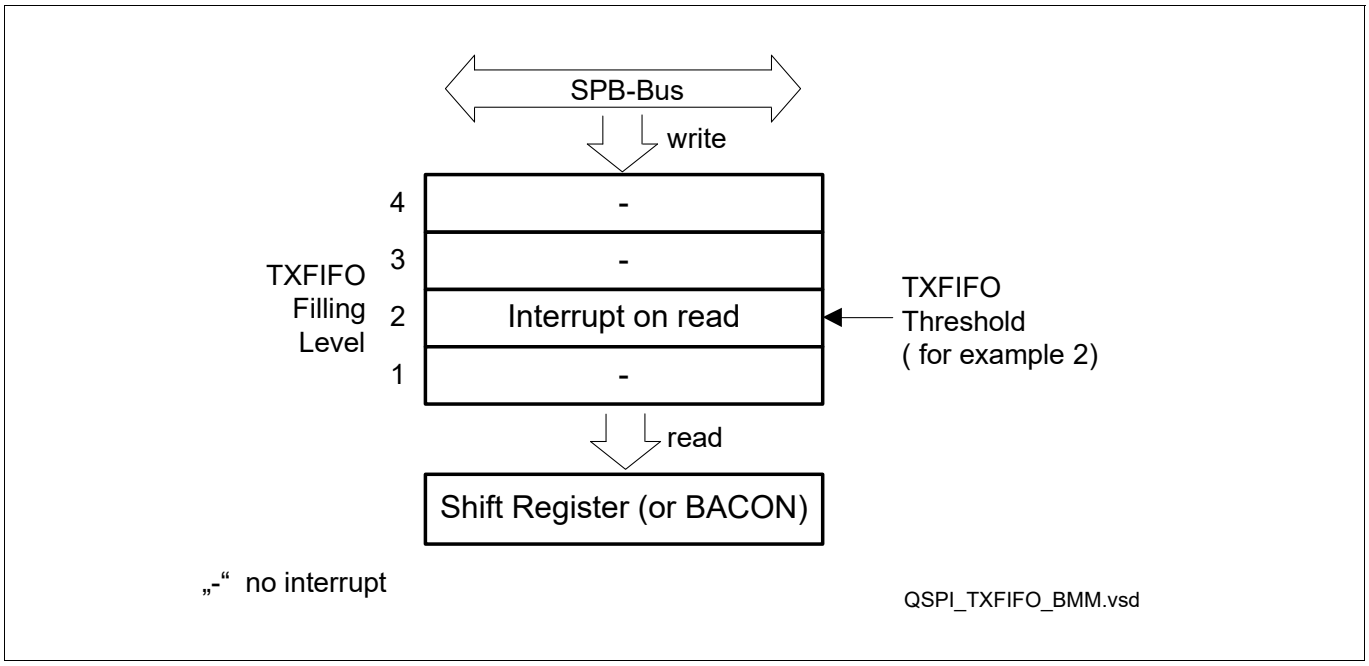


Figure 505 TXFIFO - Interrupt Generation in the Batch Move Mode

In Batch Move Mode, an interrupt is generated only at one point in the TXFIFO, when the filling level falls below the programmed threshold, implying that there are at least the predefined number of empty TXFIFO elements available.

Attention: The TXFIFO must be refilled until the filling level has risen above the interrupt threshold (ensured by polling the filling level), in order to guarantee that the next interrupt will occur.

At high baud rates and short frames, it could be possible that a couple of frames have been transmitted until the moment the TXFIFO is refilled. For example, if the threshold is set to three full elements, then the interrupt will be set when the filling level falls below three, making two elements free. If the refill moment of the TXFIFO is delayed so long that two frames have been transmitted (or frame and BACON), the TXFIFO will become empty, and refill sequence of two will not reach the level of three. So, the next interrupt would never come unless the filling level is being polled and the CPU keeps refilling in a loop until the threshold has been passed (ideally the TXFIFO would be filled completely with each batch of moves).

Combined Mode

TXFIFO generates interrupt each time a data is fetched from it below the pre-programmed threshold, as defined in the **GLOBALCON1**.TXFIFOINT. Consequently, it generates two close interrupts when delivering BACON and the first following data. If the BACON TX interrupt is not serviced until the DATA1 TX interrupt comes, which will always be the case due to their time proximity, the BACON TX interrupt will be lost. Therefore, it is recommended that a DMA should always perform two moves per interrupt trigger (transfer size of two). Otherwise, the TXFIFO average filling level will go down with the time and at the end remain oscillating between zero and one (except in continuous mode). The DMA transaction loss event that may be raised by the DMA should be ignored.

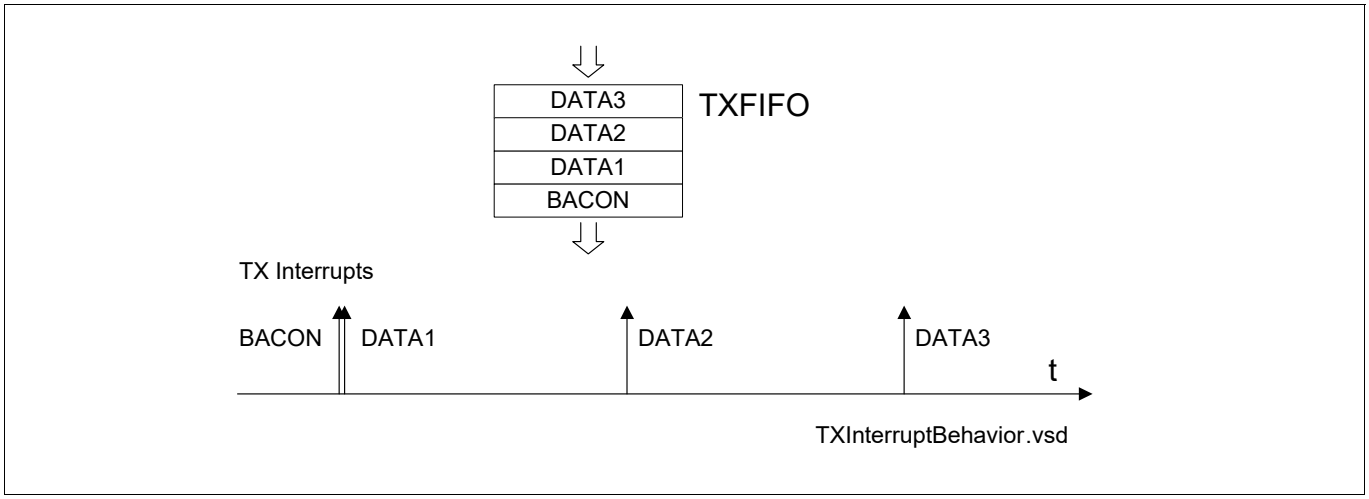


Figure 506 Events and Interrupts Overview

37.3.6.4 RXFIFO Interrupt Generation

The RXFIFO provides two interrupt generation modes, selected by the bit field **GLOBALCON1**.RXFM:

- Single Move Mode
- Batch Move Mode
- Combined Mode

Single Move Mode

The purpose of the Single Move Mode is to keep the RXFIFO as empty as possible, by fetching the received elements one by one as soon as possible.

An example use case could be using single move mode with a DMA performing single move per RXFIFO interrupt. In this mode the DMA keeps the RXFIFO as empty as possible. A DMA request is triggered each time a read from the RXFIFO is performed, and the RXFIFO is afterwards still not empty. If the read makes the RXFIFO empty, an interrupt is not generated in order to prevent underflow. It is generated later when the shift register (or STATUS) writes new element to the empty RXFIFO.

The **Figure 507**, shows the filling levels of the RXFIFO, and the events associated with each filling level triggering a RXFIFO refill interrupt.

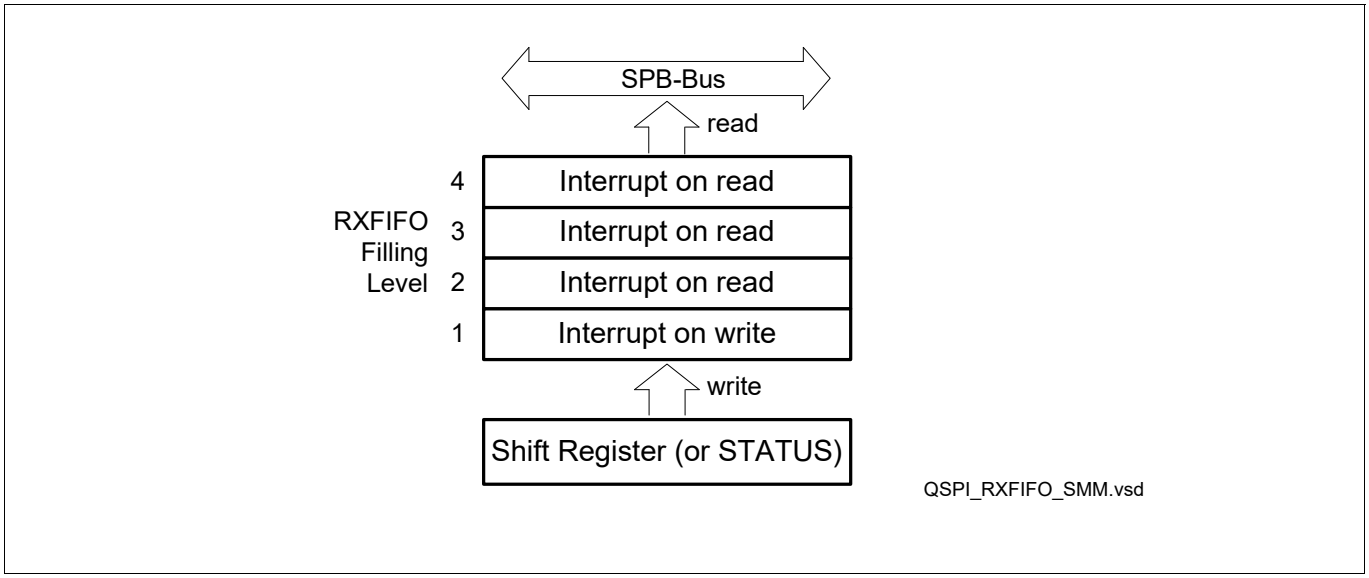


Figure 507 RXFIFO - Interrupt Triggering in the Single Move Mode

The initial RXFIFO interrupt is triggered by the Shift Register, after it delivers the first received element. Afterwards, the DMA trigger-chain of re-fetch interrupts is self sustaining until the whole transaction is over. At the end of the DMA transaction, there is no service request remaining active in the service request node, due to the fact that the read of the last element in the RXFIFO does not trigger an interrupt.

Attention: ***In Single Move Mode multiple software reads or block DMA moves lead to multiple interrupts and (false) transaction lost events. Therefore they should be avoided - only single moves should be used.***

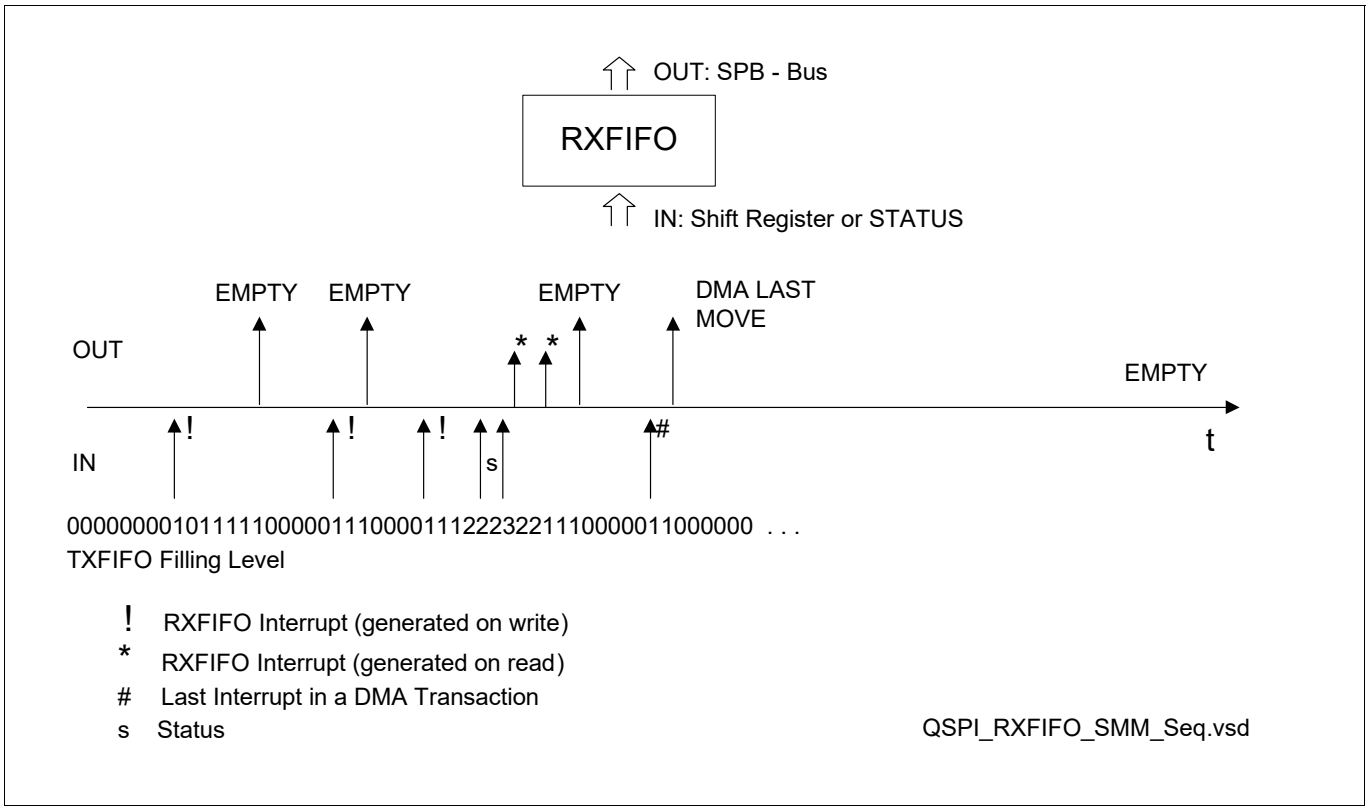


Figure 508 RXFIFO - Interrupt Operation in Single Move Mode

Batch Move Mode

The purpose of the Batch Move Mode is to reduce the number of interrupts by triggering an interrupt when more than one RXFIFO element is full. For example, a CPU can be interrupted less frequently and it can perform more than one move per interrupt.

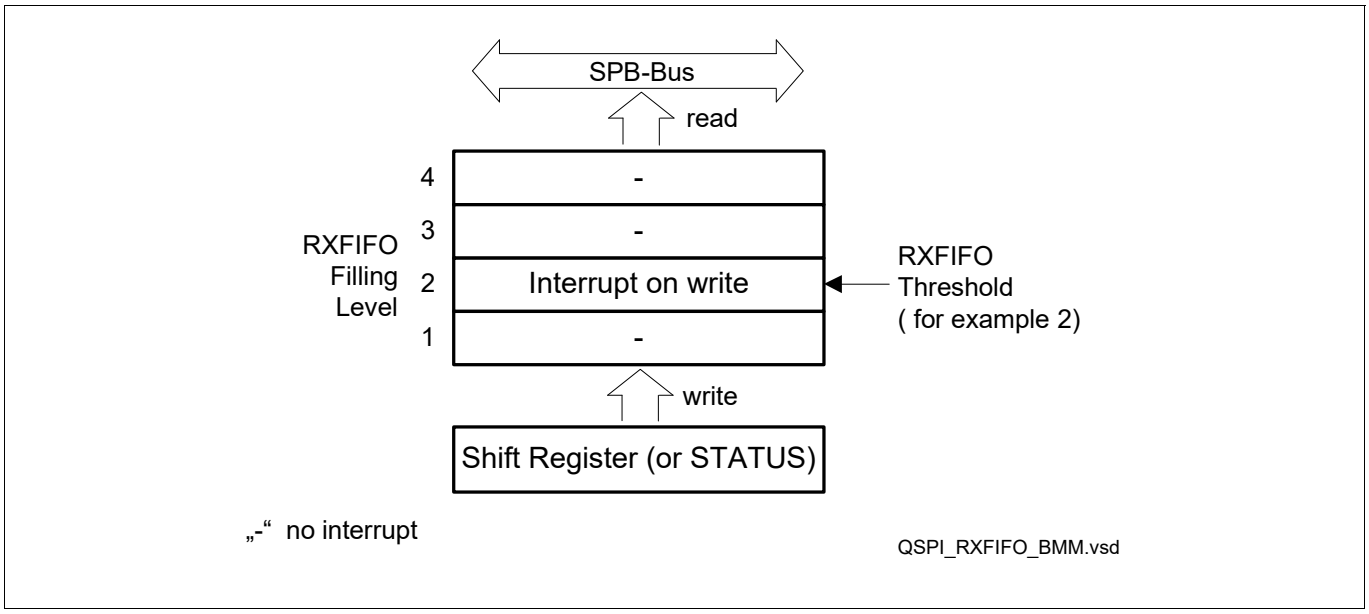


Figure 509 RXFIFO - Interrupt Triggering in the Batch Move Mode

In Batch Move Mode, an interrupt is generated only at one point in the RXFIFO, when the filling level rises above the programmed threshold, implying that there are at least the predefined number of full RXFIFO elements available.

Attention: It must be guaranteed that the CPU keeps emptying the RXFIFO and polling the RXFIFO filling level until the filling level has fallen below the interrupt threshold (or the RXFIFO is empty), in order to guarantee that next interrupt will occur.

At high baud rates and short frames, it could be possible that more than one frame has been received until the moment the RXFIFO is re-emptied. For example, if the threshold is set to two full elements, then the interrupt will be set when the filling level rises above two. If the emptying moment of the RXFIFO is delayed so long that two additional frames have been received (or a frame and STATUS), the RXFIFO will become full, and re-empty sequence of two will not reach the level of two. So, the next interrupt would never come. This effect cannot occur with threshold levels of 3 and 4. The threshold level of 1 is possible, but does not make much sense (use single move mode instead).

Combined Mode

RXFIFO generates interrupt each time a data is written to it above the preprogrammed level, as defined in the **GLOBALCON1**.RXFIFOINT. Consequently, it generates two close interrupts when filled with data and STATUS. If the data RX interrupt is not serviced until the STATUS RX interrupt comes, which will always be the case due to their time proximity, the STATUS RX interrupt will be lost. In such a case, it is recommended that the DMA performs two moves per interrupt trigger (transfer size of two). Otherwise, the RXFIFO average filling level will go up with the time and at the end overflow.

37.3.7 Reset Behavior

There are two sources of reset provided by the module for it's operation:

- BPI (Bus Peripheral Interface) registers **KRST0**/**KRST1**
- Reset bits in the kernel register - **GLOBALCON**.RESETS

BPI reset puts the whole QSPI module in reset state. All outputs get the reset value.

Software writing the **GLOBALCON**.RESETS bit will select what will be Reset (TX_FIFO, RX_FIFO, protocol statemachine, registers)

KRSTx resets the whole Module while **GLOBALCON**.RESETS 11 resets only a subset of all the available registers in the module.

37.3.8 QSPI Module Implementation

This section describes the interfaces of the QSPI module instances with the clock control, port connections, interrupt control, and address decoding.

37.3.8.1 Interfaces of the QSPI Modules

Figure 510 shows the implementation details and interconnections of the QSPI module.

Each of the QSPI modules is supplied with a separate clock control, interrupt control, and address decoding logic. Two interrupt outputs can be used to generate DMA requests. The QSPIx I/O lines are connected to the ports as described in the pinning and ports chapters.

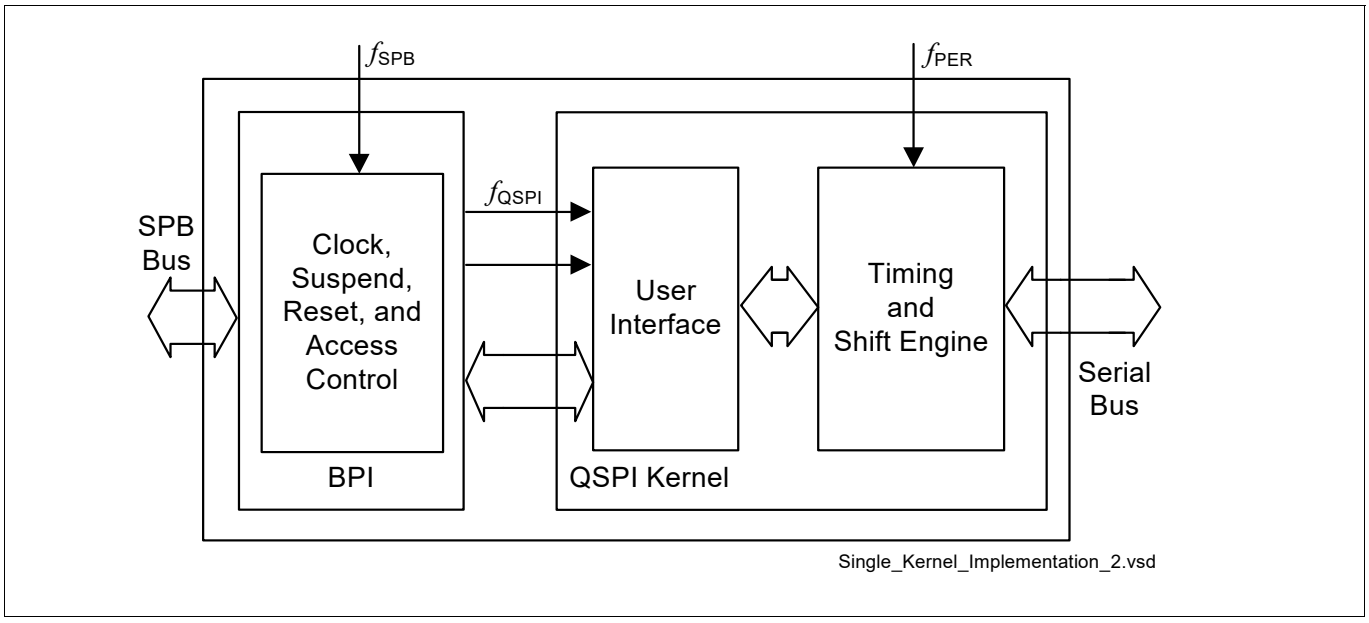


Figure 510 Module Implementation Overview

37.3.8.2 On-Chip Connections

This section describes the on-chip connections of the QSPIx module instances.

SCU Connections

Wake-up from sleep is done by the SCU.ERU triggering a software interrupt on a programmable edge on SLSI pin.

DMA Requests

There are no dedicated DMA request lines. The interrupt or service request signals are general purpose request signals that can be routed to each service provider.

Port/Pin Connections

The connections of the QSPI modules to the pins / ports is described in the chapters regarding the pinning and the ports.

The MTSR and MRST signals are also mapped to LVDS output and input pads. For details see the Ports chapter and the Pinning chapter.

Note: If LVDSH receiver pad is used, the software must take into account the corresponding control and configuration bits, located in the HSCT module and in the corresponding port registers.

Input signals not connected to ports/pins are connected to the following voltage levels:

- The unconnected SCLKI signals are connected to low level “0”
- The unconnected SLSI signals are connected to (inactive) high level “1”
- The unconnected MRST input signals are connected to low level “0”.

Interrupt Connections

The interrupts of the QSPI module are controlled by six service request control registers, located in the IR (Interrupt Router) module:

- QSPIxTX Transmit Interrupt
- QSPIxRX Receive Interrupt
- QSPIxERR Error Interrupt
- QSPIxPT Phase Transition Interrupt
- QSPIxHC High Speed Capture Interrupt
- QSPIxU User Interrupt

37.3.8.3 BPI_FPI Module Registers

Figure 511 shows all registers associated with the BPI_FPI module, configured for one kernel.

BPI_FPI Registers Overview

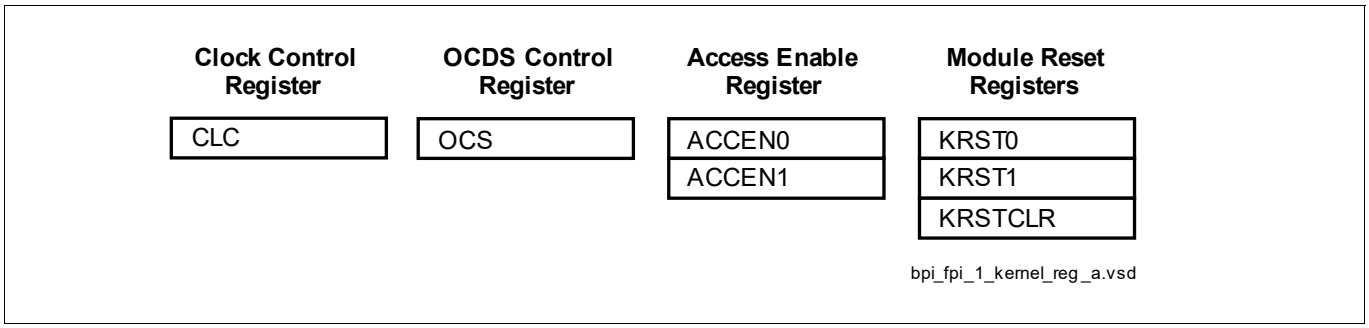


Figure 511 BPI_FPI Registers

The writes of the bus masters to the QSPI module is controlled by Access Protection registers ACCENx.

The QSPI implements two ACCENx registers, ACCEN0 and ACCEN1.

The ACCENx registers are protected by Safety Endinit mechanism.

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the $f_{<>}$ module clock signal, sleep mode and disable mode for the module.

CLC

Clock Control Register **(000_H)** **Application Reset Value: 0000 0003$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Enable Control** Used to enable the module's sleep mode. 0_B Enabled 1_B Disabled
0	2, 31:4	r	**Reserved** Read as 0; Should be written with 0.

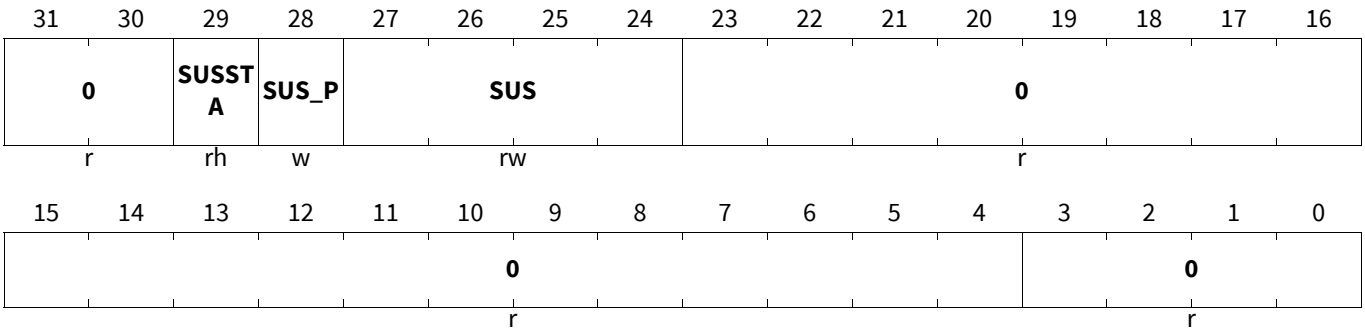
OCDS Control and Status

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32 bit wide only and requires Supervisor Mode

OCS

OCDS Control and Status **($0E8_H$)** **Debug Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. 2_H Soft suspend **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	3:0, 23:4, 31:30	r	**Reserved** Reserved, shall be written with '0' to read '0'.

Access Enable Register 0

The Access Enable Register 0 controls write access for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, ... , EN31 -> TAG ID 011111B. The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

ACCEN0

Access Enable Register 0 **(0FC$_H$)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, ... , EN31 -> TAG ID 111111B. The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers.

ACCEN1

Access Enable Register 1 **(0F8$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

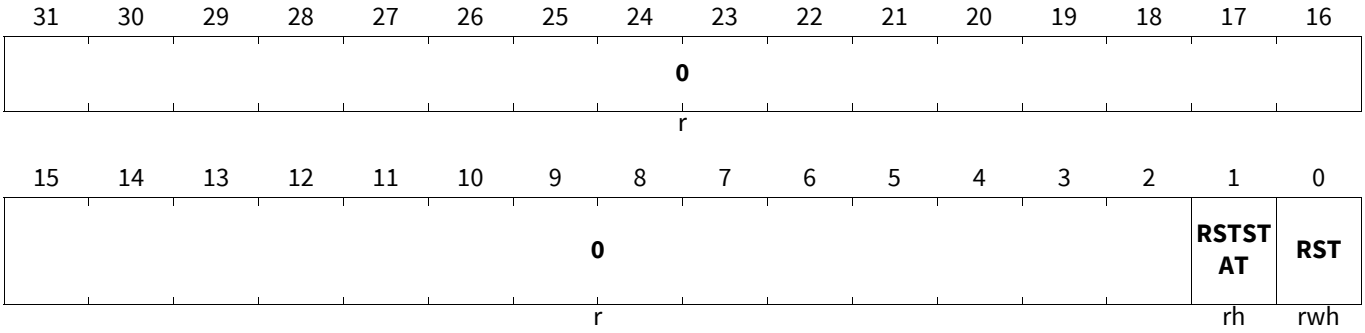
Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **($0F4_H$)** **Application Reset Value: $0000\ 0000_H$**



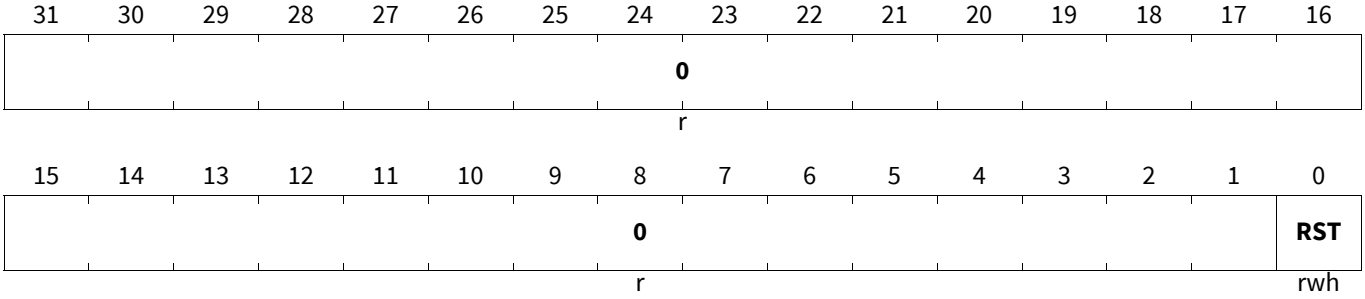
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1

Kernel Reset Register 1 **(0F0$_H$)** **Application Reset Value: 0000 0000$_H$**



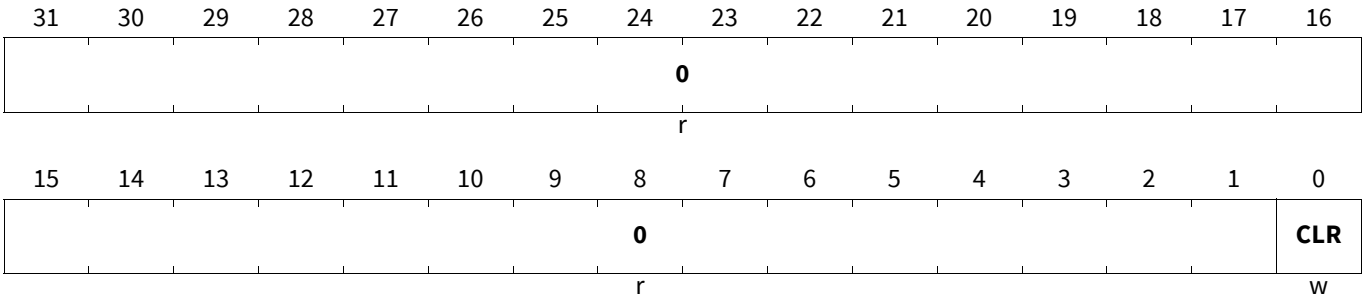
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **(0EC$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

37.3.8.4 Further QSPI Related External Registers

Figure 512 summarizes the module-related external registers which are required for QSPI programming. These are:

- port registers and
- service request node registers,

described in the corresponding chapters.

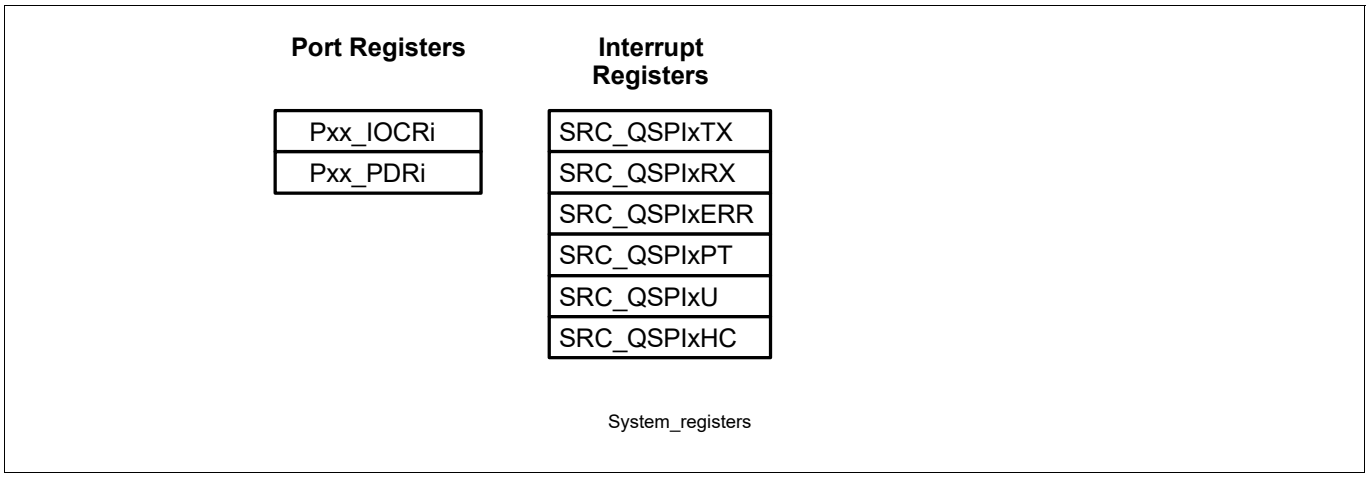


Figure 512 Implementation-specific Special Function Registers

37.4 Functional Description HSIC

The HSIC sub-module provides high precision measurement of relatively low speed input signals. It contains 15-bit timer can be configured to continuously measure:

- whole period (rising edge to rising edge),
- high time (rising edge to falling edge) or
- low time (falling edge to rising edge)
- whole period (falling edge to falling edge)

The resolution of the counter is f_{PER}. The result is provided in a capture register and an interrupt is raised after each capture.

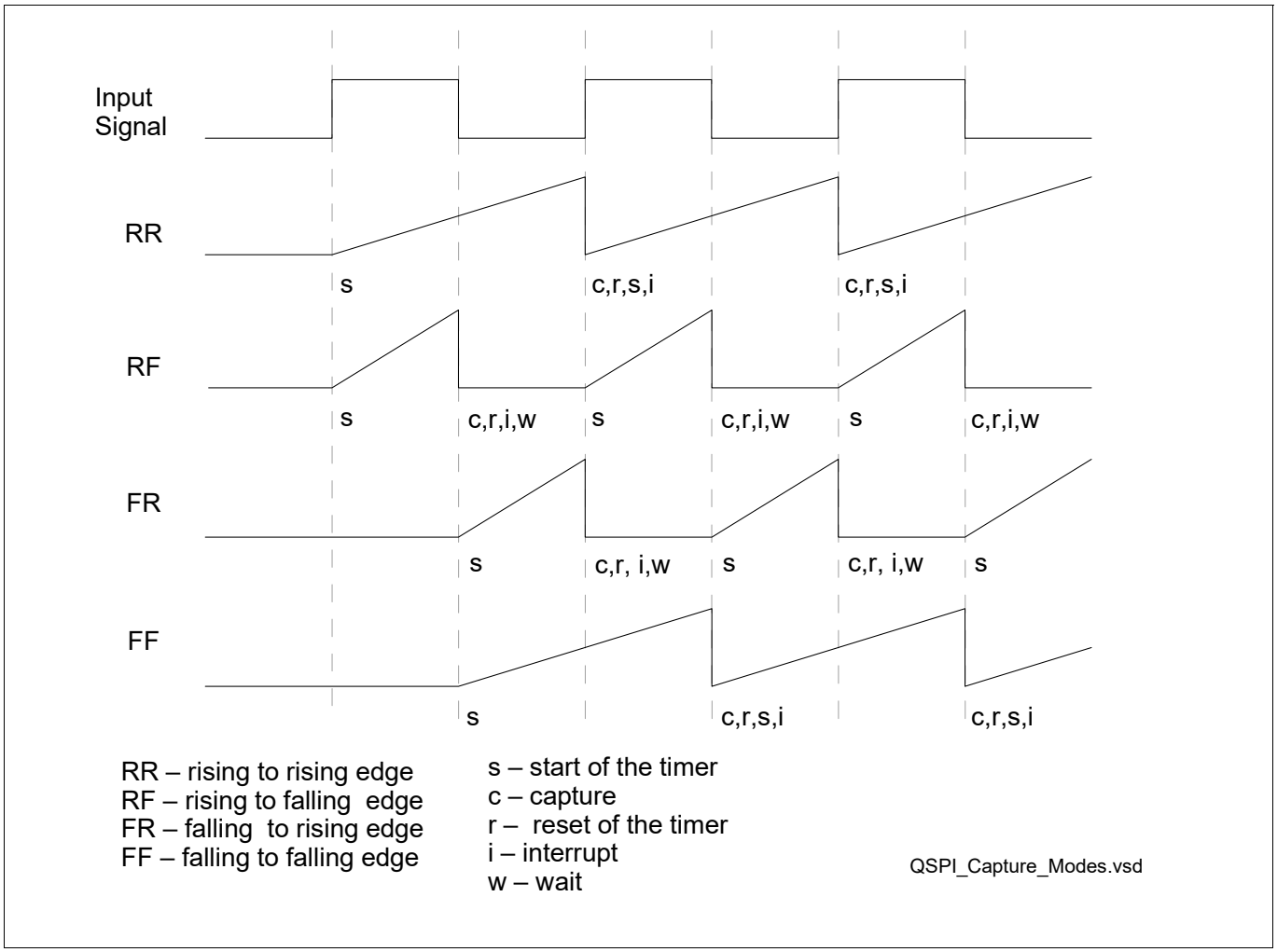


Figure 513 HSIC Modes of Operation

Each capture event causes the content of a 15-bit counter to be captured to the **CAPCON**.CAP bit field, see **Figure 514**.

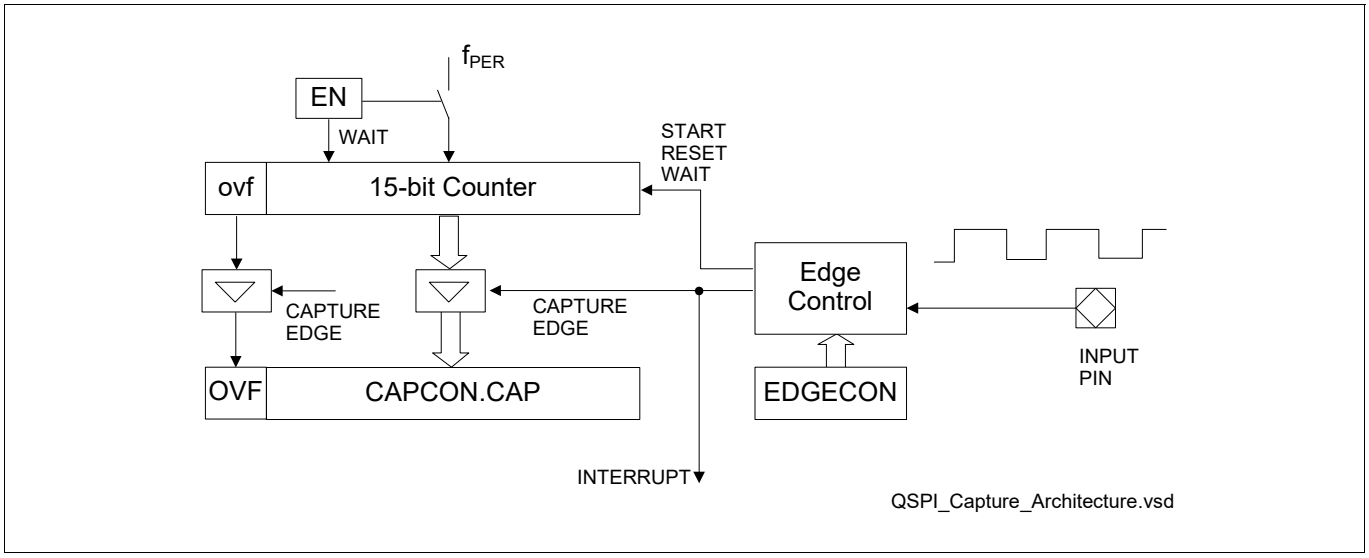


Figure 514 Capture Timer Architecture

The interrupt raised by the capture event is routed to the HC interrupt, see **Figure 515**. The flag bit, set, clear and select bits are located in the **CAPCON** register.

The first interrupt after enabling the HSIC block by setting **CAPCON**.EN may deliver random **CAPCON**.CAP value, depending on the level of the input signal at the moment of enabling. Therefore, the software should discard the first value captured immediately after enabling the HSIC block.

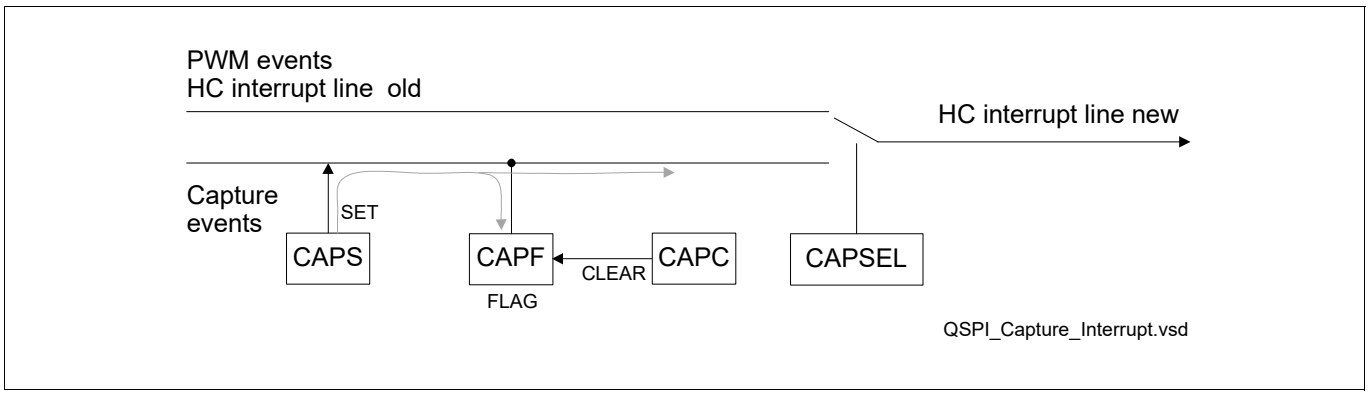


Figure 515 HSIC Interrupt Path

Note: The frequency range of the input signal used for the verification of the HSIC feature should be 3KHz to 30KHz.

Disable Request

Clearing the disable request bit **CLC**.DISR switches the clock of the whole QSPI module including the HSIC block on.

Setting the bit **CAPCON**.EN enables the HSIC sub-module. The counter first enters wait state, waiting for the first start edge as defined in the **CAPCON**.EDGECON. Afterwards, it operates according to the **Figure 513**.

If **CLC**.DISR is cleared when **CAPCON**.EN = 1, the counter is cleared to zero, waits for the next start edge (as defined in **CAPCON**.EDGECON) to start counting, and continues operating according to the **Figure 513**.

Debug Suspend

Suspend of the QSPI module causes suspend of the HSIC; separate suspend is not possible. Nevertheless, the targeted use cases are either HSIC timer active, or QSPI communication, but not both.

Active and enabled hard debug suspend signal stops the HSIC block immediately (QSPI kernel disabled). After the suspend signal is deactivated, the counter continues counting. Writing to a register in hard suspend state causes the HSIC block to receive a couple of clock cycles, which causes the counter to increase its value.

Active and enabled soft debug suspend signal stops the HSIC block immediately (QSPI kernel disabled). After the suspend signal is deactivated, the counter is cleared to zero, waits for the next start edge (as defined in **CAPCON**.EDGECON) to start counting, and continues operating according to the **Figure 513**.

Sleep Mode

Active and enabled sleep signal stops the HSIC block. After the sleep signal is deactivated, the counter is cleared to zero, waits for the next start edge (as defined in **CAPCON**.EDGECON) to start counting, and continues operating according to the **Figure 513**.

Reset

The **GLOBALCON**.RESETS do not reset the HSIC. Other resets (like Application resets ; **KRST0**/**KRST1**) of the QSPI module causes reset of the HSIC sub-module. A separate reset of the HSIC only is not possible.

37.4.1 HSIC Implementation

This Feature is not in all products present. Please look up in the Appendix at Chapter 1.1 IP Specific Configuration if the is HSIC is implemented. The HSIC block is included in the QSPI2 and QSPI3.

For the connections to the pins, see the pinning chapter.

The unconnected input lines are tied to zero.

37.5 Registers

All Registers of the QSPI kernel and the HSIC Sub-module are described in this chapter. Further Registers required for the System integration of the module are explained in the chapter QSPI Module Implementation.

37.5.1 Kernel Registers

This section describes the kernel registers of the QSPI module. All QSPI kernel register names described in this section will be referenced in other parts of the module name prefix “QSPI0_” for the QSPI0 interface, “QSPI1_” for the QSPI1 interface and so on.

All registers in the QSPI address spaces are reset with the application reset.

QSPI Kernel Register Overview

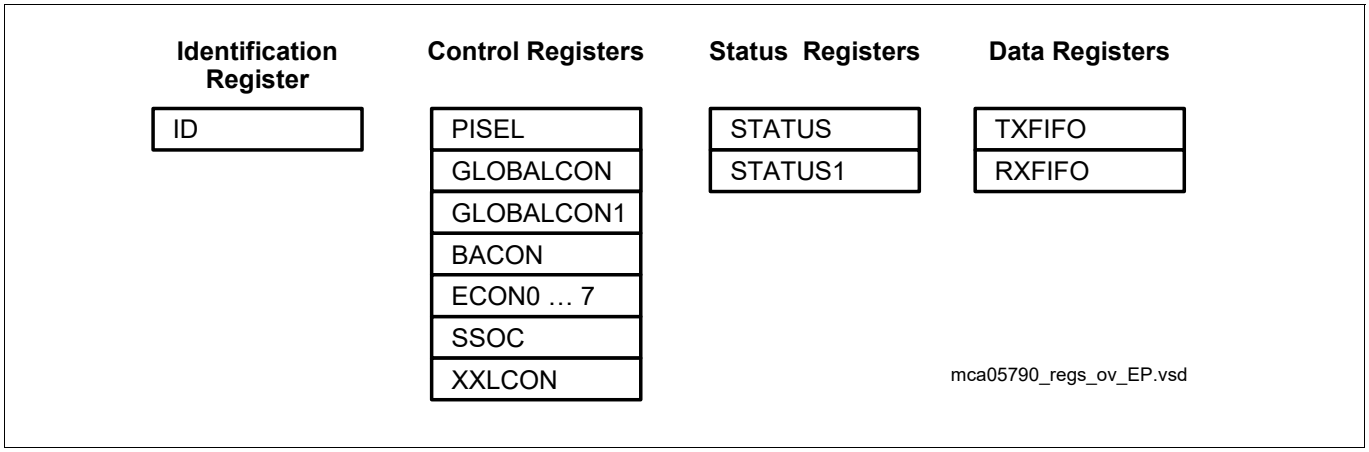


Figure 516 QSPI Kernel Registers

Table 329 Register Overview - QSPI (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	000_H	U,SV	SV,E,P	Application Reset	67
PISEL	Port Input Select Register	004_H	U,SV	SV,P	Application Reset	78
ID	Module Identification Register	008_H	U,SV	BE	Application Reset	80
GLOBALCON	Global Configuration Register	010_H	U,SV	SV,P	Application Reset	80
GLOBALCON1	Global Configuration Register 1	014_H	U,SV	SV,P	Application Reset	83
BACON	Basic Configuration Register	018_H	U,SV	BE	Application Reset	86
ECONz	Configuration Extension z	020_H+z*4	U,SV	SV,P	Application Reset	89
STATUS	Status Register	040_H	U,SV	U,SV,P	Application Reset	90

Table 329 Register Overview - QSPI (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
STATUS1	Status Register 1	044_H	U,SV	U,SV,P	Application Reset	93
SSOC	Slave Select Output Control Register	048_H	U,SV	SV,P	Application Reset	94
FLAGSCLEAR	Flags Clear Register	054_H	U,SV	U,SV,P	Application Reset	95
XXLCON	Extra Large Data Configuration Register	058_H	U,SV	U,SV,P	Application Reset	96
MIXENTRY	MIX_ENTRY Register	$05C_H$	U,SV	U,SV,P	Application Reset	100
BACONENTRY	BACON_ENTRY Register	060_H	U,SV	U,SV,P	Application Reset	101
DATAENTRYx	DATA_ENTRY Register x	064_H+x* 4	U,SV	U,SV,P	Application Reset	101
RXEXIT	RX_EXIT Register	090_H	U,SV	BE	Application Reset	101
RXEXITD	RX_EXIT Debug Register	094_H	U,SV	BE	Application Reset	102
CAPCON	Capture Control Register	$0A0_H$	U,SV	U,SV,P	Application Reset	103
MC	Move Counter Register	$0A4_H$	U,SV	U,SV,P	Application Reset	96
MCCON	Move Counter control Register	$0A8_H$	U,SV	U,SV,P	Application Reset	98
OCS	OCDS Control and Status	$0E8_H$	U,SV	SV,P	Debug Reset	68
KRSTCLR	Kernel Reset Status Clear Register	$0EC_H$	U,SV	SV,E,P	Application Reset	71
KRST1	Kernel Reset Register 1	$0F0_H$	U,SV	SV,E,P	Application Reset	70
KRST0	Kernel Reset Register 0	$0F4_H$	U,SV	SV,E,P	Application Reset	70
ACCEN1	Access Enable Register 1	$0F8_H$	U,SV	SV,SE	Application Reset	69
ACCEN0	Access Enable Register 0	$0FC_H$	U,SV	SV,SE	Application Reset	68

List of Access Protection Abbreviations

U - User Mode

SV - Supervisor Mode

BE - Bus Error

nBE - no Bus Error

P - Access Protection, as defined by the ACCEN Register

E - ENDINIT

SE - Safety ENDINIT

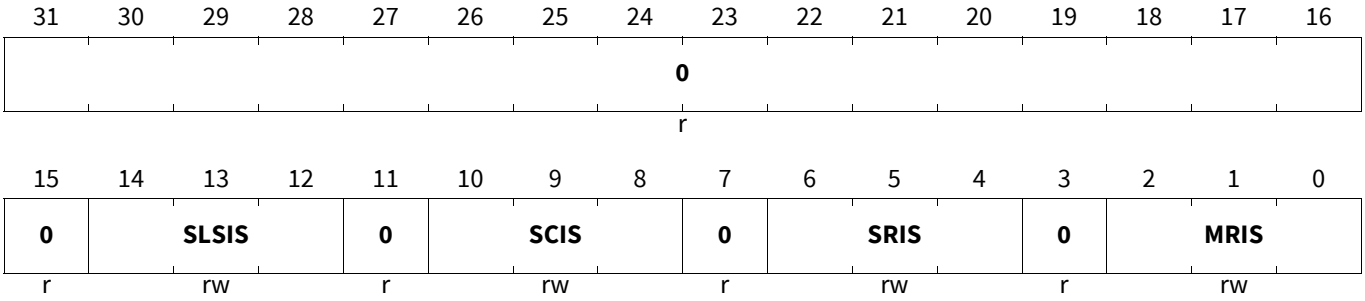
37.5.1.1 Register Description

Port Input Select Register

The PISEL register controls the input signal selection of the SSC module.

PISEL

Port Input Select Register **(004_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
MRIS	2:0	rw	**Master Mode Receive Input Select** MRIS selects one out of eight MRST receive input lines, used in Master Mode. Note that not all inputs are used in every device of the family. Selecting an unused input returns a continuous low value. The following signal sources are available in this product (if supported by the package!) 000_B MRST input line A is selected for operation 001_B MRST input line B is selected for operation 010_B MRST input line C is selected for operation 011_B MRST input line D is selected for operation 100_B MRST input line E is selected for operation 101_B MRST input line F is selected for operation 110_B MRST input line G is selected for operation 111_B MRST input line H is selected for operation

Field	Bits	Type	Description
SRIS	6:4	rw	**Slave Mode Receive Input Select** SRIS selects one out of eight MTSR receive input lines, used in Slave Mode. Note that not all inputs are used in every device of the family. Selecting an unused input returns a continuous low value. The following signal sources are available in this product (if supported by the package!) 000_B MTSR input line A is selected for operation 001_B MTSR input line B is selected for operation 010_B MTSR input line C is selected for operation 011_B MTSR input line D is selected for operation 100_B MTSR input line E is selected for operation 101_B MTSR input line F is selected for operation 110_B MTSR input line G is selected for operation 111_B MTSR input line H is selected for operation
SCIS	10:8	rw	**Slave Mode Clock Input Select** SCIS selects one out of eight module kernel SCLK input lines that is used as clock input line in slave mode. Note that not all inputs are used in every device of the family. Selecting an unused input returns a continuous low value. The following signal sources are available in this product (if supported by the package!) 000_B SCLK input line A is selected for operation 001_B SCLK input line B is selected for operation 010_B SCLK input line C is selected for operation 011_B SCLK input line D is selected for operation 100_B SCLK input line E is selected for operation 101_B SCLK input line F is selected for operation 110_B SCLK input line G is selected for operation 111_B SCLK input line H is selected for operation
SLSIS	14:12	rw	**Slave Mode Slave Select Input Selection** The SLSIS must be programmed properly before the slave mode is set with GLOBALCON.MODE and the module is set to RUN mode. The following signal sources are available in this product (if supported by the package!) 000_B Slave select input lines are deselected QSPI is operating without slave select input functionality. 001_B SLSI input line A is selected for operation 010_B SLSI input line B is selected for operation 011_B SLSI input line C is selected for operation 100_B SLSI input line D is selected for operation 101_B SLSI input line E is selected for operation 110_B SLSI input line F is selected for operation 111_B SLSI input line G is selected for operation
0	3, 7, 11, 31:15	r	**Reserved** Read as 0; should be written with 0.

Module Identification Register

The Module Identification Register ID contains read-only information about the module version.

ID
Module Identification Register **(008$_H$)** **Application Reset Value: 00C0 C0XX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODNUMBER															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODTYPE								MODREV							
r								r							

Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** MODREV defines the module revision number. The value of a module revision starts with 01$_H$ (first revision).
MODTYPE	15:8	r	**Module Type** This bit field is C0$_H$. It defines a 32-bit module.
MODNUMBER	31:16	r	**Module Number Value** This bit field together with MODTYPE uniquely identifies a module.

Global Configuration Register

GLOBALCON contains configuration parameters which affect all channels.

Note: If the EN bit is cleared first, and then the partial state machine reset activated, then the state machine goes into PAUSE state.

*Note: Switching the peripheral clock off and on is done by using two writes: first a write of zero to the CLKSEL bit field, and then a second write switching on the clock source. Between the first and the second write a delay of minimum 4 * (1/f_{PER}) + 2 * (1/f_{CLC}) must be inserted by software, where f_{PER} is the frequency being switched off with the first write. This also applies if between the writes the clock frequency is changed in the CCU.*

GLOBALCON
Global Configuration Register **(010$_H$)** **Application Reset Value: 000F 30FF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESETS		CLKSEL	0	AREN	MS		EN	0	STIP	SRF	STROBE				
w		rw	r	rw	rw		rwh	r	rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEL0	LB	EXPECT				SI	0	TQ							
rw	rw	rw				rw	r	rw							

Field	Bits	Type	Description
TQ	7:0	rw	**Global Time Quantum Length** Common n-divider scaling the baud rates of all channels in direction of higher or lower baud rates. Must not be changed during a running transaction. 00_H division by 1 01_H division by 2 … FF_H division by 256
SI	9	rw	**Status Injection** Selects if the status register content injection into the RxFIFO is enabled or disabled. The status injections, if enabled, is performed after each data block, depending on the **BACON**.DL and BYTE setting. 0_B Disabled 1_B Enabled
EXPECT	13:10	rw	**Time-Out Value for the Expect Phase** expressed in T_{QSPI} units 0_H 64 (2^6) 1_H 128 (2^7) … F_H 2.097152E6 (2^21)
LB	14	rw	**Loop-Back Control** Selects if the transmit output is internally connected to the receive input for test purposes. For detailed description, see the Loop-Back Mode section. 0_B Loop-Back inactive 1_B Loop-Back active
DEL0	15	rw	**Delayed Mode for SLSO0** Switches the delayed mode (external slave select expansion mode) on and off 0_B Delayed mode off 1_B Delayed mode on
STROBE	20:16	rw	**Strobe Delay for SLSO0 in Delayed Mode** Defines the length of the SLSO0 delay in T_Q time units as defined for channel z (T_Q units) selected by the current **BACON**.CS, if **GLOBALCON**.DEL0 = 1. 00_H 1 01_H 2 … $1F_H$ 32

Field	Bits	Type	Description
SRF	21	rw	**Stop on RxFIFO Full** If this bit is set, the data fetching out of the TxFIFO by the shift register stops when the RxFIFO is full, in order to prevent RxFIFO overflow. Master mode only. 0_B Feature disabled 1_B Feature enabled
STIP	22	rw	**Slave Transmit Idle State Polarity** This bit determines the logic level of the Slave Mode transmit signal MRST when the QSPI slave select input signals are inactive (PISEL.SLSIS ¹ 000_B). 0_B MRST = 0 when QSPI is deselected in Slave Mode. 1_B MRST = 1 when QSPI is deselected in Slave Mode
EN	24	rwh	**Enable Bit** Used to request transition between PAUSE and RUN mode per software. Cleared by hardware automatically at leaving the following states: disabled, suspend and sleep. In order to determine if the requested state has actually been reached, the **STATUS**.PHASE bit field should be polled. See also Operation Modes. 0_B PAUSE requested 1_B RUN requested
MS	26:25	rw	**Master Slave Mode** Selects if the module operates in master or slave mode. This bit field must be configured before the first write to the TXFIFO. 00_B Master Transmit and Receive 01_B Reserved 10_B Slave Transmit and Receive 11_B Slave Transmit and Receive
AREN	27	rw	**Automatic Reset Enable** Enables the reset of the **GLOBALCON**.EN on baud rate and spike error in slave mode. 0_B Disabled 1_B Enabled
CLKSEL	29	rw	**Clock Select** Selects the clock source for the asynchronous block. 0_B no clock 1_B f_{PER}

Field	Bits	Type	Description
RESETS	31:30	w	**Bits for resetting sub-modules per software** Write to this bit field triggers a reset operation. The duration of reset operation depends on the used clock setups. It is 10 times of the slowest clock between SPB and Kernel. *Note: CLOCKSEL shall be enabled to use GLOBALCON.RESETS. After the Reset the CLOCKSEL will be also cleared in case of 10 11.* For resetting the whole module kernel, use alternatively the registers **KRST0** / **KRST1** reset mechanism. 00_B No reset triggered 01_B State Machine, TXFIFO and RXFIFO reset, registers not reseted. Only control structures shall be reset. It avoids a re-configuration of the whole IP. After this reset Communication can be restarted. 10_B Registers reset. Clears the configuration of the QSPI registers. New setup have to be programmed. 11_B State Machine, TXFIFO and RXFIFO reset and registers reset
0	8, 23, 28	r	**Reserved** Read as 0; should be written with 0.

Global Configuration Register 1

GLOBALCON1 contains bit fields for control of the extension of the slave select signals by using an external demultiplexer.

GLOBALCON1
Global Configuration Register 1 (014_H) Application Reset Value: $0005\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**RXFM**		**TXFM**		**PT2**			**PT1**			**RXFIFOINT**		**TXFIFOINT**	
r	r	rw		rw		rw			rw			rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USREN	**0**	**0**	**PT2EN**	**PT1EN**	**RXEN**	**TXEN**	**ERRORENS**								
rw	r	r	rw	rw	rw	rw	rw								

Field	Bits	Type	Description
ERRORENS	8:0	rw	**Error Enable Bits** Bits for enabling interrupt on all available error types 000_H All errors disabled 001_H Parity Error (PAREEN) 002_H Unexpected Configuration Error 004_H Baud Rate Error (slave mode) BEN 008_H TXFIFO overflow (software error) 010_H TXFIFO underflow (slave mode) TEN 020_H RXFIFO overflow REN 040_H RXFIFO underflow (software error) 080_H EXPECT timeout 100_H SLSI misplaced inactivation enable
TXEN	9	rw	**Tx Interrupt Event Enable** Enables the Tx interrupt. 0_B Disabled 1_B Enabled
RXEN	10	rw	**RX Interrupt Event Enable** Enables the Rx interrupt. 0_B Disabled 1_B Enabled
PT1EN	11	rw	**Interrupt on PT1 Event Enable** Enables the PT interrupt on an PT1 event, as selected by the PT1 bit field. 0_B Disabled 1_B Enabled
PT2EN	12	rw	**Interrupt on PT2 Event Enable** Enables the PT interrupt on an PT2 event, as selected by the PT2 bit field. 0_B Disabled 1_B Enabled
USREN	15	rw	**Interrupt on USR Event Enable** Enables the USR interrupt on an USR event, as selected by the PT1 bit field. *Note: In Move Counter mode of operation, User Interrupt line is reused for the IAL and IBL interrupts, which can be enabled by **MCCON**.IALEN and **MCCON**.IBLEN. Bit USREN does not influence these interrupts.* 0_B Disabled 1_B Enabled

Field	Bits	Type	Description
TXFIFOINT	17:16	rw	**Transmit FIFO Interrupt Threshold** In Combined Mode, as long as the TXFIFO filling level is equal or less than this threshold, than each move of data or configuration from the TXFIFO triggers a transmit interrupt. In Batch Mode, interrupt is generated only at the moment of filling level falling below the threshold level. In Single Mode, this bit field is don't care. Reset value of the level is 01_B. 00_B 1 01_B 2 10_B 3 11_B 4
RXFIFOINT	19:18	rw	**Receive FIFO Interrupt Threshold** In Combined Mode, as long as the RXFIFO filling level is equal or greater than this threshold, than each move of data or status (if enabled) into the RXFIFO triggers a receive interrupt. In Batch mode, interrupt is generated only at the moment of filling level exceeding this threshold level. In Single Mode, this bit field is don't care. Reset value of the level is 01_B. 00_B 0 01_B 1 10_B 2 11_B 3
PT1	22:20	rw	**Phase Transition Event 1** Selects the first phase transition to trigger the PT interrupt. 000_B BUSY Master Mode: End of WAIT phase Slave Mode: Transmit data is present waiting for the shift clock 001_B SCLKPC Master Mode: Serial clock polarity change 010_B SOF Master Mode: Start of Frame Slave Mode: Transmission of the first data bit started 011_B TBE Master Mode: Transmit Buffer Emptied Slave Mode: Data is taken from the TXFIFO by the QSPI shift engine 100_B RBF Master Mode: Receive Buffer Filled Slave Mode: Received data is written to the RXFIFO) 101_B EOF Master Mode: End of Frame Slave Mode: The last data bit has been received 110_B DNA Master Mode: Data not Available = Start of Expect 111_B CONT Master Mode: End of EXPECT phase

Field	Bits	Type	Description
PT2	25:23	rw	**Phase Transition Event 2** Selects the second phase transition to trigger the PT2 interrupt. In Master Mode, the following events are available. In Slave Mode, only the SLSI signal (rising edge) triggers the interrupt for PT2. For this purpose, always use the setting 101 (EOF). This interrupt is independent of the CS value. 000_B BUSY Master Mode: end of WAIT phase 001_B SCLKPC Master Mode: serial clock polarity change 010_B SOF Master Mode: Start of Frame 011_B TBE Master Mode: Transmit Buffer Emptied 100_B RBF Master Mode: Receive Buffer Filled 101_B EOF Master Mode: End of Frame Slave Mode: SLSI deactivated (rising edge on the SLSI pin) 110_B DNA Master Mode: Data not Available = Start of Expect 111_B CONT Master Mode: End of EXPECT phase
TXFM	27:26	rw	**TXFIFO Mode** Selects the TXFIFO mode. 00_B Combined Move Mode 01_B Single Move Mode 10_B Batch Move Mode 11_B reserved
RXFM	29:28	rw	**RXFIFO Mode** Selects the RXFIFO mode. 00_B Combined Move Mode 01_B Single Move Mode 10_B Batch Move Mode 11_B Reserved
0	13, 14, 30, 31	r	**Reserved** Read as 0; should be written with 0.

Basic Configuration Register

Defines the basic configuration parameters for the current slave select. It can be read by software, or written through a write to the TXFIFO.

If an application does not use the SLSO signals of the QSPI module, but uses software channel selection, then the alternate function multiplexer in the ports must be configured so that the signal SLSO is not selected, but the corresponding bit Pn.OUT.

Alternatively, to emulate SLSO functionality, the software can disable (low) all the **SSOC**.OEN bits, and then toggle the **SSOC**.AOL bits.

BACON
Basic Configuration Register **(018_H)** **Application Reset Value: 0F87 1C71$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CS				DL					BYTE	MSB	UINT	PARTYP	TRAIL		
rh				rh					rh	rh	rh	rh	rh		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPRE			LEAD			LPRE			IDLE			IPRE			LAST
rh			rh			rh			rh			rh			rh

Field	Bits	Type	Description
LAST	0	rh	**Last Word in a Frame** Defines if the following data word is last in the current frame or not 0_B Not Last 1_B Last
IPRE	3:1	rh	**Prescaler for the Idle Delay** Length in T_{PER} units 000_B 1 001_B 4 010_B 16 … 111_B 16384
IDLE	6:4	rh	**Idle Delay Length** Defines the length of both idle delays, IDLEA and IDLEB, in T_{PER} units pre scaled with IPRE 000_B 1 unit 001_B 2 units … 111_B 8 units
LPRE	9:7	rh	**Prescaler for the Leading Delay** Length in T_{PER} units 000_B 1 001_B 4 010_B 16 … 111_B 16384
LEAD	12:10	rh	**Leading Delay Length** Defines the length of the leading delay, in T_{PER} units pre scaled with LPRE 000_B 1 unit 001_B 2 units … 111_B 8 units

Field	Bits	Type	Description
TPRE	15:13	rh	**Prescaler for the Trailing Delay** Length in T_{PER} units 000_B 1 001_B 4 010_B 16 … 111_B 16384
TRAIL	18:16	rh	**Trailing Delay Length** Defines the length of the trailing delay, in T_{PER} units pre scaled with TPRE 000_B 1 unit 001_B 2 units … 111_B 8 units
PARTYP	19	rh	**Parity Type** Valid for both receive and transmit direction 0_B Even parity 1_B Odd parity
UINT	20	rh	**User Interrupt at the PT1 Event in the Subsequent Frames** This bit is an enable signal for the PT1 event routed to the User Interrupt Service Request. The interrupt signals are generated until disabled with the next BACON. 0_B Disabled 1_B Enabled
MSB	21	rh	**Shift MSB or LSB First** This bit sets the shift direction of the shift register. If the MSB option is set, and the data is a block longer than 32 bits, the block must be fed into the TXFIFO in reverse direction, from the end of the block until its beginning. 0_B Shift LSB first 1_B Shift MSB first
BYTE	22	rh	**Byte** Defines if data length is expressed in bits or bytes. In Slave Mode BYTE must be '0'. 0_B DL defines the data length in bits 1_B DL defines the data length in bytes
DL	27:23	rh	**Data Length** Defines the data length in bits or bytes of one data block, depending on the setting of the bit field BYTE. For the maximum baud rate of 50 MBaud, the minimum data length possible is four. 00_H 2 bits if BYTE=0; XXL mode if BYTE=1 01_H 2 bits or bytes … $1F_H$ 32 bits or bytes

Field	Bits	Type	Description
CS	31:28	rh	**Channel Select** Selects the channel to which the subsequent data entry belongs (channel = the SLSO signal to be activated and the corresponding ECON configuration extension) This bit field selects one slave in a range of 0 to 15, by driving one SLSO signal out of 16 available. In case of an external demux mode, this bit field appears on the lines SLS01 to SLSO4 as it is, additionally inverted or not, as defined in the **SSOC** register.

Configuration Extension z

Configuration extensions for channels 0 to 15. Register x defines several timing characteristics for two channels: z and z+8.

ECONz (z=0-7)
Configuration Extension z **(020_H+z*4)** **Application Reset Value: 0000 1450_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BE		0													
rw		r													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PAREN	CPOL	CPH	C		B		A		Q					
r	rw	rw	rw	rw		rw		rw		rw					

Field	Bits	Type	Description
Q	5:0	rw	**Time Quantum** Defines the time quantum length used by A, B, and C to define the baud rate and duty cycle by. This prescaler cascades the prescaler GLOBALCON.TQ. 00_H 1 01_H 2 … $3F_H$ 64
A	7:6	rw	**Bit Segment 1** Length expressed in time quantums of ECONz.Q. 00_B 1 01_B 2 10_B 3 11_B 4

Field	Bits	Type	Description
B	9:8	rw	**Bit Segment 2** Length expressed in time quantums of ECONz.Q. 00_B 0 01_B 1 10_B 2 11_B 3
C	11:10	rw	**Bit Segment 3** Length expressed in time quantums of ECONz.Q. 00_B 0 (if B=0, than C is minimum 1 per hardware) 01_B 1 10_B 2 11_B 3
CPH	12	rw	**Clock Phase** Delay of one half SCLK clock cycle. 0_B Disabled 1_B Enabled
CPOL	13	rw	**Clock Polarity** Idle level of the shift clock signal at the SCLK pin 0_B Idle level low 1_B Idle level high
PAREN	14	rw	**Enable Parity Check** This bit field enables both the parity generation in transmit and parity check in receive direction. 0_B Disabled 1_B Enabled
BE	31:30	rw	**Permutate bytes to / from Big Endian** 00_B Disabled 01_B 16-bit big endian 10_B 32-bit big endian 11_B Disabled
0	29:15	r	**Reserved**

Status Register

Status register contains bits flagging the current status of the module and its sub-modules.

Note: It is not recommended to set the STATUS register flags per software for purposes other than testing. In such cases, use write instructions only, not read-modify-write instructions or bit instructions which compile to read-modify-write instructions.

3. When using the RXFIFO status injection feature, only bits 22 to 31 reflect the status at the moment of injection, that is for the latest frame. Due to pipeline effects, the other bits contain not-latest information.

Note: Slave TXFIFO underflow error is activated if the transmission has been started by the master at the time when the slave FIFO was empty, or at the same moment updated. In the second case inconsistent data will be transmitted.

4. *Reading the TXFIFO filling level bit field returns a value which can be used to calculate how many writes can be performed by a CPU without causing an overflow (in case no DMA is programmed to access the TXFIFO in parallel). For example, if the TXFIFO level was one, maximum three (TXFIFO depth of four minus one) write accesses are possible. Due to the volatility of the bit field, the filling level can go down in some nanoseconds after the read.*
5. *Reading the RXFIFO filling level bit field returns a value which shows directly how many reads can be performed by a CPU without causing an underflow. Due to the volatility of the bit field, the filling level can go up in some nanoseconds after the read.*
6. *Reading the PHASE bit field shows a previous phase of the frame, and a PT1F and PT2F flags indicate previous phase transitions. If the communication speed is not too high or the phases duration not very short compared to the software latency delays, which would normally be the case, these would be the latest completed phase and the latest phase transitions. Nevertheless, in case of high baud rates, the duration of the phases must be taken into consideration, but only if these bit fields are used in an application, and not only for debugging purposes.*

STATUS

Status Register **(040_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PHASE				TPV	RPV	SLAVESEL				RXFIFOLEVEL			TXFIFOLEVEL		
rh				rh	rh	rh				rh			rh		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USRF	0	0	PT2F	PT1F	RXF	TXF	ERRORFLAGS								
rwh	r	r	rwh	rwh	rwh	rwh	rwh								

Field	Bits	Type	Description
ERRORFLAGS	8:0	rwh	**Sticky Flags Signalling Errors** Writing 1 sets the error Flag and triggers an error interrupt, if enabled. Writing 0 has no effect. 000_H No Error 001_H Parity Error 002_H Unexpected Configuration Error 004_H Baud Rate Error (slave mode) 008_H TXFIFO overflow (software error) 010_H TXFIFO underflow (slave mode) 020_H RXFIFO overflow 040_H RXFIFO underflow (software error) 080_H EXPECT time out error 100_H SLSI misplaced inactivation (slave mode)
TXF	9	rwh	**Transmit Interrupt Request Flag** Flags an occurrence of a request to feed the TXFIFO, which is generated when an element is fetched from the FIFO, and the FIFO filling level is equal or less than the set threshold level. Writing 1 sets the flag and triggers an interrupt if **GLOBALCON1**.TXEN = 1. Writing 0 has no effect.

Field	Bits	Type	Description
RXF	10	rwh	**Receive Interrupt Request Flag** Flags an occurrence of a request to empty the RXFIFO, which is generated when an element is written into the FIFO, and the FIFO filling level is equal or greater than the set threshold level. Writing 1 sets the flag and triggers an interrupt if **GLOBALCON1**.RXEN = 1. Writing 0 has no effect.
PT1F	11	rwh	**Phase Transition 1 Flag** Flags an occurrence of a PT1 event, as selected with the **GLOBALCON1**.PT1, and triggers an interrupt if **GLOBALCON1**.PT1EN = 1. Writing 1 sets the flag and triggers an error interrupt. Writing 0 has no effect.
PT2F	12	rwh	**Phase Transition 2 Flag** In master mode, flags an occurrence of a PT2 event, as selected with the **GLOBALCON1**.PT2, and triggers an interrupt if **GLOBALCON1**.PT2EN = 1. In slave mode, set by the SLSI deactivated event. Writing 1 sets the flag and triggers an error interrupt. Writing 0 has no effect.
USRF	15	rwh	**User Interrupt Request Flag** Flags an occurrence of an USR event. Writing 1 sets the flag and triggers an interrupt if **GLOBALCON1**.USREN = 1. Writing 0 has no effect.
TXFIFOLEVEL	18:16	rh	**TXFIFO Filling Level** Shows how many entries in the TXFIFO are waiting for transmission 000_B 0 001_B 1 010_B 2 011_B 3 100_B 4 **others**, reserved
RXFIFOLEVEL	21:19	rh	**RXFIFO Filling Level** Shows how many entries in the RXFIFO are waiting for software to move them to RAM 000_B 0 001_B 1 010_B 2 011_B 3 100_B 4 **others**, reserved
SLAVESEL	25:22	rh	**Currently Active Slave Select Flag** Displays the currently active slave select.
RPV	26	rh	**Received Parity Value** Displays the last received parity bit, if parity was enabled. Else if the parity is disabled, reads 0.

Field	Bits	Type	Description
TPV	27	rh	**Transmitted Parity Value** Displays the last transmitted parity bit, if parity was enabled. Else 0.
PHASE	31:28	rh	**Flags the ongoing phase** Displays the current phase number. Relevant only in master mode. In slave mode this bit field indicates always 0. Not 0 means busy. 0_H Wait 1_H Idle A 2_H Idle B 3_H Lead 4_H Data 5_H Trail 6_H Expect 7_H Lead Strobe 8_H Trail Strobe
0	13, 14	r	**Reserved** Read as 0; should be written with 0.

Status Register 1

If the **STATUS**.ERRORFLAGS[2] bit is cleared via **FLAGSCLEAR**.ERRORCLEARS[2], the both flags SPD and BRD are automatically cleared.

STATUS1
Status Register 1 **(044_H)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPD	**SPDEN**	**BRD**	**BRDEN**	**0**											
rwh	rw	rwh	rw	r											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								**BITCOUNT**							
r								r							

Field	Bits	Type	Description
BITCOUNT	7:0	r	**Number of bits shifted out** Supports up to 256 bits. A BITCOUNT value of greater than 0 indicates that a transmission is in progress. The value is not accurate; it may be lower than the number of bits shifted. After transmission of the last bit, BITCOUNT is set to zero.
BRDEN	28	rw	**Baud Rate Deviation Enable** Enables the signal path. 0_B Disabled 1_B Enabled

Field	Bits	Type	Description
BRD	29	rwh	**Baud Rate Deviation Flag** Shows if baud rate deviation has been detected. Write of 1 sets the bit and raises the event per software. Write of 0 has no effect. 0_B no event 1_B event detected
SPDEN	30	rw	**Spike Detection Enable** Enables the signal path. 0_B Disabled 1_B Enabled
SPD	31	rwh	**Spike Detection Flag** Shows if spike has been detected. Write of 1 sets the bit and raises the event per software. Write of 0 has no effect. 0_B No event 1_B Event detected
0	27:8	r	**Reserved** Read as 0; should be written with 0.

Slave Select Output Control Register

SSOC controls the level of each slave select and enables/disables each one individually.

SSOC

Slave Select Output Control Register (048$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEN															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AOL															
rw															

Field	Bits	Type	Description
AOL	15:0	rw	**Active Output Level for the SLSO Outputs** The idle level is the inverted one. “0” at certain position means active low level for the corresponding SLSO. “1” means active high.
OEN	31:16	rw	**Enable Bits for the SLSO Outputs** In disabled state the SLSO output drives the idle level as defined by the AOL bit field. “0” at certain position means that the corresponding SLSO is disabled. “1” means enabled.

Flags Clear Register

FLAGSCLEAR
Flags Clear Register **(054_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USRC	0	0	PT2C	PT1C	RXC	TXC	ERRORCLEARS								
w	r	r	w	w	w	w	w								

Field	Bits	Type	Description
ERRORCLEARS	8:0	w	**Write Only Bits for Clearing the Error Flags** Writing 1 clears the corresponding error flag in the ERORRFLAGS bit field. Reading returns 0. 000_H No clear 001_H Parity Error clear 002_H Unexpected Configuration Error clear 004_H Baud Rate Error clear 008_H TXFIFO overflow clear 010_H TXFIFO underflow clear 020_H RXFIFO overflow clear 040_H RXFIFO underflow clear 080_H EXPECT time out clear 100_H SLSI misplaced inactivation clear
TXC	9	w	**Transmit Event Flag Clear** Write of 1 clears the **STATUS**.TXF bit. Write of 0 has no effect. Read delivers 0. 0_B No action 1_B Clear
RXC	10	w	**Receive Event Flag Clear** Write of 1 clears the **STATUS**.RXF bit. Write of 0 has no effect. Read delivers 0. 0_B No action 1_B Clear
PT1C	11	w	**PT1 Event Flag Clear** Write of 1 clears the **STATUS**.PT1F bit. Write of 0 has no effect. Read delivers 0. 0_B No action 1_B Clear
PT2C	12	w	**PT2 Event Flag Clear** Write of 1 clears the **STATUS**.PT2F bit. Write of 0 has no effect. Read delivers 0. 0_B No action 1_B Clear

Field	Bits	Type	Description
USRC	15	w	**User Event Flag Clear** Write of 1 clears the **STATUS**.USRF bit. Write of 0 has no effect. Read delivers 0. 0_B No action 1_B Clear
0	13, 14, 31:16	r	**Reserved** Read as 0; should be written with 0.

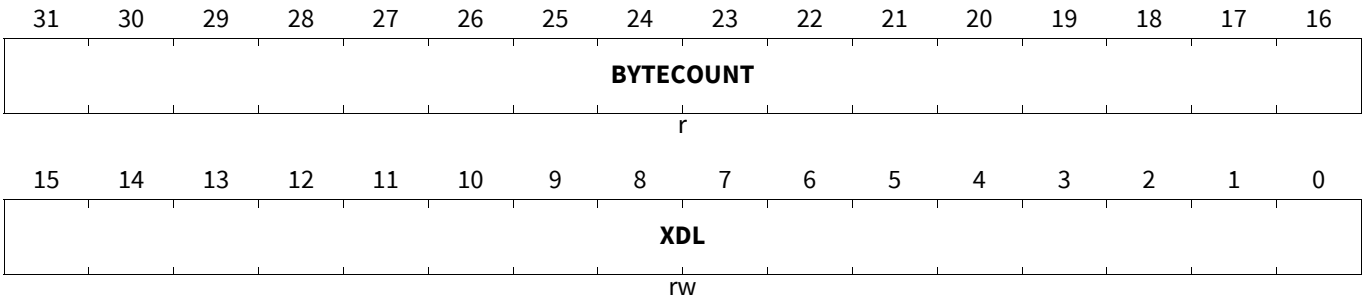
Extra Large Data Configuration Register

The XXLCON register provides counter for sending frames with very long blocks of data by extending the long data mode. It avoids a need for further BACON entries, like those needed in continuous mode.

Data length in this register overrides the **BACON**.DL setting in XXL mode, when **BACON**.BYTE=1 and **BACON**.DL=0. The data is sent to the corresponding slave as defined by **BACON**.CS bit field.

XXLCON

Extra Large Data Configuration Register (058$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
XDL	15:0	rw	**Extended Data Length** Defines the length of the data block in bytes in range of 2 to 65536. Overrides **BACON**.DL when **BACON**.BYTE=1 and **BACON**.DL=0. 0000_H2 bytes 0001_H2 bytes … $FFFF_H$65536 bytes
BYTECOUNT	31:16	r	**Extended Data Length** In the XXL mode, shows the current state of the internal byte down counter (bytes remaining to be sent). In short and long modes, the value of this bit field is don't care. 0000_H0 bytes 0001_H2 bytes … $FFFF_H$65536 bytes

Move Counter Register

In Move Counter Mode, the slave select signal SLSO remains active during the time **MC**.MCOUNT write accesses (for example DMA moves) to the TXFIFO are being shifted out. The **MC** mode is enabled when **MCCON**.MCEN=1

and **BACON**.BYTE = 0 and **BACON**.LAST = 0. After the frame has been completed, SLSO is automatically deactivated. Then, the software should clear the MCEN if the move counter is not needed any more. The move counter is a down-counter that triggers an interrupt (if enabled):

- after the move before last has been shifted out
- after the last move has been shifted out

The range of counting is 8191 to 1. The bit field **MC**.CURRENT shows the currently transmitted frame, the last one being 1. The update of the CURRENT bit field takes place with a couple of cycles delay due to the asynchronous frequency domain of the shift engine. The decrementing is done after shifting the last bit of each data. **MC**.CURRENT= 0 means no data pending, the complete data of the frame has been shifted out.

The enable, flag, set and clear bits for the Interrupt Before Last (IBL) and Interrupt After Last (IAL) are defined in **MCCON** register. The interrupt signals are mapped to the User Interrupt line, but have separate enable bits in the **MCCON** register and do not depend on **BACON**.UINT. Therefore on user interrupt in **MC** mode, the software must also check the **MCCON**.IALF and IBLF flags. All other standard QSPI phase interrupts are triggered as usual. If the data is not fed to the TXFIFO in time, EXPECT phase is activated.

An example sequence of using **MC** mode is: CPU writes the configuration for the frame once at the start in the **BACONENTRY** address, afterwards DMA writes DATA in DATAENTRY. MIXENTRY shall not be used. The TXFIFO interrupts trigger the DMA, which itself counts the moves in parallel to the **MC** register.

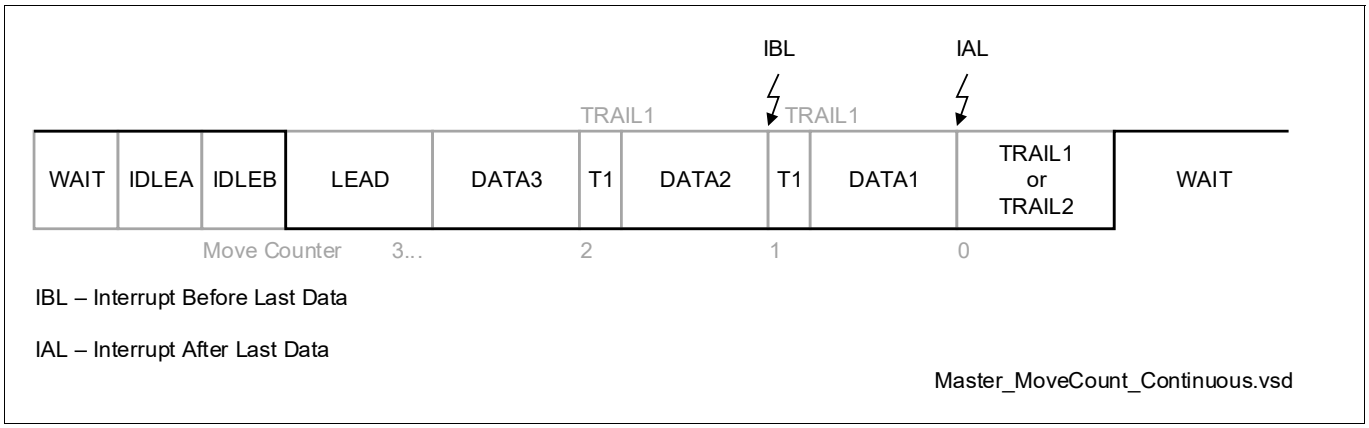


Figure 517 Frame Sequence of Phases in Short Move Counter Continuous Mode

Another usage example of using MC mode is daisy-chaining: Here the CPU writes the number of frames for all slaves to the MC register. Afterwards it writes the configuration for the frame of the fist slave (BACON3 in the figure) in the BACONENTRY address, afterwards DMA writes the for the first slave (DATA3_2 and DATA3_1) in DATAENTRY. MIXENTRY shall not be used. The TXFIFO interrupts trigger the DMA, which until all data for the first slave has been written. Then the configuration of the second slave (BACON2) needs to be written to BACONENTRY, followed by its data (DATA2_3, DATA2_2 and DATA2_1). This process may be repeated until the whole daisy-chain is filled (in the figure this is an additional BACON1 and DATA1_1). The different BACONs may only vary in the BACON.DL and/or BACON.MSB setting.

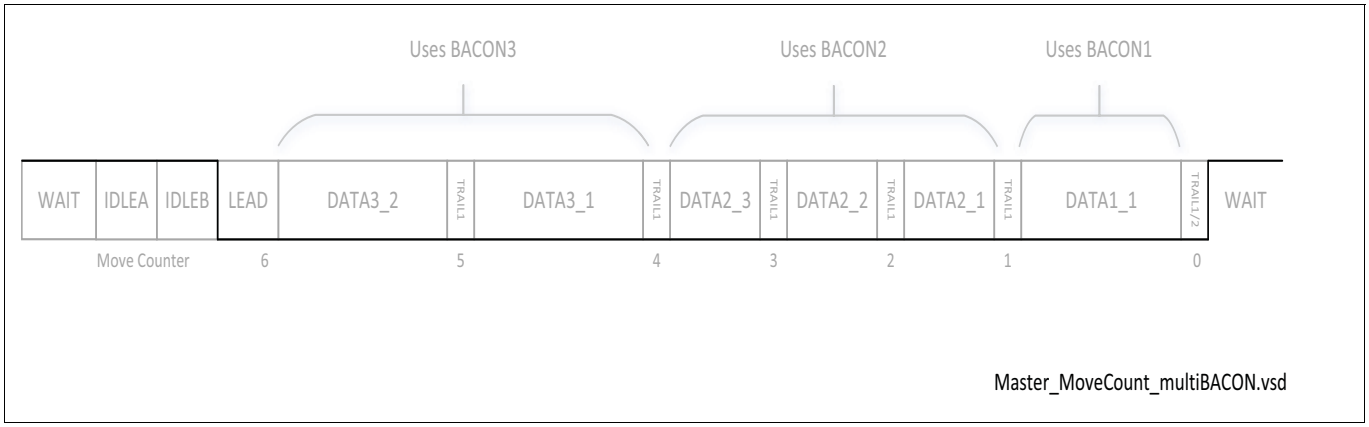


Figure 518 Frame Sequence of Phases in Move Counter implementing daisy-chaining

It is possible to combine Continuous Long Mode with Move Counter Mode so that a frame starts with one or more Continuous Long Mode transmissions and ends with one Move Counter Mode transmission.

MC

Move Counter Register (0A4$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**CURRENT**												
r			rh												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**MCOUNT**												
r			rw												

Field	Bits	Type	Description
MCOUNT	12:0	rw	**Move Count** Defines the number of moves to be performed in short mode, in range of 1 to 8191.
CURRENT	28:16	rh	**Current Status of the Move Counter** Shows the current status of the Move Counter, that is, how many data blocks are to be transmitted until the end of the frame. 0000_H0 0001_H1 0002_H2 … $1FFF_H$8191
0	15:13, 31:29	r	**Reserved** Read as 0; should be written with 0.

Move Counter control Register

Contains the control bits for the Move Counter Mode. See the description **MC** register.

MCCON

Move Counter control Register **(0A8$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCEN	T2EN	0						IALS	IALC	IALF	IALEN	IBLS	IBLC	IBLF	IBLEN
rw	rw	r						w	w	rh	rw	w	w	rh	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										TRAIL2			TPRE2		
r										rw			rw		

Field	Bits	Type	Description
TPRE2	2:0	rw	**Prescaler for the Trailing Delay 2** Trailing delay injected in the configuration register for the last data if T2EN is set. Length in units T_{PER} 000_B 1 001_B 4 010_B 16 … 111_B 16384
TRAIL2	5:3	rw	**Last Trailing Delay** Trailing delay injected in the configuration register for the last data if T2EN is set. Defines the length of the leading delay, in T_{PER} units pre scaled with TPRE 000_B 1 unit 001_B 2 units … 111_B 8 units
IBLEN	16	rw	**Interrupt Before Last Enable** Enable bit for this event. 0_B Disabled 1_B Enable
IBLF	17	rh	**Interrupt Before Last Flag** Flag bit for this event. 0_B No event 1_B Event occurred
IBLC	18	w	**Clear Bit for IBLF** Writing 1 clears the IBLF. Writing 0 has no effect. Returns 0 on read.
IBLS	19	w	**Set Bit for IBLF** Writing 1 sets the IBLF and triggers an interrupt (if enabled). Writing 0 has no effect. Returns 0 on read.
IALEN	20	rw	**Interrupt After Last Enable** Enable bit for this event. 0_B Disabled 1_B Enable

Field	Bits	Type	Description
IALF	21	rh	**Interrupt After Last Flag** Flag bit for this event. 0_B No event 1_B Event occurred
IALC	22	w	**Clear Bit for IALF** Writing 1 clears the IALF. Writing 0 has no effect. Returns 0 on read.
IALS	23	w	**Set Bit for IALF** Writing 1 sets the IALF and triggers an interrupt (if enabled). Writing 0 has no effect. Returns 0 on read.
T2EN	30	rw	**TRAIL 2 Injection Enable** This bit has to be configured before the transmission of the frame starts. If set, a new value for the last trailing delay will be injected for the last data block, as defined with the bit field TRAIL2. If not set, the TRAIL value from the latest **BACON** will be valid also as the last trailing delay.
MCEN	31	rw	**Move Counter Enable** Enables the Move Counter feature. If enabled, the MCOUNT value is taken in consideration, otherwise the standard continuous mode is active.
0	15:6, 29:24	r	**Reserved** Read as 0; should be written with 0.

MIX_ENTRY Register

MIXENTRY
MIX_ENTRY Register **($05C_H$)** **Application Reset Value: $0000\ 0000_H$**

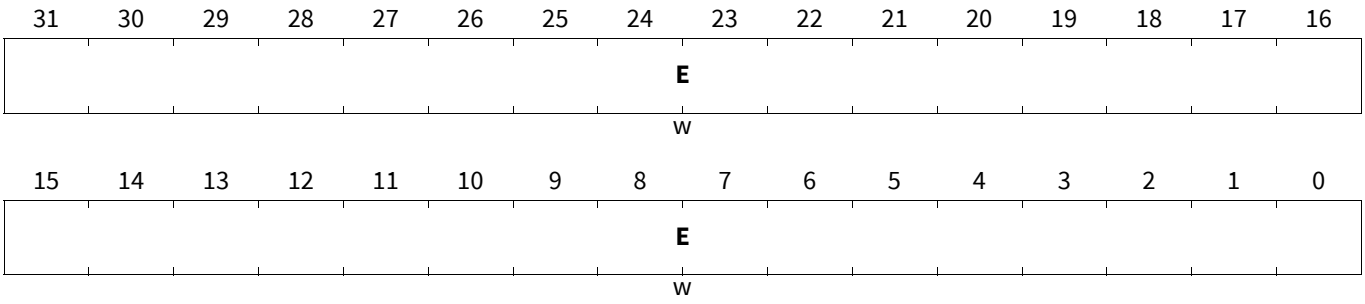
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
E															
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E															
w															

Field	Bits	Type	Description
E	31:0	w	**Entry Point to the TxFIFO**

BACON_ENTRY Register

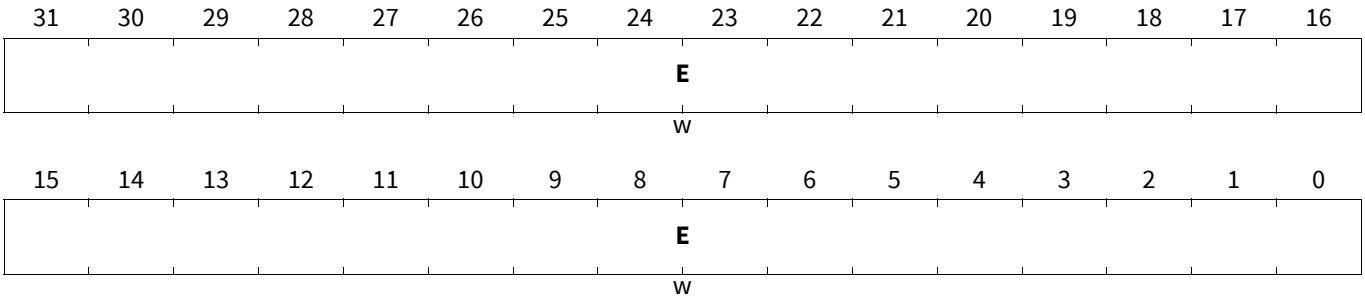
BACONENTRY
BACON_ENTRY Register **(060_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
E	31:0	w	**Entry Point to the TxFIFO**

DATA_ENTRY Register x

DATAENTRYx (x=0-7)
DATA_ENTRY Register x **(064_H+x*4)** **Application Reset Value: 0000 0000_H**



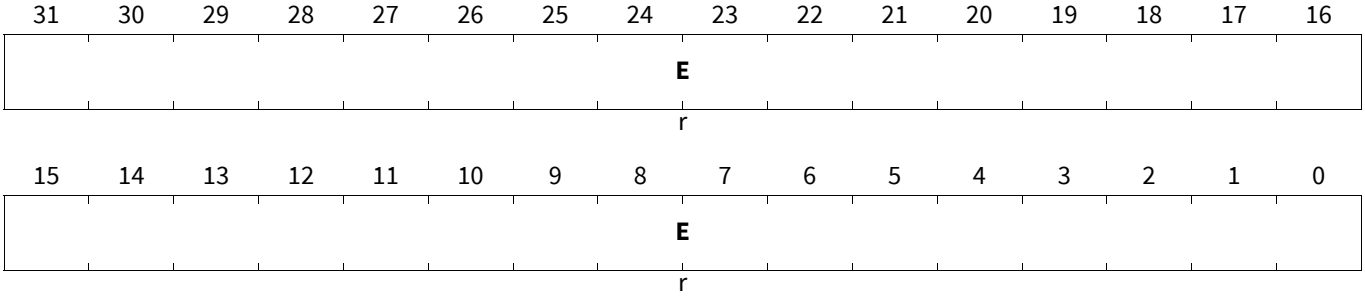
Field	Bits	Type	Description
E	31:0	w	**Entry Point to the TxFIFO**

RX_EXIT Register

The RXFIFO has a property that a read access from an empty RXFIFO generates an underflow interrupt, and delivers only “1” bits, which overrules the reset value. Therefore reading from a non initialized RXFIFO delivers all “1” and not all ”0”.

RXEXIT

RX_EXIT Register **(090$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
E	31:0	r	**Read Point from the RxFIFO**

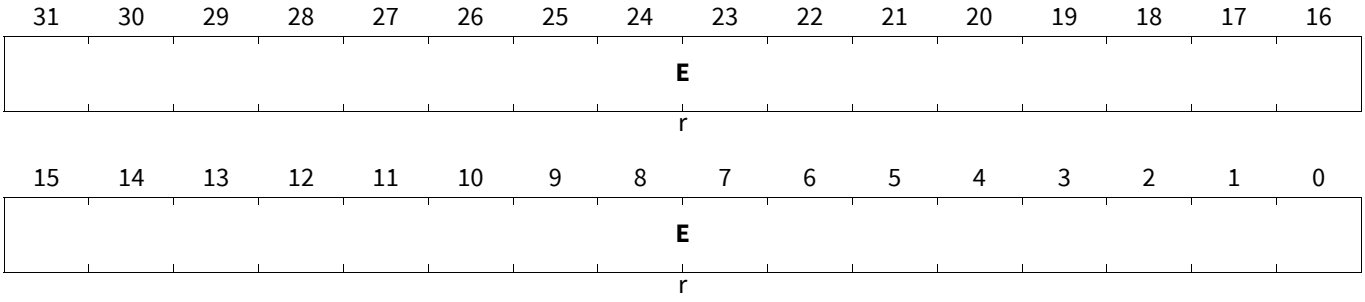
RX_EXIT Debug Register

The register **RXEXITD** provides a non-destructive address, showing the next available value in the RXFIFO.

*Note: This register provides a non-volatile access to the RXFIFO. It delivers the same value as **RXEXIT**, but without affecting read pointer and the filling level of the RXFIFO.*

RXEXITD

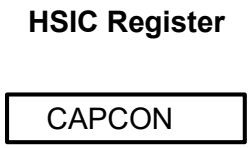
RX_EXIT Debug Register **(094$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
E	31:0	r	**Read Point from the RxFIFO**

37.5.2 HSIC Registers

The HSIC sub-module is optionally added to the QSPI and have one Register.



HSIC-reg

Figure 519 HSIC Registers

Capture Control Register

CAPCON is the only control register for the HSIC sub-module. It contains the configuration, control bits and the captured value of the counter.

CAPCON
Capture Control Register **(0A0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CAPSEL	**CAPF**	**CAPS**	**CAPC**	**0**							**EN**	**INS**		**EDGECON**	
rw	rh	w	w	r							rw	rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVF	**CAP**														
rh	rh														

Field	Bits	Type	Description
CAP	14:0	rh	**Captured Value** Provides the latest capture value from the capture timer.
OVF	15	rh	**Overflow Flag** Signals if an overflow event of the capture timer occurred in the latest measurement. Shows a value of 1 if one or more overflows of the capture timer occurred during the latest measurement. Each capture event refreshes this bit with the new value.
EDGECON	17:16	rw	**Edge Configuration** Configures the capture mode of the counter, defining which edge starts the counting, which edge captures its contents. 00_B RR mode (rising edge to rising edge) 01_B RF mode (rising edge to falling edge) 10_B FR mode (falling edge to rising edge) 11_B FF mode (falling edge to falling edge)

Field	Bits	Type	Description
INS	19:18	rw	**Input Selection** Selects one of four input lines for the capture signal. 00_B INA selected 01_B INB selected 10_B INC selected 11_B IND selected
EN	20	rw	**Enable Bit of the Capture Timer** Enables the capture timer clock. Enabled timer waits for the next start edge as defined in EDGECON, before it starts counting. 0_B Disabled 1_B Enabled
CAPC	28	w	**Capture Flag Clear** Writing 1 to this bit clears the flag CAPF. Writing 0 has no effect. Returns 0 on read.
CAPS	29	w	**Capture Flag Set** Writing 1 to this bit sets the flag CAPF. Writing 0 has no effect. Returns 0 on read.
CAPF	30	rh	**Capture Flag** Indicates if a capture event has occurred since the last clear event by the bit CAPC.
CAPSEL	31	rw	**Capture Interrupt Select Bit** Select if the capture events are routed to the HC interrupt line. 0_B Not routed (capture interrupt disabled) 1_B Routed (capture interrupt enabled)
0	27:21	r	**Reserved**

37.6 IO Interfaces

Table 330 List of QSPI Interface Signals

Interface Signals	I/O	Description
TX_INT	out	**QSPI Transmit Service Request**
RX_INT	out	**QSPI Receive Service Request**
ERR_INT	out	**QSPI Error Service Request**
PT_INT	out	**QSPI Phase Transition Service Request**
U_INT	out	**QSPI User Defined Service Request**
HC_INT	out	**QSPI High Speed Capture Service Request**
HSICINA	in	**Highspeed capture channel**
HSICINB		
HSICINC		
HSICIND		

Table 330 List of QSPI Interface Signals (cont'd)

Interface Signals	I/O	Description
SLSIA	in	**Slave select input**
SLSIB		
SLSIC		
SLSID		
SLSIE		
SLSIF		
SLSIG		
SLSO(15:0)	out	**Master slave select output**
SCLKA	in	**Slave SPI clock inputs**
SCLKB		
SCLKC		
SCLKD		
SCLKE		
SCLKF		
SCLKG		
SCLKH		
SCLK	out	**Master SPI clock output**
SCLKN	out	**Master SPI clock output (LVDS N line)**
SCLKP	out	**Master SPI clock output (LVDS P line)**
MRST	out	**Slave SPI data output**
MRSTA	in	**Master SPI data input**
MRSTB		
MRSTC		
MRSTD		
MRSTE		
MRSTF		
MRSTG		
MRSTH		
MRSTAN	in	**Master SPI data input (LVDS N line)**
MRSTBN		
MRSTCN		
MRSTDN		
MRSTEN		
MRSTFN		
MRSTGN		
MRSTHN		

Table 330 List of QSPI Interface Signals (cont'd)

Interface Signals	I/O	Description
MRSTAP	in	**Master SPI data input (LVDS P line)**
MRSTBP		
MRSTCP		
MRSTDP		
MRSTEP		
MRSTFP		
MRSTGP		
MRSTHP		
MTSRA	in	**Slave SPI data input**
MTSRB		
MTSRC		
MTSRD		
MTSRE		
MTSRF		
MTSRG		
MTSRH		
MTSR	out	**Master SPI data output**
MTSRN	out	**Master SPI data output (LVDS N line)**
MTSRP	out	**Master SPI data output (LVDS P line)**

37.7 Revision History

Table 331 Revision History (including HSIC)

Reference	Change to Previous Version	Comment
V3.0.20		
	No changes.	

38 Micro Second Channel (MSC)

This chapter describes the Micro Second Channel Interface. It contains the following sections:

- Functional description of the MSC kernel (see **Page 3**)
- MSC kernel register descriptions (see **Page 54**)
- Implementation-specific details and registers of the MSC module (port connections and control, interrupt control, address decoding, and clock control, see **Page 43**)

*Note: The MSC kernel register names described in **Section 38.4** are also referenced in theAURIX™ TC3xx PlatformUser's Manual by the module name prefix "MSCx_".*

MSC Applications

The MSC is a serial interface that is especially designed to connect external power devices. The serial data transmission capability minimizes the number of pins required to connect such external power devices. Parallel data information (coming from the timer units) or command information is sent out to the power device via a high- speed synchronous serial data stream (downstream channel). The MSC receives data and status back from the power device via a low-speed asynchronous serial data stream (upstream channel).

Parallel requests from on chip bus masters to a module will be executed sequentially via the on chip bus system. Read-modify-write feature provides an atomic read/write sequence where no other master can access the module in between. Module hardware semaphores are not supported.

Figure 520 shows a typical application in which an MSC interface controls two power devices. Output data is provided by the module.

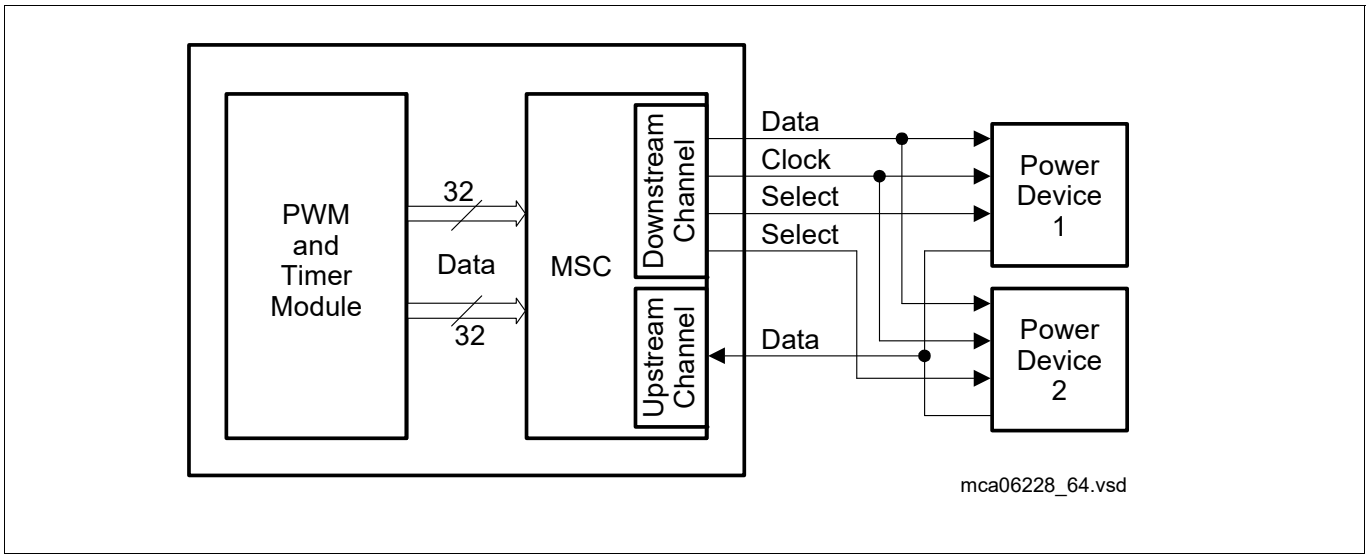


Figure 520 MSC to External Power Device Connection

Some applications are:

- Control of the external power switching unit via the downstream channel
- Receiving information back from power switching unit
- Serial connections to other peripheral devices

38.1 Feature List

- Fast synchronous serial interface to connect power switches in particular, or other peripheral devices via serial buses

- High-speed synchronous serial transmission on downstream channel
  - Serial output clock frequency: $f_{FCL} = f_{MSC}/2$ (f_{MSCmax} = 100 MHz)
  - Fractional clock divider for precise frequency control of serial clock f_{MSC}
  - Command, data, and passive frame types
  - Start of serial frame: Software-controlled, timer-controlled, or free-running
  - Transmission with or without SEL bit
  - Flexible chip select generation indicates status during serial frame transmission
  - Emergency stop without CPU intervention
- Low-speed asynchronous serial reception on upstream channel
  - Baud rate: f_{MSC} divided by 4, 8, 16, 32, 64, 128, or 256 (f_{MSCmax} = 100 MHz)
  - Standard asynchronous serial frames
  - Programmable upstream data frame length (16 or 12 bits)
  - Parity error checker
  - 8-to-1 input multiplexer for SDI lines
  - Built-in spike filter on SDI lines
  - Programmable delay of the receive interrupt after the last stop bit (0 or 1 bit time)
- Selectable pin types of downstream channel interface:
  four LVDS differential output drivers or four digital GPIO pins
- Module reset available
- Advanced features:
  - Asynchronous Baud Rate Adjustment Block
  - one extra interrupt
  - 64-bit data frames extension
  - fully compatible with the 32-bit data frames module version

38.2 Overview

The MSC interface provides a serial communication link typically used to connect power switches or other peripheral devices. The serial communication link includes a fast synchronous downstream channel and a slow asynchronous upstream channel. **Figure 521** shows a global view of the MSC interface signals.

The MSC module features functional extensions, but is backward compatible with the previous microcontroller generations.

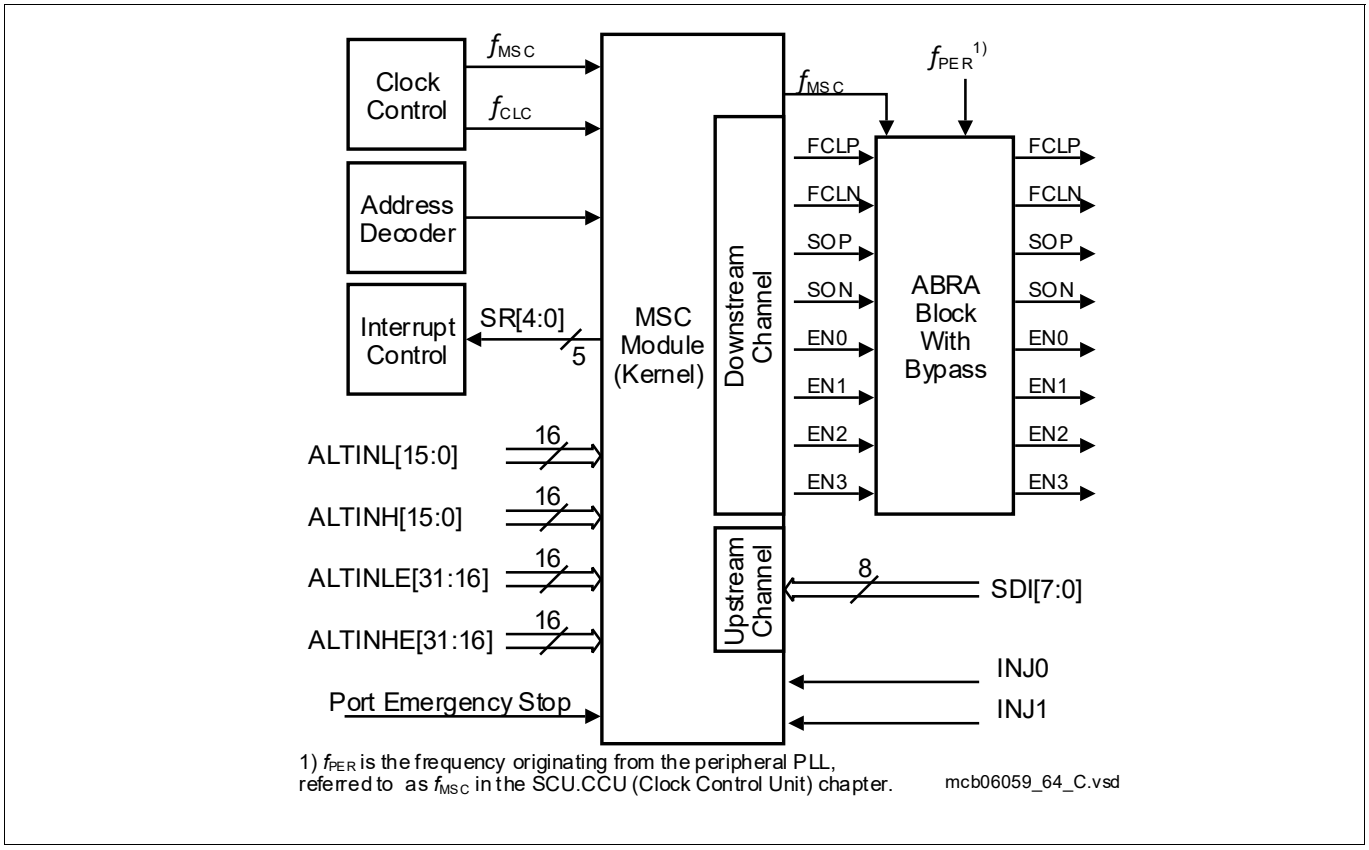


Figure 521 General Block Diagram of the MSC Interface

The downstream and upstream channels of the MSC module communicate with the external world via nine I/O lines. Eight output lines are required for the serial communication of the downstream channel (clock, data, and enable signals). One out of eight input lines SDI[7:0] is used as serial data input signal for the upstream channel. The source of the serial data to be transmitted by the downstream channel can be MSC register contents or data that is provided at the ALTINL/ALTINH input lines. These input lines are typically connected to other on-chip peripheral units (for example with a timer unit like the GTM). An Port Emergency Stop input signal makes it possible to set bits of the serial data stream to dedicated values in emergency case.

Clock control, address decoding, and interrupt service request control are managed outside the MSC module kernel. Service request outputs are able to trigger an interrupt or a DMA request.

38.3 Functional Description

This section describes the MSC kernel, the Command Extension Mode. the Asynchronous Baud Rate Adjustment Block an the module implementation.

38.3.1 MSC Kernel Description

This section describes the functionality of the MSC kernel.

38.3.1.1 Downstream Channel

The downstream channel performs a high-speed synchronous serial transmission of data to external devices. Its 64-bit shift register is divided into two 32-bit parts, SRL and SRH. Each bit of SRL and SRH can be selected to be delivered by the Downstream Data Register DD, by the Downstream Command Register DC, or by two 16-bit wide input signal buses ALTINL and ALTINH. In the 64-bit extended mode, set with the bit **DSCE**.EXEN = 1, the MSC module provides two 16-bit extension buses, ALTINLE and ALTINHE.

Figure 522 is a diagram of the MSC downstream channel.

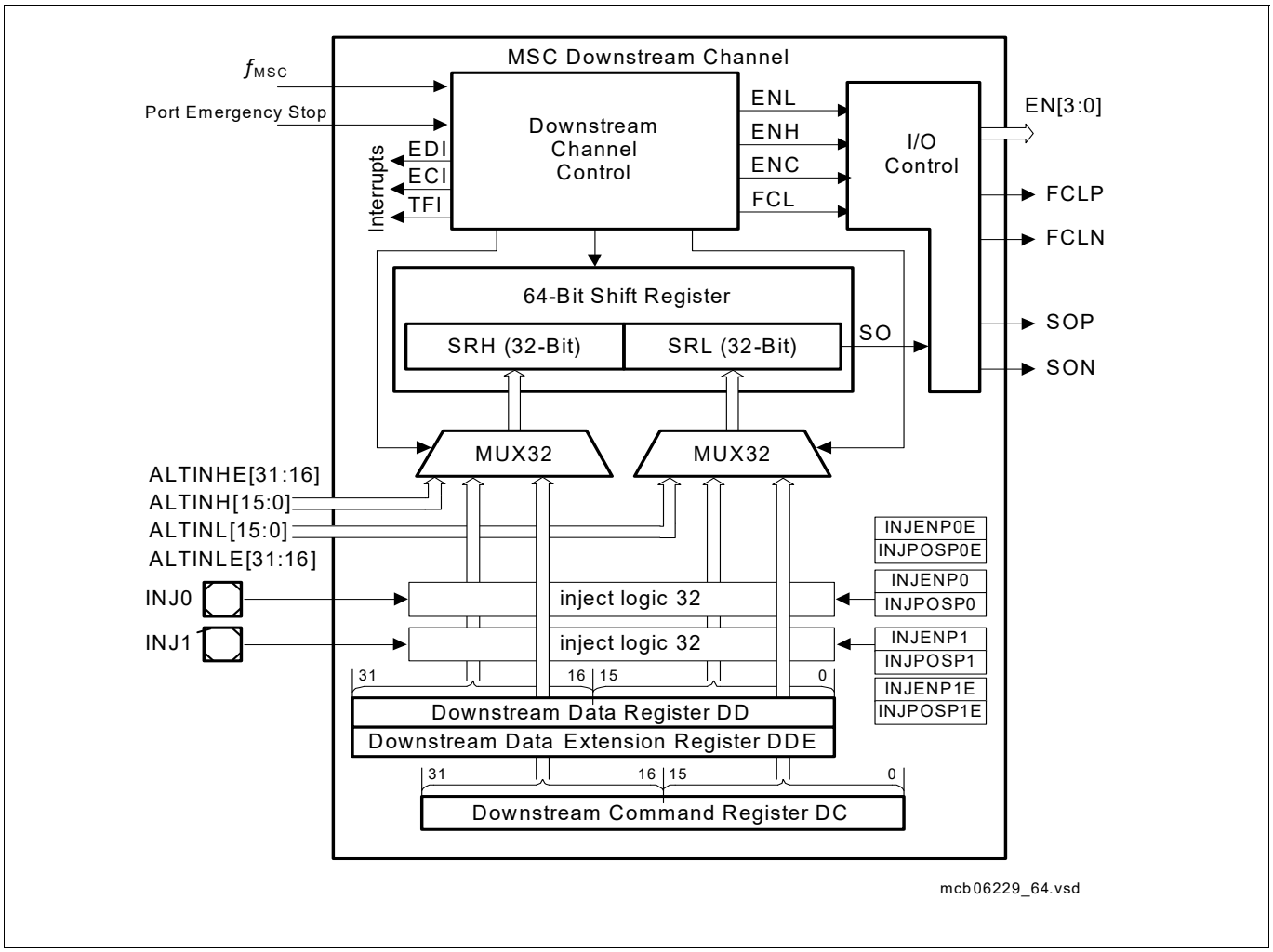


Figure 522 Downstream Channel Block Diagram

The enable signals ENL, ENH, and ENC indicate certain phases of the serial transmission in relation to the serial clock FCL. In the I/O control logic, these signals can be combined to four enable/select outputs EN[3:0]. For supporting differential output drivers, the serial clock output FCL and the serial data output SO are available in both polarities, indicated by the signal name suffix “P” and “N”.

The Port Emergency Stop input is used to indicate an emergency stop condition of a power device. In emergency case, shift register bits can be loaded bit-wise from the downstream data register instead from the ALTINL, ALTINH, ALTINLE and ALTINHE buses.

38.3.1.1.1 Frame Formats and Definitions

This section describes the frame formats and definitions of the MSC.

Basic Definitions

Figure 523 shows the layout and definitions of a downstream frame. A downstream frame is composed of an active phase and a passive phase. During the active phase, data transmission takes place and during the passive phase no data is transmitted at SO. The active phase is split into two parts: The SRL active phase in which the content of the shift register low part SRL is transmitted, and the SRH active phase in which the content of the shift register high part SRH is transmitted. At the beginning of the SRL and SRH active phase, a selection bit (SELL) can be optionally inserted into the serial data stream. In the frame shown in **Figure 523**, SELL is generated at the beginning of the SRL active phase (not for the SRH active phase). The least significant bits of SRL and SRH are sent out first.

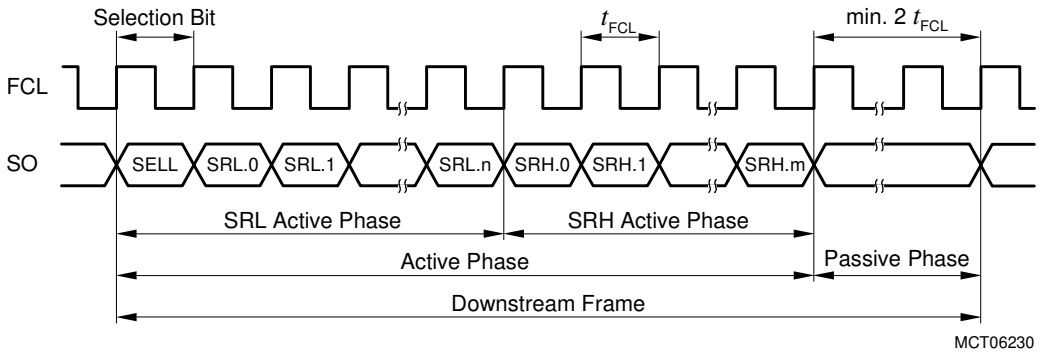


Figure 523 Downstream Channel Frame

The MSC downstream channel uses three types of frame formats for operation:

- Command frames, indicated by SELL = 1
- Data frames, indicated by SELL = 0 or SELL bit insertion disabled
- Passive time frame, indicated by ENL = ENH = 0

Command Frames

A command frame has two active phase parts, SRL active phase and SRH active phase. The command frame always starts with a high-level selection bit, independently whether the selection bit insertion (as defined by bit DSC.ENSELL) is enabled or not. The number of the bits transmitted during SRL and SRH active phases (except the selection bit) is defined by bit field **DSC**.NBC. SRL and SRH are combined to a value up to 32-bit whose length can be selected from 0 up to 32 bits. In other words, whenever bits of SRH are transmitted, they are always preceded by the transmission of the complete SRL content. In the command extension mode, activated with the **DSTE**.CX bit, the commands can be up to 64-bit long (see **CX (Command Extension) Mode**).

During the active phase of a command frame, the enable output signal ENC becomes active. The enable output signals ENL and ENH remain inactive.

The passive phase of a command frame always has a fixed length of 2 x t_{FCL}. The diagram shown in **Figure 524** assumes that the FCL clock is only generated during the active phase of the command frame (OCR.CLKCTRL = 0).

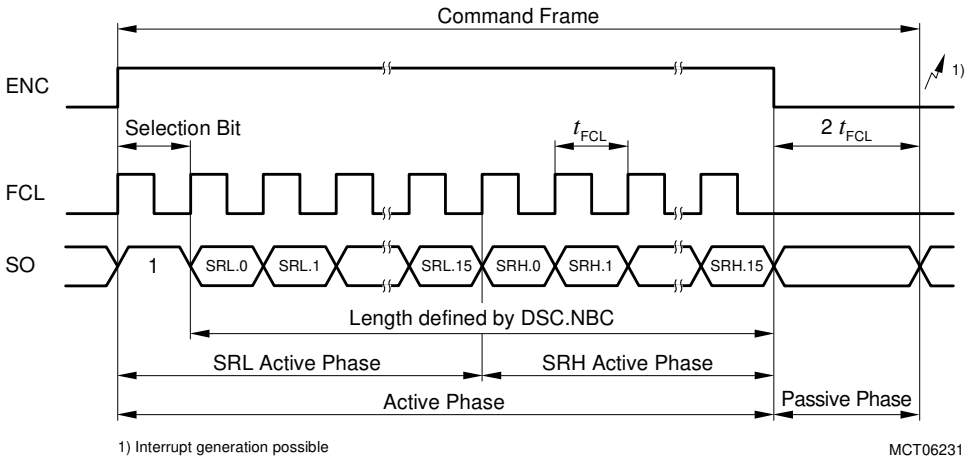


Figure 524 Command Frame Layout

Table 332 shows the programming of the bits to be transmitted and the resulting length of the complete command frame.

Table 332 Command Frame Length

DSC.NBC	SRL/SRH Bits that are Transmitted in Active Phase	Command Frame Length in t_{FCL} Periods
000000_B	No bit shifted out	1 + 0 + 2 = 3
000001_B	SRL[0] shifted out	1 + 1 + 2 = 4
000010_B	SRL[1:0] shifted out	1 + 2 + 2 = 5
000011_B	SRL[2:0] shifted out	1 + 3 + 2 = 6
…	…	…
001111_B	SRL[14:0] shifted out	1 + 15 + 2 = 18
010000_B	SRL[15:0] shifted out	1 + 16 + 2 = 19
010001_B	SRL[15:0] and SRH[0] shifted out	1 + 17 + 2 = 20
010010_B	SRL[15:0] and SRH[1:0] shifted out	1 + 18 + 2 = 21
010011_B	SRL[15:0] and SRH[2:0] shifted out	1 + 19 + 2 = 22
…	…	…
011111_B	SRL[15:0] and SRH[14:0] shifted out	1 + 31 + 2 = 34
100000_B	SRL[15:0] and SRH[15:0] shifted out	1 + 32 + 2 = 35
Other NBC combinations	Reserved; do not use these bit combinations.	

Data Frames

A data frame has two active phase parts, SRL active phase and SRH active phase. The number of bits that are transmitted can be programmed separately for each of these two phases. Bit field DSC.NDBL determines the number of SRL bits that are transmitted during the SRL active phase and DSC.NDBH determines the number of SRH bits that are transmitted during the SRH active phase.

SRL and SRH active phases can start with a low-level selection bit when enabled by bits DSC.ENSELL or DSC.ENSELH.

During the SRL active phase of a data frame, the enable output signal ENL becomes active and during the SRH active phase of a data frame, the enable output signal ENH becomes active. The enable output signal ENC remains inactive.

The length of the data frame's passive phase is variable and is defined by bit field DSC.PPD. It can be within a range of 2 x t_{FCL} up to 31 x t_{FCL}. The diagram shown in **Figure 525** assumes that the FCL clock is only generated during the active phase of the data frame (OCR.CLKCTRL = 0).

Table 333, **Table 334**, and **Table 336** show the definitions of the five data frame parameters that determine the layout of the data frame.

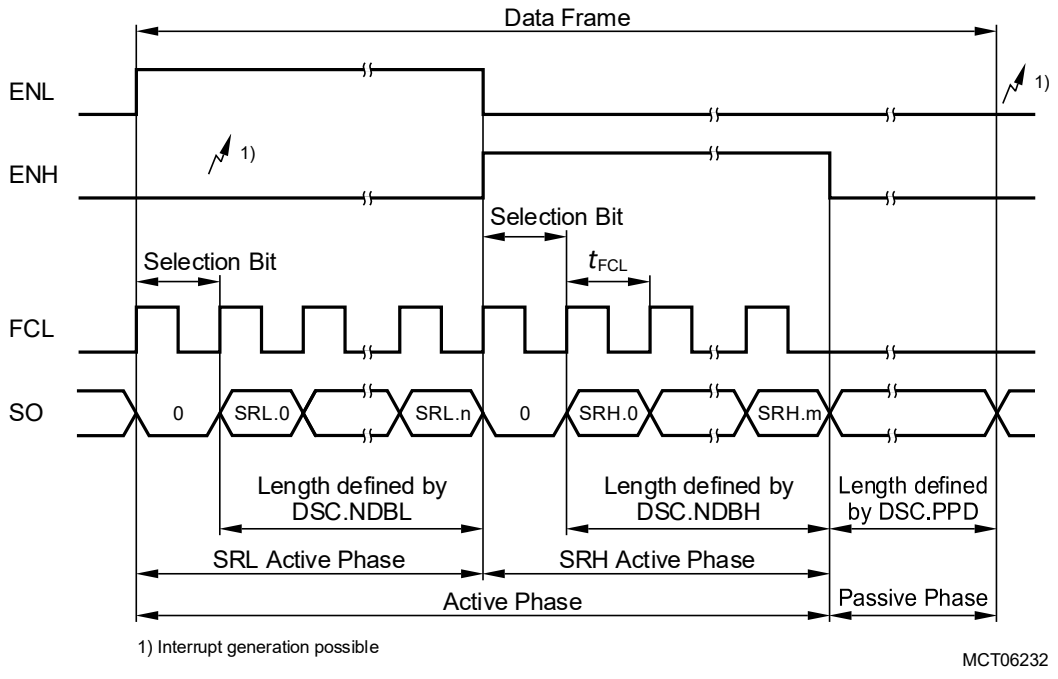


Figure 525 Data Frame Layout

Table 333 Data Frame Selection Bit Parameters

DSC.ENSELL	Selection Bit	DSC.ENSELH	Selection Bit
0	No selection bit inserted at the beginning of the SRL active phase	0	No selection bit inserted at the beginning of the SRH active phase
1	A low level selection bit is inserted at the beginning of the SRL active phase	1	A low level selection bit is inserted at the beginning of the SRH active phase

*Note: The MSC module can operate in two modes: standard (up to 32 data bits) or extended (up to 64 data bits). The mode is selected by using the bit **DSCE**.EXEN.*

Table 334 Data Frame SRL/SRH Length Parameters, Standard Mode

DSCE.NDBLE=0and DSC.NDBL	SRL Bits Transmitted in SRL Active Phase	DSCE.NDBHE=0 and DSC.NDBH	SRH Bits Transmitted in SRH Active Phase
$0\ 00000_B$	No SRL bit transmitted	$0\ 00000_B$	No SRH bit transmitted
$0\ 00001_B$	SRL[0]	$0\ 00001_B$	SRH[0]
$0\ 00010_B$	SRL[1:0]	$0\ 00010_B$	SRH[1:0]
$0\ 00011_B$	SRL[2:0]	$0\ 00011_B$	SRH[2:0]
…	…	…	…
$0\ 01111_B$	SRL[14:0]	$0\ 01111_B$	SRH[14:0]
$0\ 10000_B$	SRL[15:0]	$0\ 10000_B$	SRH[15:0]
Other bit combinations	Reserved; do not use these bit combinations.	Other bit combinations	Reserved; do not use these bit combinations.

Table 335 Data Frame SRL/SRH Length Parameters, Extended Mode

DSCE.NDBLE=1 and DSC.NDBL	SRL Bits Transmitted in SRL Active Phase	DSCE.NDBHE=1 and DSC.NDBH	SRH Bits Transmitted in SRH Active Phase
$1\ 00000_B$	No SRL bit transmitted	$1\ 00000_B$	No SRH bit transmitted
$1\ 00001_B$	SRL[16:0]	$1\ 00001_B$	SRH[16:0]
$1\ 00010_B$	SRL[17:0]	$1\ 00010_B$	SRH[17:0]
$1\ 00011_B$	SRL[18:0]	$1\ 00011_B$	SRH[18:0]
…	…	…	…
$1\ 01111_B$	SRL[30:0]	$1\ 01111_B$	SRH[30:0]
$1\ 10000_B$	SRL[31:0]	$1\ 10000_B$	SRH[31:0]
Other bit combinations	Reserved; do not use these bit combinations.	Other bit combinations	Reserved; do not use these bit combinations.

Table 336 Data Frame Passive Phase Length

DSC.PPD	Passive Phase Length
00000_B	2 x t_{FCL}
00001_B	2 x t_{FCL}

Table 336 Data Frame Passive Phase Length (cont'd)

DSC.PPD	Passive Phase Length
00010_B	2 x t_{FCL}
00011_B	3 x t_{FCL}
…	…
11110_B	30 x t_{FCL}
11111_B	31 x t_{FCL}

The following formula determines the number of t_{FCL} cycles of an up to 32-bit data frame:
All parameters (bits and bit fields) are located in register DSC.

$$Ncycles = ENSELL + NDBL + ENSELH + NDBH + PPD \quad (38.1)$$

Note that in the formula above, PPD must be set to 2 when DSC.PPD ≤ 00010_B.

The following formula determines the number of t_{FCL} cycles of an extended, up to 64-bit data frame, in case DSC.NDBL and DSC.NDBH are not equal to zero:

$$Ncycles = ENSELL + NDBL + NDBLE \cdot EXEN \cdot 16 + ENSELH + NDBH + NDBHE \cdot EXEN \cdot 16 + PPD + PPDE \cdot 32 \quad (38.2)$$

Otherwise, the corresponding factor NDBLE*EXEN*16 or NDBHE*EXEN*16 must be taken as zero (see **Table 335**).

Note that in the formula above, PPD must be set to 2 when DSC.PPD ≤ 00010_B.

The parameters (bits and bit fields) are located in registers DSC and DSCE.

Attention: ***In order to calculate the time between two consecutive TRPs (Time Reference Point), always use ENSELL=1 in the formulas above. ENSELL does not influence the time between two consecutive TRPs, but only the length of a data frame. The difference in the starting point of a frame with ENSELL=0 and 1 is shown in the Figure 531 and Figure 532.***

Passive Time Frames

A passive time frame has the length defined by the five data frame parameters according **Equation (38.1)**. They are generated only in Data Repetition Mode. Under special conditions (command frame insertion), passive time frames can be shortened (see **Figure 529**).

During passive time frames, the data output SO has to be considered as invalid at the receiving device and the clock output FCL may toggle or not (as selected by bit OCR.CLKCTRL). The ENL and ENH enable signals remain at low level during a passive time frame.

38.3.1.1.2 Shift Register Operation

This section describes the SRL and SRH shift register loading.

SRL Shift Register Loading

During the SRL/SRH shift register load operation at the beginning of each downstream frame transmission, several parameters determine which information is loaded into the bits of the shift register. **Figure 526** shows the logic that is implemented for the SRL shift register loading operation. The logic for the SRH shift register loading operation is equivalent to the one for the SRL register. Its differences in data sources and register controls are described later in this section.

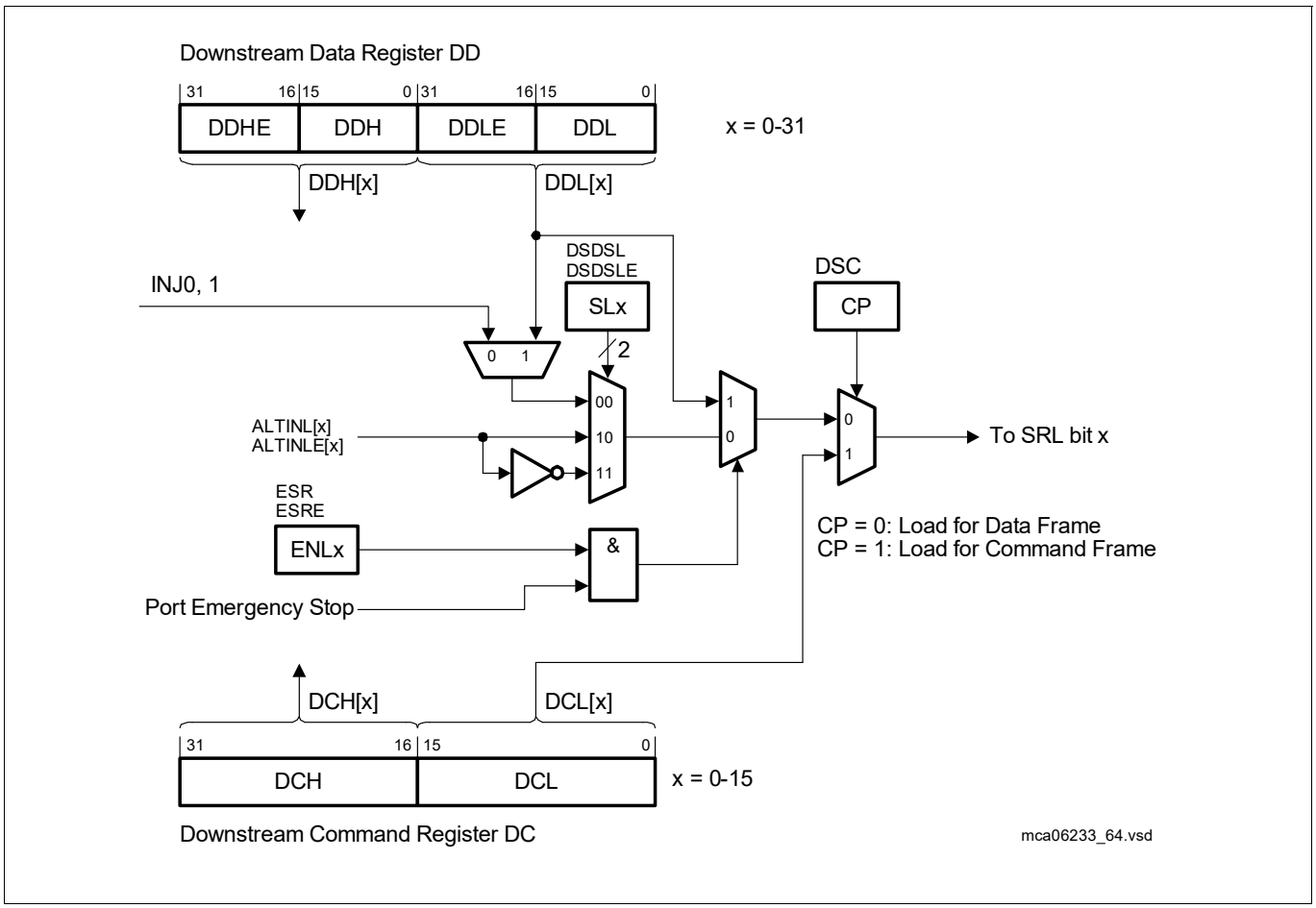


Figure 526 SRL Shift Register Data Loading Control

Four data sources can be selected for each SRL bit by using several control bits and one control signal:

- ALTINL, ALTINLE input line (non-inverted)
- ALTINL, ALTINLE input line (inverted)
- Bit of DD.DDL, DDLE (downstream data register)
- Bit of DC.DCL (downstream control register)

When SRL is loaded for data frame transmission (DSC.CP = 0), bit fields DSDSL.SLx determine bit-wise which data is loaded into SRL bit x. The data source selection as controlled by DSDSL.SLx will only be effective when Port Emergency Stop is inactive (at low level). When Port Emergency Stop = 1 (active) during the load operation, all SRL[x] bits that are enabled for the emergency stop feature (bit ESR.ENLx = 1) are loaded directly with the corresponding bit DDL[x] of the downstream data register DD.

When SRL is loaded for command frame transmission (DSC.CP = 1), always the lower 16-bit part DCL of the downstream control register is loaded completely into SRL.

Table 337 summarizes all SRL data source selection capabilities (x = 0-31):

- in standard mode, **DSCE**.EXEN = 0, the selection configuration of up to 16 bits (x = 0-15) is taken into account
- in extended mode, **DSCE**.EXEN = 1, the selection configuration of up to 32 bits (x = 0-31) is taken into account

Table 337 SRL Data Source Selection Capabilities

DSC. CP	DSDSL.SLx DSDSLE.SLx	ESR. ENLx	Port Emergency Stop	Selection
0	00_B	0	–	Bit DD.DDL[x] is loaded into SRL[x].
	01_B			Reserved.
	10_B			State of ALTINL[x] input is loaded into SRL[x].
	11_B			Inverted state of ALTINL[x] input is loaded into SRL[x].
	XX_B	1	1	Bit DD.DDL[x] is loaded into SRL[x].
1	XX_B	X	X	Bit fields DCL and DCH are completely loaded into SRL and SRH, respectively.

*Note: The data signals can be overruled by injected pin signals INJ0 and INJ1. See **DSCE** register.*

SRH Shift Register Loading

The SRH shift register load operation is equivalent to the SRL shift register load operation. The following differences must be taken into account for SRH shift register loading:

- Input lines ALTINH are connected instead of ALTINL input lines.
- DSDSH register bits control data source selection instead of DSDSL register.
- Emergency stop is enabled by ESR.ENHx bits instead of ESR.ENLx bits.
- Bits of the downstream data register high part DDH are selected instead of DDL.
- Downstream control register high part DCH is selected instead of DCL.

38.3.1.1.3 External Signal Injection

Input signals from two pins can be independently injected at two data bit positions in a data frame. The injection takes place both in normal and in extended mode. The external signals are always injected at the defined position, and if the data frame is shorter than this position, than these signals remain unused. For example, if **DSCE**.INJPOS0 defines injection at the position 14 of the SRL, and the **DSC**.NBDL=12, then all the data bits from position 12 and up remain unused, including the injected signal.

In normal mode the bits are counted from 0 to 31 from LSB to MSB, and in extended mode the bits are counted from 0 to 63 from LSB to MSB, see **Figure 527**. The injection position is always counted as if the frame has the maximal length. In the first step of the preparation for a transmission the injection takes place, and then in the second step, if some fields are configured to be shorter or omitted (for example DDLE or DDH or any other), the unused bits are discarded and only the remaining valid bits are shifted out.

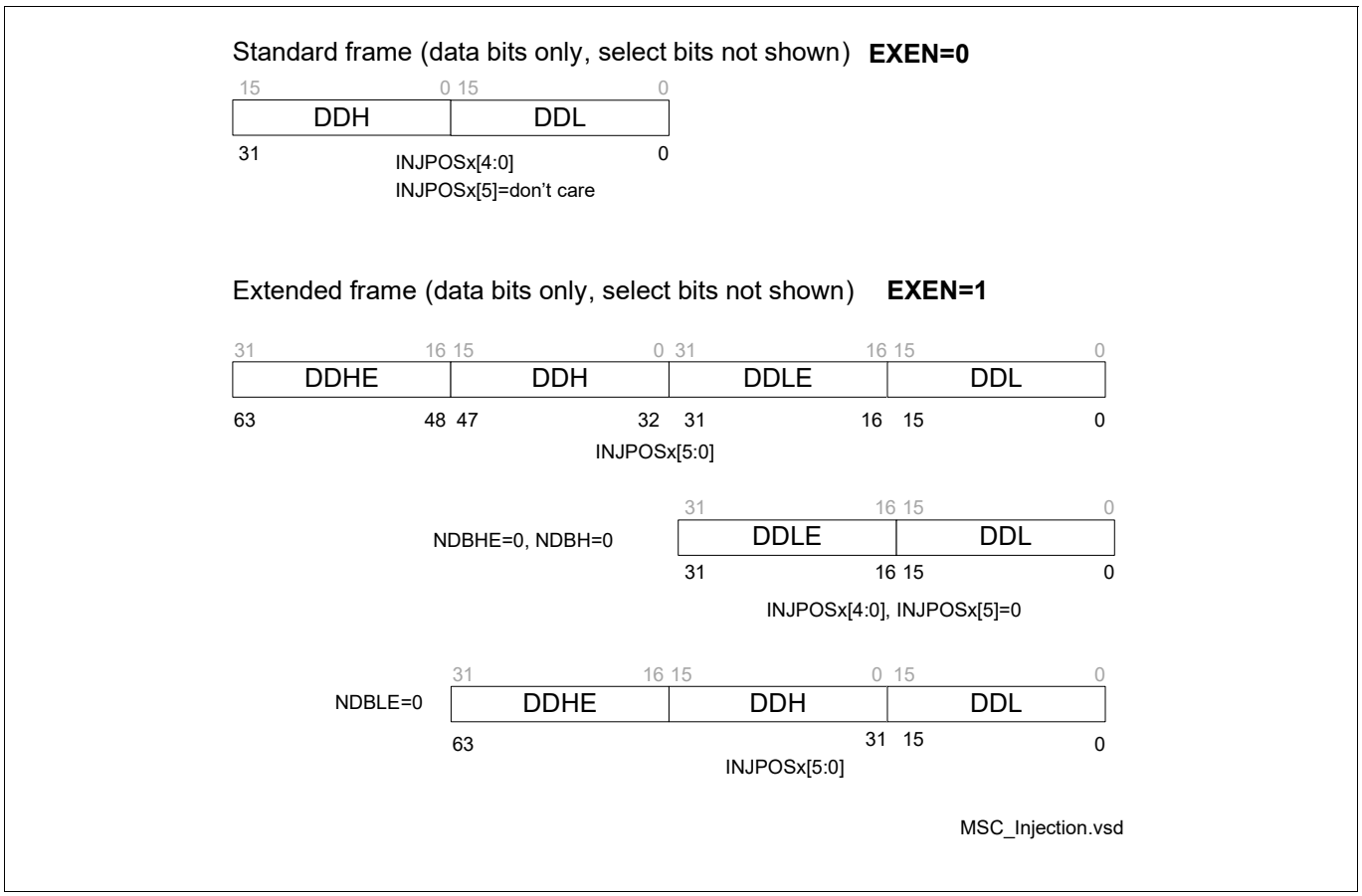


Figure 527 External Signal Injection

38.3.1.1.4 Transmission Modes

The downstream channel of the MSC makes it possible to select between two transmission modes:

- Triggered Mode, selected by DSC.TM = 0, or
- Data Repetition Mode, selected by DSC.TM = 1

Triggered Mode

In Triggered Mode, command frames or data frames are sent out as a result of a software event. When a frame transmission has been finished and no further frame transmission has been requested, the downstream channel returns to idle state and waits for the next frame transmission to be triggered by software.

Setting the DSC.TM bit triggers immediately a frame transmission using the values existing at that moment.

When the Downstream Command Register DC is written, the command pending bit DSC.CP becomes set and a command frame will be immediately started and sent out if the downstream channel is idle. If a data or command frame is currently processed and output, the command frame transmission is delayed, and started when the active downstream frame has been finished. The command pending bit DSC.CP becomes cleared by hardware.

If the downstream channel is idle and the data pending bit DSC.DP is set by writing bit ISC.SDP with 1, a data frame will be immediately started and sent out if the downstream channel is idle. If a data frame or a command frame is currently processed and output, the data frame transmission is delayed and started when the active downstream frame has been finished. The data pending bit DSC.DP becomes cleared by hardware.

A command frame always has priority over the data frame. This means that if both frame pending bits are set (DSC.DP = DSC.CP = 1), the command frame will always be sent first. Therefore, a pending data frame transmission will be delayed as long as no further command frame transmission is running or requested.

Figure 528 is a flow diagram of the Triggered Mode. This diagram especially shows the behavior of the data and command pending bits DSC.DP and DSC.CP. If both frame pending bits are set (DSC.DP = DSC.CP = 1), the command frame will always be sent first, followed by the data frame (assuming no further command frame has been requested).

The type of the active frame that is currently processed and output is indicated by two status flags: DSS.DFA is set during a data frame transmission and DSS.CFA is set during a command frame transmission. Further, the downstream counter DSS.DC indicates the number of shift clock periods that have been elapsed since the start of the current frame.

In Triggered Mode, the shift register loading event as described in **Section 38.3.1.1.2** occurs just before a command or data frame transmission is started.

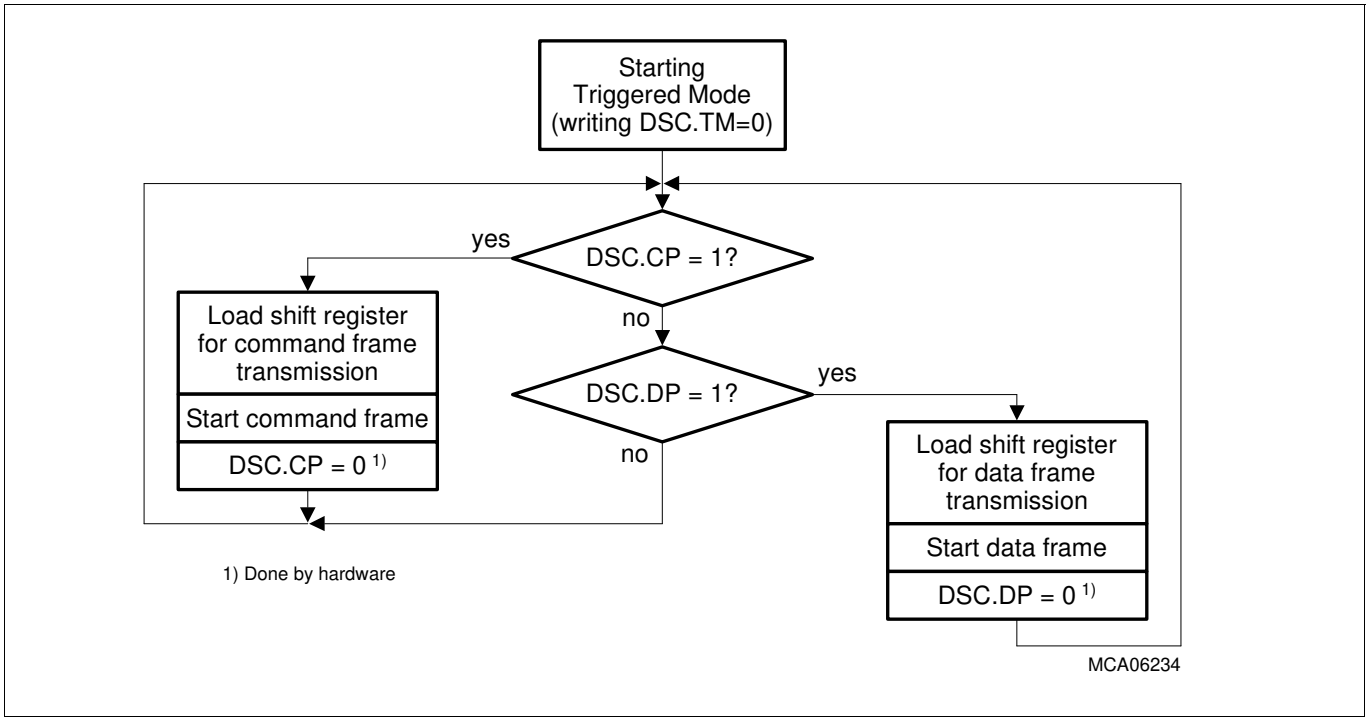


Figure 528 Triggered Mode Flow Diagram

Data Repetition Mode

In Data Repetition Mode, data frames are sent out continuously without any software interaction. In the time gap between two consecutive data frames, passive time frames can be inserted. The number of passive time frames to be inserted (0 to 15) is defined by bit field DSS.NPTF. The duration of data frame (t_{DF}) and passive time frames (t_{PTF}) is determined by the data frame parameters as described in **Equation (38.1)** and **Equation (38.2)**. These parameters determine the time reference points (TRP) at which a data, command or passive time frame is started (see diagram A in **Figure 529** and the **Note on Page 10**).

The automatic data frame generation is controlled by the data pending bit DSC.DP. This bit is set at the end of the last passive time frame if a command frame is running or starts at this TRP, overruling a data frame. DSC.DP is cleared by hardware when the pending data frame starts with transmission at the next TRP. Data Frames are always aligned to time reference points. This means they always start at a TRP. Passive time frames can be shortened. This is especially the case when command frames are inserted.

Continuous data frame transmission can be interrupted by insertion of command frames. Command frames are initiated by software. When the downstream command register DC is written, the command pending bit DSC.CP is set by hardware. CP = 1 indicates that the MSC starts a command frame at the next TRP, independently of whether a data frame (indicated by DSC.DP = 1) or passive time frame should be started with the next TRP. This means also that command frames are always aligned to time reference points.

The bit field **DSCE**.CDCM provides an option for automatic insertion of data frame between two consecutive command frames.

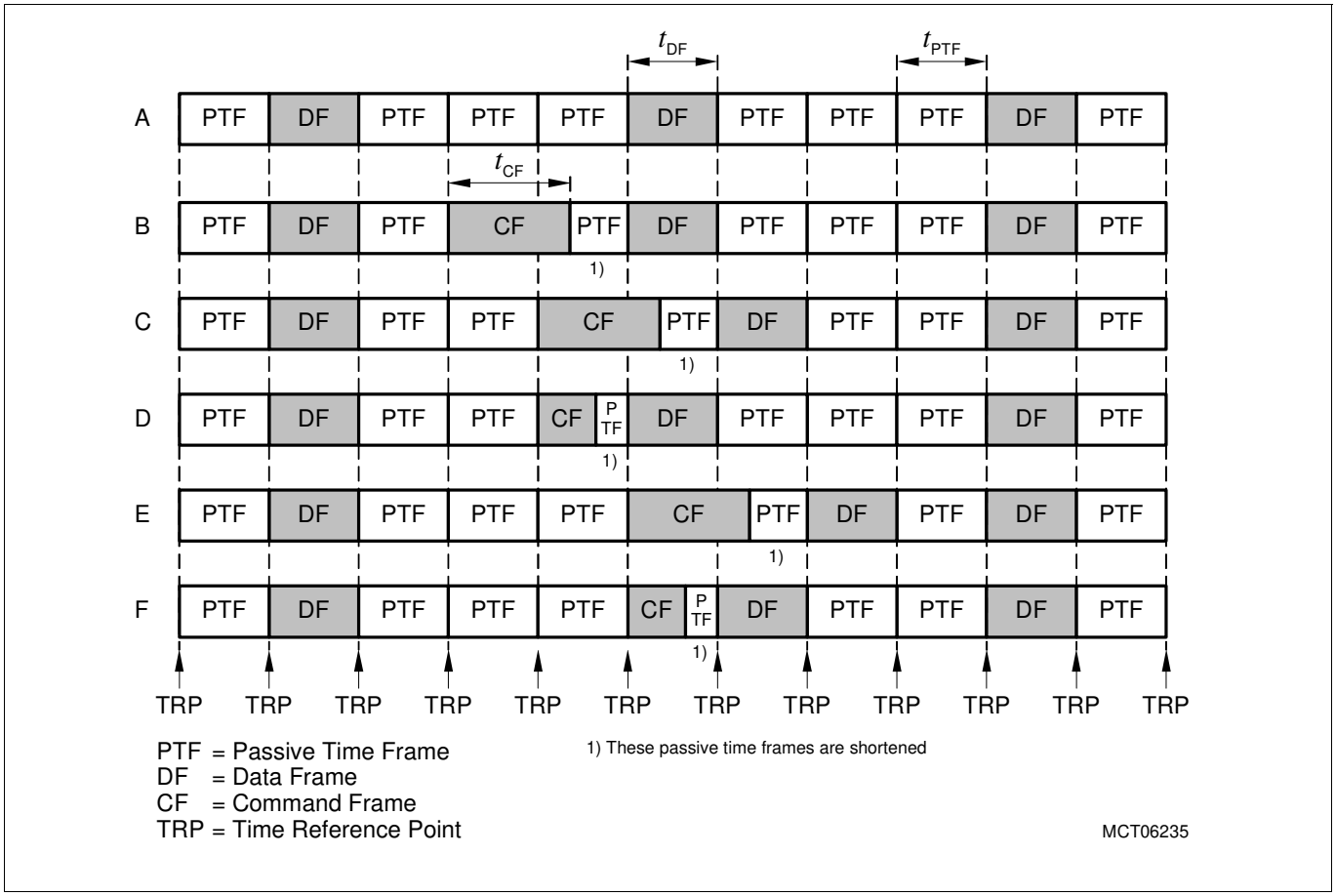


Figure 529 Data Repetition Mode Frame Examples with DSS.NPTF = 0011_B

Diagrams B to F in **Figure 529** show the command frame insertion in Data Repetition Mode.

In diagram B, a command frame has been requested during the first passive time frame after the data frame, and is inserted at the next TRP. In diagrams C and D, a command frame has been requested during the second passive time frame, and is inserted at the time reference point of the last nominal passive time frame.

When the command frame and data frame is not of the same length (this is the case in diagram B to F), a shortened passive time frame is inserted until the next TRP is reached. This ensures that the next data or normal passive time frame is again aligned to a TRP.

When the last passive frame is transmitted and in case of a conflict with command frame, DSC.DP becomes set by hardware. This triggers the start of a data frame when the next TRP is reached.

The type of the active frame (data or command frame) that is currently processed and output is indicated by two status flags: DSS.DFA is set during a data frame transmission and DSS.CFA is set during a command frame transmission. Further, the downstream counter DSS.DC indicates the number of shift clock periods that have been elapsed since the start of the current data, command, or passive time frame.

As in Triggered Mode, the shift register loading event as described in **Section 38.3.1.1.2** occurs in Data Repetition Mode just before a TRP, this means shortly before a command or data frame transmission is started.

Passive Frame Counter in Data Repetition Mode

In Data Repetition Mode, a passive time frame counter DSS.PFC indicates how many time frames have been already transmitted after the last regular data frame occurrence. The passive time frame counter counts up from 0000_B to the value which has been written into bit field DSS.NPTF (number of passive time frames). DSS.PFC = 0000_B indicates that a data frame is requested for transmission.

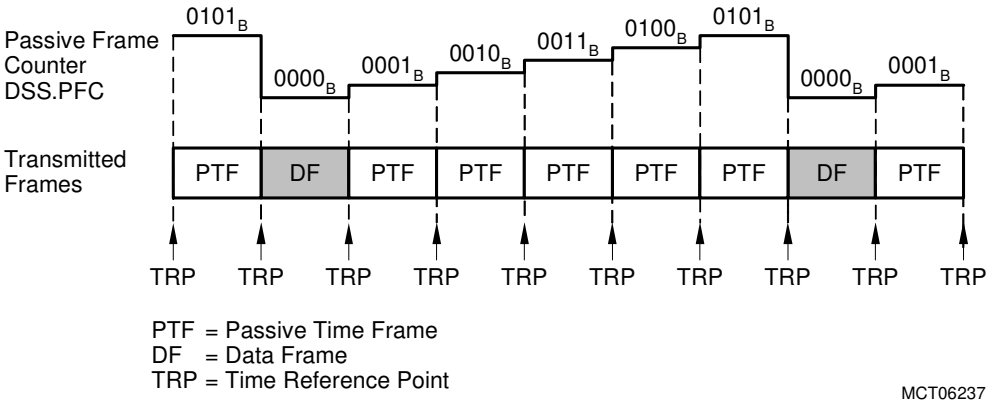


Figure 530 Passive Frame Counter Operation (with DSS.NPTF = 0101_B)

38.3.1.1.5 Downstream Counter and Enable Signals

During downstream channel operation, a 8-bit downstream counter DSS.DC is counting FCL shift clock periods. With the loading of the shift register, the downstream counter is reset to 00_H and started for counting up to the end of the downstream frame (end of passive phase).

In Triggered Mode, the downstream counter stops counting at the end of the passive phase and waits until a new downstream frame is started.

In Repetition Mode, the downstream counter does not stop at the end of the passive phase but is reset and starts counting up again with the next frame, independently whether a data frame, command frame, or passive time frame is started as next frame.

Figure 531 shows an example of downstream channel data frame transmission. In this example, the selection bit for the SRL active frame is enabled (ENSELL = 1), and the selection bit for the SRH active frame is disabled (ENSELH = 0). With loading of the shift register SRL/SRH, the downstream counter is reset and then starts counting up with each FCL clock until the end of the passive phase. ENL is set to high level at the beginning of the SRL active frame selection bit.

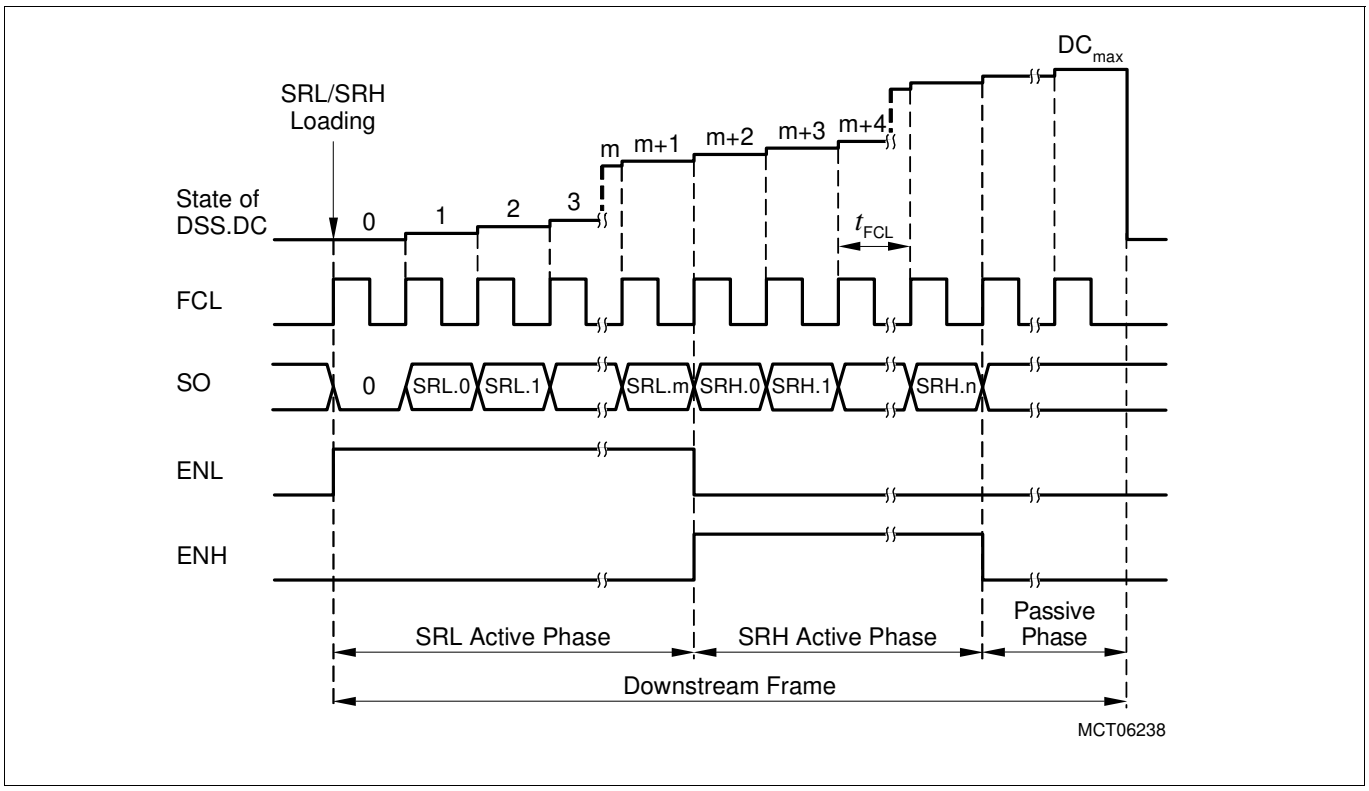


Figure 531 Shift Clock Counting: Data Frame with ENSELL = 1 and ENSELH = 0

When the selection bit for the SRL active frame is disabled (ENSELL = 0, see **Figure 532**), the loading of the shift register SRL/SRH (and reset of the downstream counter) occurs one FCL clock cycle before the first data bit SRL.0 is output. ENL is set to high level with the beginning of the first data bit SRL.0.

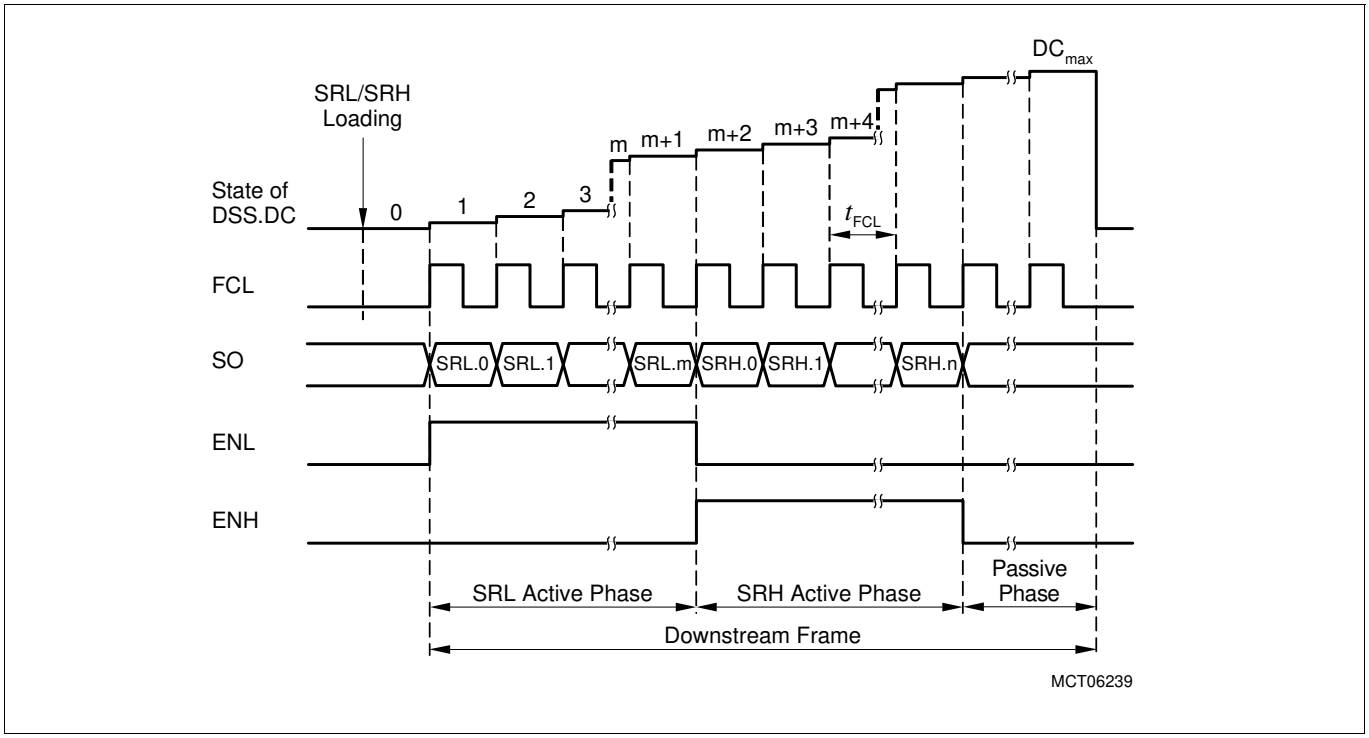


Figure 532 Shift Clock Counting: Data Frame with ENSELL = 0 and ENSELH = 0

38.3.1.1.6 Baud Rate

The baud rate of the downstream channel's serial transmission is defined by the frequency of the serial clock FCL, and is always $f_{MSC}/2$. The f_{MSC} generation is device specific and depends on the implementation of the MSC module. The clock generation is described on **Page 49**.

38.3.1.1.7 Abort of Frames

Only a reset condition of the device can abort a current transmission. The MSC module does not start a new frame transmission when the downstream channel becomes disabled, the suspend mode is requested, or the sleep mode is entered. If one of these three conditions becomes active during a running frame transmission, the frame transmission is completely finished before the requested abort state is entered. Note that in this case no Time Frame Finished Interrupt is generated any more.

38.3.1.2 Upstream Channel

The MSC upstream channel is an asynchronous serial receiver based on the standard asynchronous data transfer protocol. It is dedicated to receive a serial data stream from a peripheral device via its serial data input SDI, using two specific data frame formats.

Figure 533 is a block diagram of the MSC upstream channel.

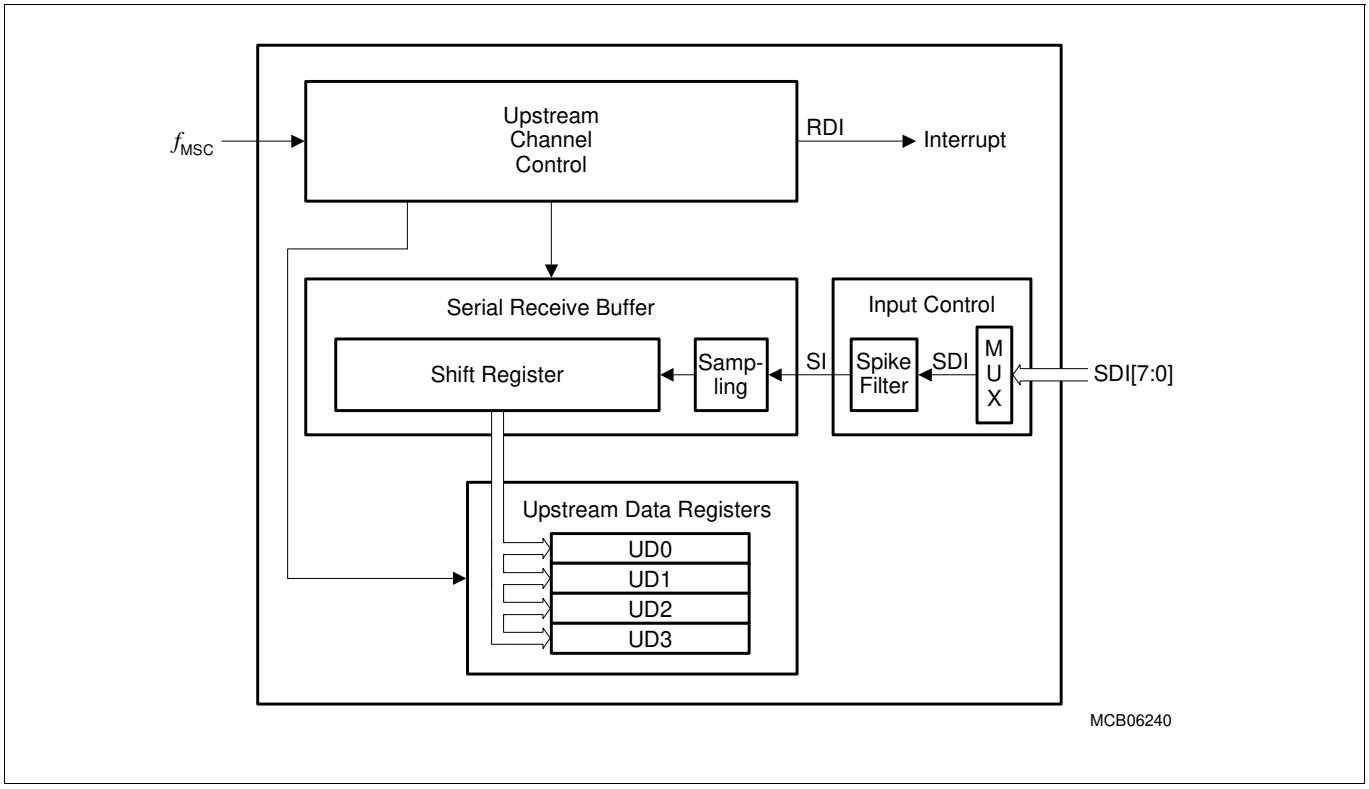


Figure 533 Upstream Channel Block Diagram

The incoming data at SI is sampled after it has been filtered for spikes. The detected logic states of the serial input are clocked into a shift register. After the complete reception of the serial data frame, the content of the shift register is transferred into one of the four data registers, and an interrupt can be generated optionally.

The reception baud rate is directly coupled to the module clock f_{MSC}, and can be within a range of $f_{MSC}/4$ up to $f_{MSC}/256$.

Note: If the ABRA block is used, fractional divider with a fractional ratio generates the upstream sampling rate, and in order to compensate the additional fractional divider jitter, the reception baud rate must be less than $f_{MSC}/10$, or $f_{MSC}/(8 \text{ to} 256)^* n/1024 < f_{MSC}/10$.

38.3.1.2.1 Data Frames

The asynchronous data frames used by the upstream channel include four basic parts:

1. One start bit, always at low level
2. An 8-bit data field D[7:0] with LSB first
3. An optional 4-bit address field A[3:0] with LSB first
4. One parity bit and two stop bits, that are always at high level

The receive interrupt is generated after the second stop bit, but its generation can be delayed by one bit time by setting the bit **USR**.SRDC.

As shown in **Figure 534**, the 16-bit upstream data frame includes an additional 4-bit address field. The upstream frame type is selected by bit USR.UFT.

- USR.UFT = 0: 12-bit upstream data frame selected
- USR.UFT = 1: 16-bit upstream data frame with 4-bit address field selected

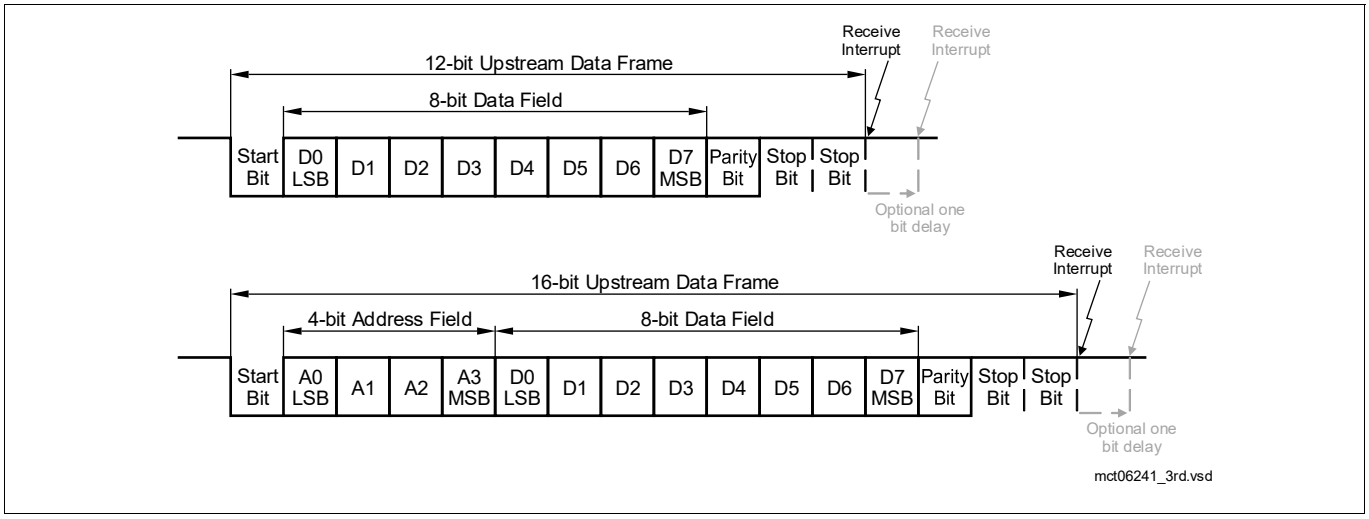


Figure 534 Upstream Channel Frame Types

38.3.1.2.2 Parity Checking

The incoming parity bit of the data frames can be checked by the upstream channel. When a parity error is detected, the parity error flag PERR in the related Upstream Data Register UDx is set. Note that a setting of the parity error flag PERR does not generate an interrupt. The PERR bits must be checked by software. The UDx registers also store the parity bit of the incoming data frame (UDx.P) and the parity bit that is generated internally (UDx.IPF).

Bit USR.PCTR determines the parity mode, even or odd, that is selected for parity checking. With USR.PCTR = 0, even parity mode is selected. Even parity means that the parity bit is set on an odd number of 1s in the data field (12-bit upstream data frame) or in the address plus data field (16-bit upstream data frame). With USR.PCTR = 1, odd parity mode is selected. In odd parity mode, the parity bit is set on an even number of 1s of the related data.

The parity checking logic in the upstream channel also controls whether start bit and the two stop bits of the upstream data frame are at correct logic level. If the start bit is not at low level and the two stop bits are not at high level at the end of the frame reception, the parity error flag UDx.PERR is set, too.

38.3.1.2.3 Data Reception

The reception of the upstream frame is started with a falling edge (1-to-0 transition) on the SI line. When the start bit is detected, serial reception is enabled and the receive circuit begins to sample the incoming serial data and

to buffer it in the receive buffer. After the second stop bit has been detected, the content of the receive buffer is transferred to one of four upstream data registers UDx. The receive circuit then waits for the next start bit (1-to-0 transition) at the SI line. When the content of the receive buffer has been transferred to UDx, the valid bit UDx.V is set by hardware, and a receive interrupt can be generated, independent if PERR occurred or not.

The data reception mechanism does not involve destructive read mechanisms, that is reading the UDx registers does not change any bits automatically.

*Note: The SI input line is the filtered non-inverted (OCR.ILP = 0) or inverted (OCR.ILP = 1) SDI input signal. The SI input signal selection is described on **Page 28**).*

Frame Reception with Address Field

Frame reception for a 16-bit data frame (see **Figure 535**) is selected by USR.UFT = 1. When the content of the receive buffer has been received completely, it is transferred to one of the four UDx registers. The two most significant address bits A[3:2] of the received 4-bit address field select the number x of register UDx in which the received frame content is stored. Register UDx is loaded with the two least significant address bits A0 and A1 (UDx.LABF), the 8-bit data (UDx.DATA), the received parity bit (UDx.P), the calculated parity bit (UDx.IPF), and the parity checking result (UDx.PERR). Finally, the valid bit UDx.V is set to indicate that the UDx register contains valid data.

The current state of the frame reception is indicated by the content of an upstream counter that is readable via bit field USR.UC. The upstream counter is a 5-bit counter that counts the upstream frame bits during reception. As shown in **Figure 535**, the upstream counter is loaded with 10000_B at the detection of a start bit. It counts down and is again at 00000_B when the second stop bit has been detected and the frame reception is finished.

The state of the serial input data line SI is sampled in the middle of a bit cell and shifted into the receive buffer at the end of the bit cell. The frequency of the shift clock f_{BR} depends on the selected baud rate (see **Page 24**).

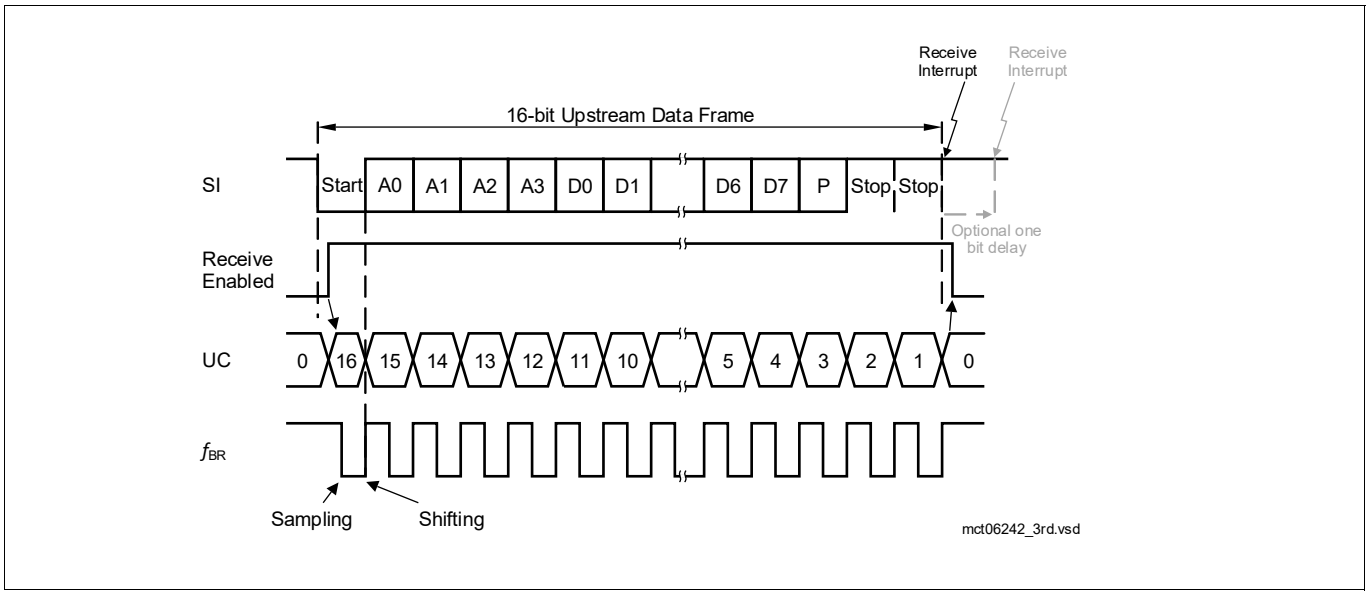


Figure 535 16-bit Upstream Reception

Data Reception without Address Field

Frame reception for a 12-bit data frame is selected by USR.UFT = 0. The reception scheme is comparable with that of the 16-bit data frame reception but there are a few differences:

- The upstream counter is initially loaded with 01100_B.
- The received frame content is always stored in register UD0.

- Bit field UD0.LABF is always loaded with 00_B when the frame is stored.

38.3.1.2.4 Baud Rate

The baud rate of the upstream channel is derived from the MSC module clock f_{MSC}. **Figure 536** shows the configuration of the upstream channel clock circuitry.

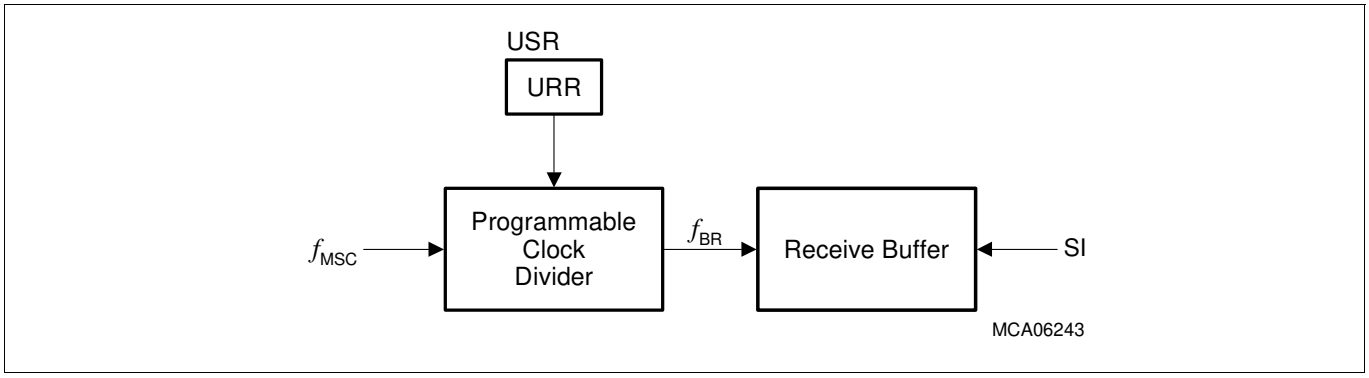


Figure 536 Upstream Channel Clock Circuitry

The serial data input SI is evaluated with the baud rate clock f_{BR} in the middle of each bit cell, and latched in case of a data bit. The baud rate clock f_{BR} is derived from f_{MSC} by a programmable clock divider. The frequency of f_{BR} determines the width of a received bit cell and therefore the baud rate for the received data. The content of bit field USR.URR selects the baud rate according **Table 338**. The resulting baud rate formula is:

$$\text{Baud rate}_{\text{MSC Upstream Channel}} = \frac{f_{MSC}}{DF} \quad (38.3)$$

Table 338 Upstream Channel Divide Factor DF Selection & Baud Rate

USR.URR	Divide Factor DF	Baud Rate
000_B	Reception disabled	–
001_B	4	$f_{MSC}/4$
010_B	8	$f_{MSC}/8$
011_B	16	$f_{MSC}/16$
100_B	32	$f_{MSC}/32$
101_B	64	$f_{MSC}/64$
110_B	128	$f_{MSC}/128$
111_B	256	$f_{MSC}/256$

Note: With the USR.URR = 000_B the upstream channel is disabled and data reception is not possible.

The content of bit field USR.URR determines the operation of an internal sampling reload counter that is clocked with f_{MSC}. **Figure 537** shows the operation of the sampling counter at the beginning of an upstream frame with a divide factor DF of 8 (USR.URR = 010_B is equal to DF = 8) which means eight sampling clocks per each frame bit cell.

When the upstream channel is in idle state, it waits for a falling edge (1-to-0 transition) at SI. Therefore, the sample counter starts counting up and is reset when the selected divide factor DF as shown in **Table 338** is reached. In the middle of the sampling counter's count range, the logic state at SI is evaluated and, in case of a data bit, latched in the receive buffer's shift register. With the reload of the sampling counter, the shift register is shifted by one bit position.

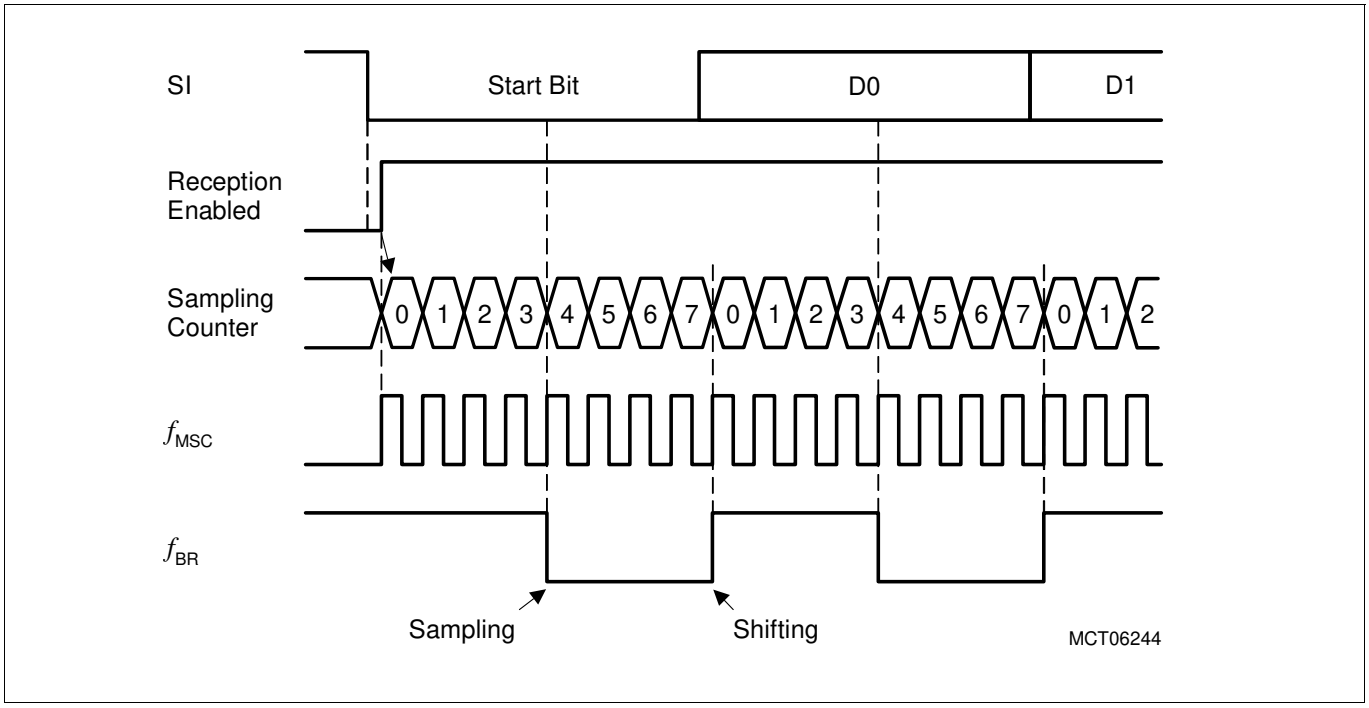


Figure 537 Upstream Channel Sampling with URR = 010_B

38.3.1.2.5 Spike Filter

The upstream channel input line SDI is sampled using a built-in spike filter with synchronization stage, both clocked with f_{MSC}. The spike filter is a chain of flip-flops with a majority decision logic (2 out of 3). A sampled value that is found at least twice in three samples is taken as data input value for SI.

38.3.1.2.6 Upstream Timer

The upstream timer monitors the time interval between two subsequent upstream frames. The timer reloads its starting values from the register **USCE** and starts counting downwards. If the time interval between two frames is longer than allowed, the counter reaches the value of zero, raises an upstream timer interrupt and stops counting. The counter reloads and starts counting on four events:

- Enabling of the MSC module after reset via **CLC**.DISR
- Resetting the module
- Writing the bit fields **USCE**.USTOPRE or **USCE**.USTOVAL per software
- Detecting the falling edge of the start bit of a frame

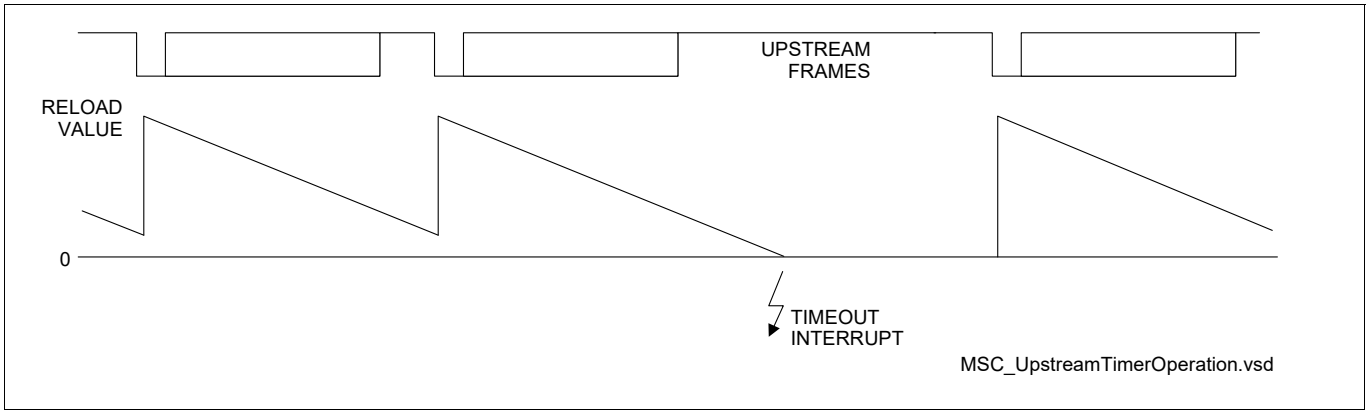


Figure 538 Upstream Timer Operation

The upstream watchdog timer counts bit times. The time window in which the next upstream frame must occur can be calculated according to the formula:

Time-out = BitTime * $2^{(USTOPRE+1)}$ * (USTOVAL+1)

38.3.1.3 I/O Control

The types of I/O control logic for the MSC module I/O lines are shown in **Figure 539**. The downstream channel generates five output signals that control eight MSC module outputs, split into four chip select outputs, two clock outputs, and two serial data outputs. The upstream channel has one input signal.

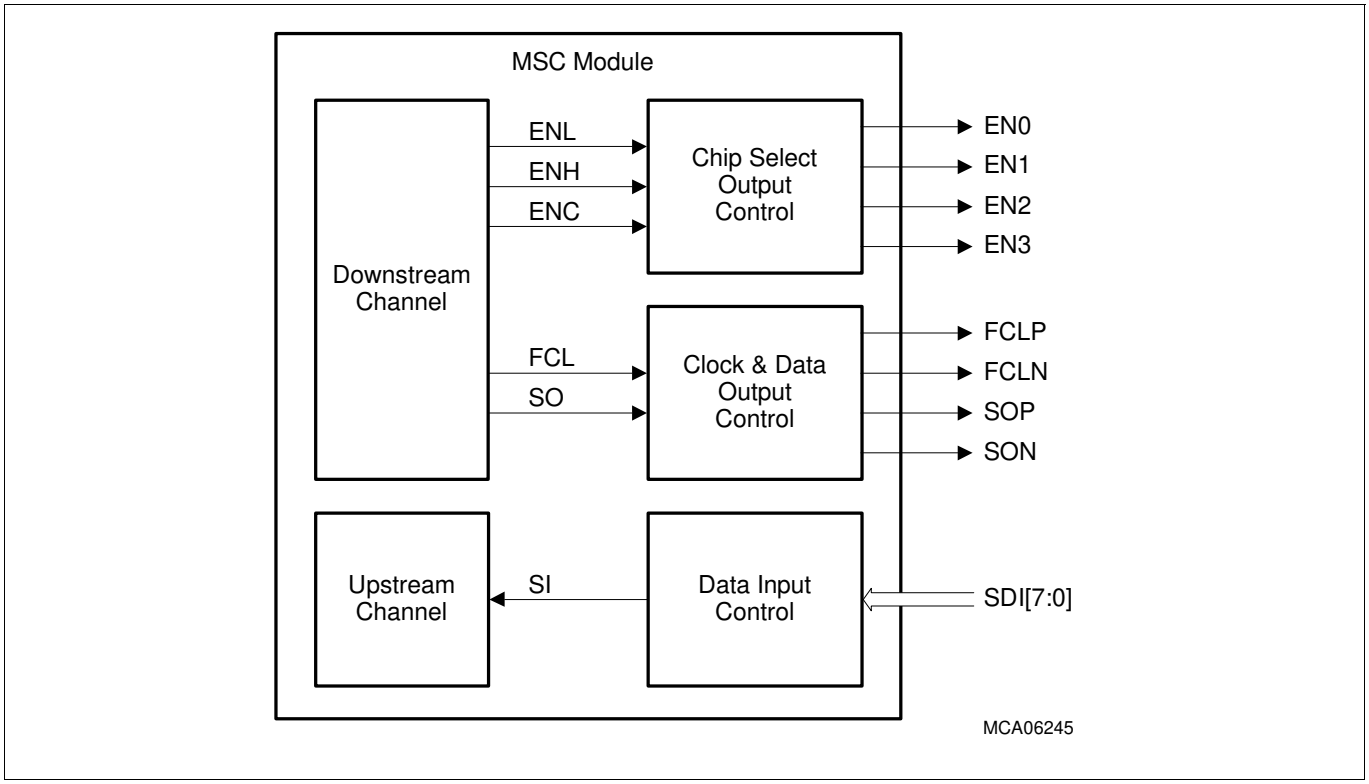


Figure 539 I/O Control

The MSC module I/O signals is controlled by bit fields that are located in the Output Control Register OCR.

38.3.1.3.1 Downstream Channel Output Control

As shown in **Figure 524** and **Figure 525**, the active phases during downstream channel operation are indicated by three enable signals:

- ENL indicates the SRL active phase of a data frame
- ENH indicates the SRH active phase of a data frame
- ENC indicates the active phase of a command frame

The chip select output control logic of the MSC uses a signal compressing scheme (similar to the interrupt request compressing scheme in **Figure 547**) that allows each of the three enable signals to be directed via a 2-bit selector to one of the four chip enable outputs EN[3:0]. This also makes it possible to connect more than one internal enable signal (ENL, ENH, ENC) to one chip enable output ENx. Three bit fields in register OCR (CSL, CSH, and CSC) determine which chip enable output becomes active on a valid internal enable signal.

In the MSC, enable signals are high-level active signals. If required in a specific application, all chip enable outputs ENx can be assigned for low-level active polarity by setting bit OCR.CSLP.

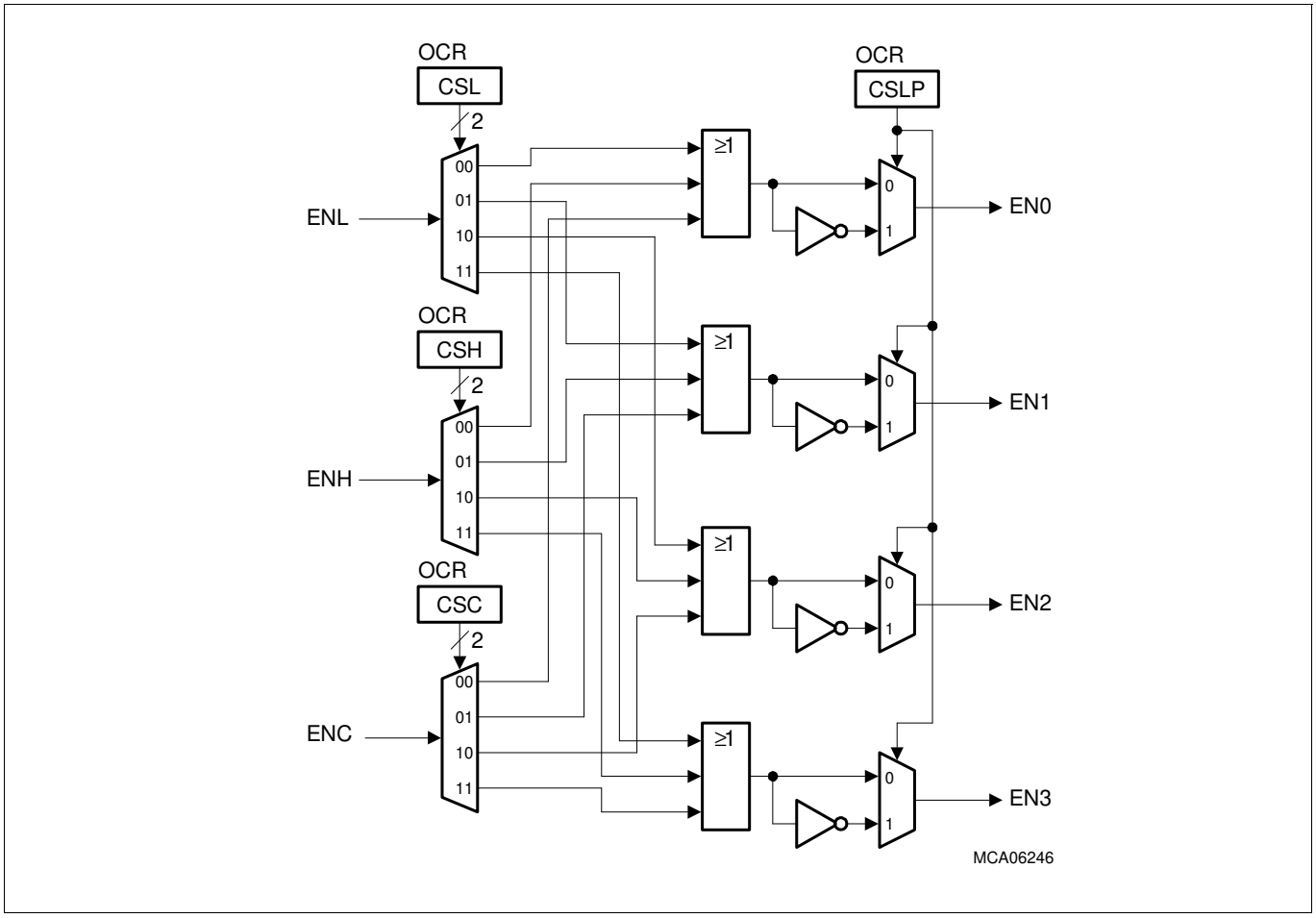


Figure 540 Downstream Channel: Chip Enable Output Control

At the MSC downstream channel, the internal serial clock output FCL and data output line SO are available outside the MSC module as two signal pairs with inverted signal polarity, FCLP/FCLN and SOP/SON. Both, clock and data outputs, are generated from the module internal signals FCL and SO according to **Figure 541**.

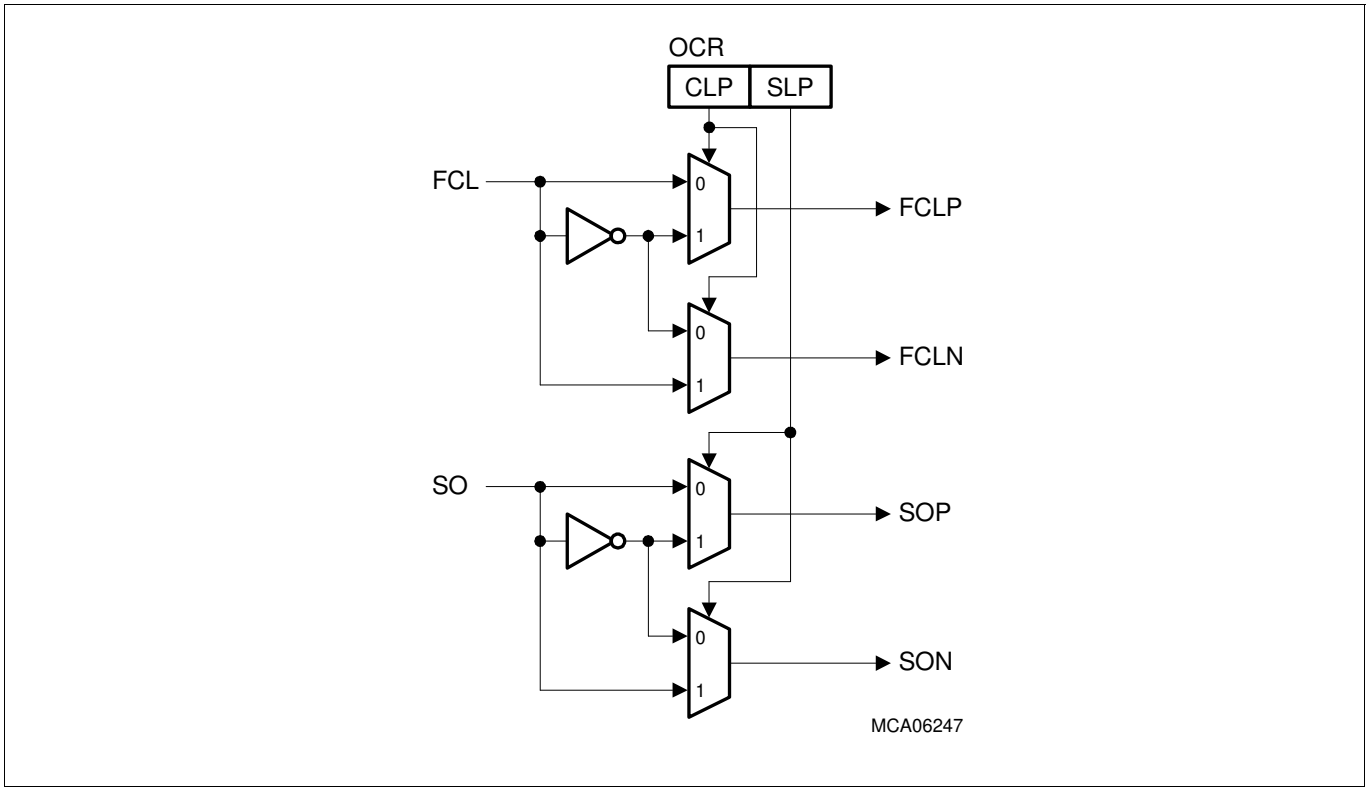


Figure 541 Downstream Channel: Clock and Data Output Control

With OCR.CLP = 0, FCLP has identical and FCLN has inverted polarity compared to FCL. Setting OCR.CLP, exchanges the signal polarities of FCLP and FCLN. An equivalent control capability is available for the SOP and SON data outputs (controlled by OCR.SLP).

One additional control capability not shown in **Figure 541** is available for the FCL signal. With OCR.CLKCTRL = 1, the FCL clock signal will always be generated, independently whether a downstream frame is currently transmitted or not. If OCR.CLKCTRL = 0, FCL becomes only active during the active phases of data or command frames (not during passive time frames).

38.3.1.3.2 Upstream Channel

As shown in **Figure 542**, the MSC upstream channel can be connected to up to eight SDI[7:0] serial inputs. Bit field OCR.SDISEL selects one out of these input lines (input signal SDI). If OCR.ILP = 0, SDI is directly connected to the serial receive buffer input SI. If OCR.ILP = 1, SDI is connected to input SI via an inverter.

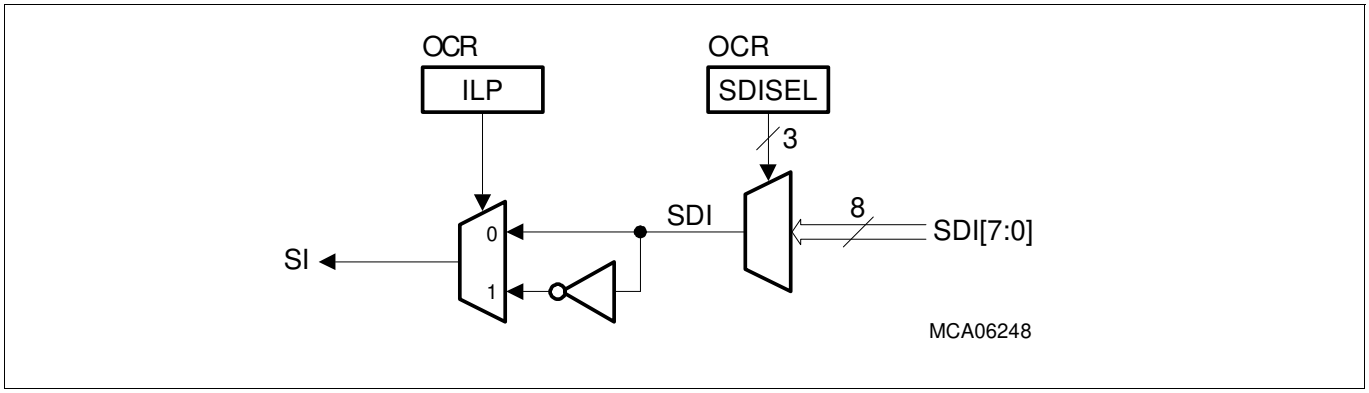


Figure 542 Upstream Channel Serial Data Input Control

38.3.1.4 MSC Interrupts

The MSC module has seven interrupt sources and five service request outputs. A service request output is able to generate interrupts (controlled by a service request control register) or DMA requests. The service request output assignment, interrupt or DMA request, is specific for each microcontroller that is using the MSC. In this section, the term “interrupt request” has the meaning of “service request” that is able to handle interrupt or DMA requests.

Each interrupt source is provided with a status flag, enable bit(s) with software set/clear capability, and an interrupt node pointer. An interrupt event, internally generated as a request pulse, is always stored in an interrupt status flag that is located in the Interrupt Status Register ISR. All interrupt status flags can be set or cleared individually by software via the interrupt Set Clear Register ISC. Software-controlled interrupt generation can be initiated by setting the interrupt status flag of the corresponding interrupt. Each interrupt source can be enabled or disabled individually. When an interrupt event is enabled, a 2-bit interrupt node pointer determines which of the service request outputs will be activated. See **Figure 547**.

Table 339 shows the seven MSC interrupt sources.

Table 339 MSC Interrupt Sources

Interrupt Type	Generated by
Data frame interrupt	Downstream Channel
Command frame interrupt	
Time frame finished interrupt	
Receive data interrupt	Upstream Channel
Upstream Time-out Interrupt	
Sync FIFO overflow	ABRA Block
Sync FIFO underflow	

38.3.1.4.1 Data Frame Interrupt

A data frame interrupt can be generated when either the first or the last data bit of the downstream channel is shifted out and becomes available at the SO output line (see also **Figure 525**). Bit ICR.EDIE selects which case is selected.

Note: If ICR.EDIE = 10_B, an interrupt at the first data bit is only generated if DSC.NDBL is not equal 00000_B. This means, at least one SRL bit must be shifted out for the first data bit shifted interrupt to become active.

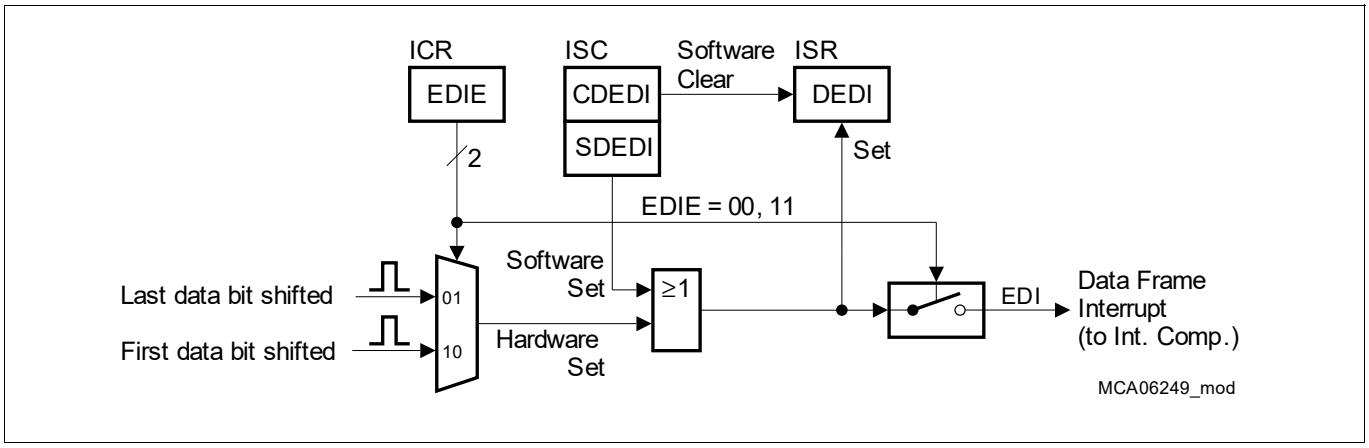


Figure 543 Data Frame Interrupt Control

38.3.1.4.2 Command Frame Interrupt

A command frame interrupt can be generated at the end of a downstream channel command frame, directly after the active phase, see also **Figure 524**.

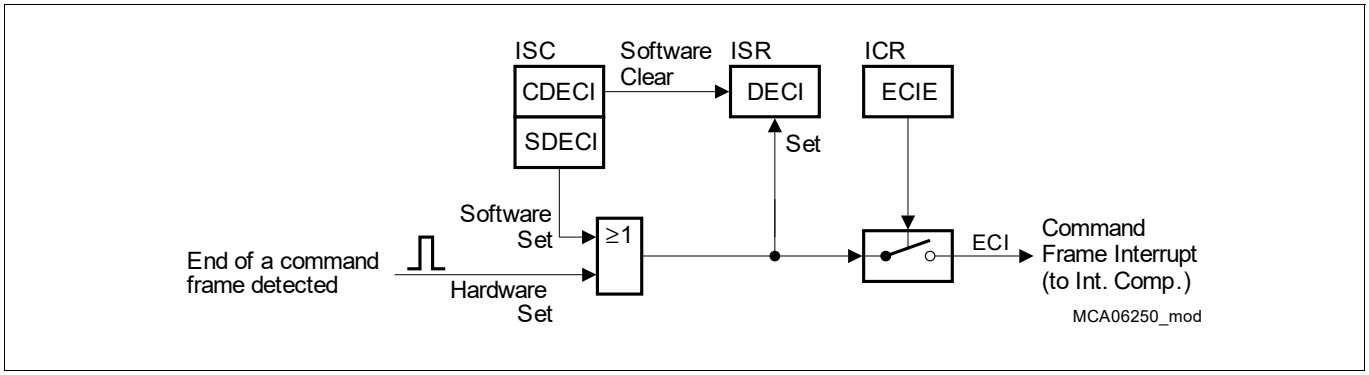


Figure 544 Command Frame Interrupt Control

38.3.1.4.3 Time Frame Finished Interrupt

A time frame finished interrupt can be generated at the end of a downstream channel passive phase.

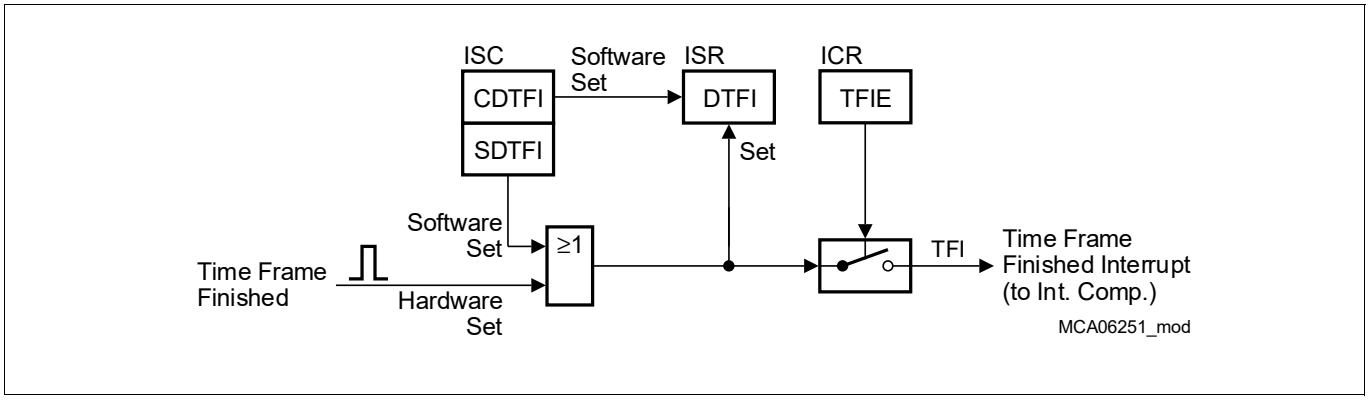


Figure 545 Time Frame Finished Interrupt Control

38.3.1.4.4 Receive Data Interrupt

Whenever the upstream channel receives data in registers UDx (x = 0-3), the MSC is able to generate an interrupt. Three interrupt generation conditions can be selected for the receive data interrupt:

- Each update of UDx (x = 0-3) generates a receive data interrupt.
- Each update of UDx (x = 0-3) generates a receive data interrupt when the updated value is not equal 00_H.
- Only an update of register UD3 generates a receive data interrupt.

The selection of the interrupt generation condition is controlled by bit field ICR.RDIE. Setting ICR.RDIE = 0 disables the receive data interrupt in general. ISR.URDI is the interrupt status flag that can be set or cleared when writing bits ISC.SURDI or ISC.CURDI with a 1. If the fractional divider is used, then the software should clear the URDI flag by using ISC.CURDI with a delay of one bit time after the interrupt signal has been generated. The same applies to clearing the UDx.V flag via UDx.C. Otherwise, due to the latencies introduced by the fractional divider, the clear may fail.

The receive interrupt is generated either after the second stop bit, or delayed for one bit time by setting the **USR**.SRDC, as shown in **Figure 534**.

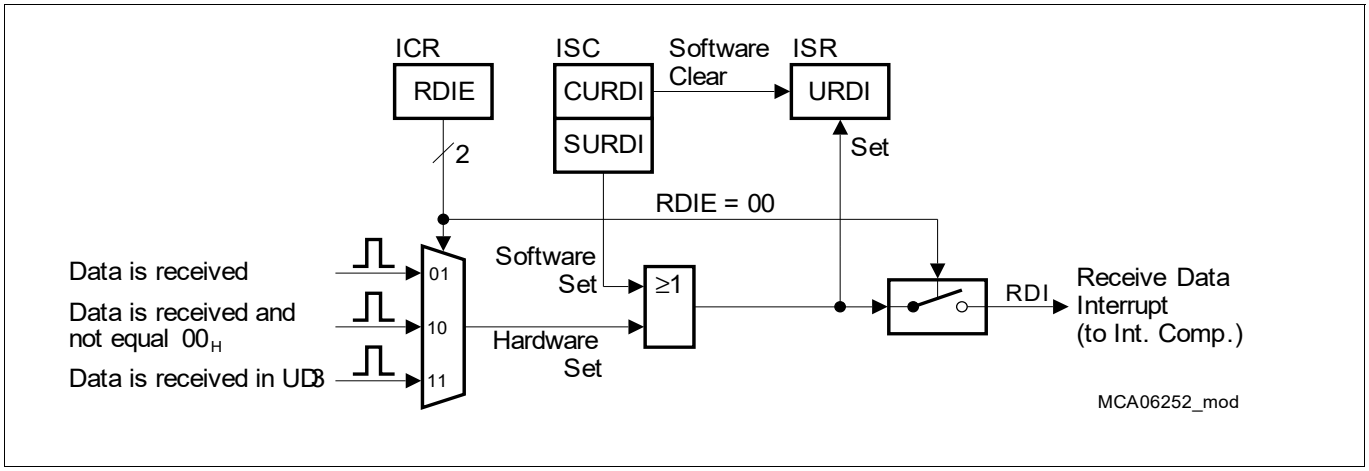


Figure 546 Receive Data Interrupt Control

38.3.1.4.5 Interrupt Request Compressor

The interrupt control logic of the MSC uses an interrupt compressing scheme that allows high flexibility in interrupt processing. The seven interrupt sources can be directed to five service request outputs SR[4:0] as shown in the **Figure 547**. This makes it possible to connect more than one interrupt source to one interrupt output SRx.

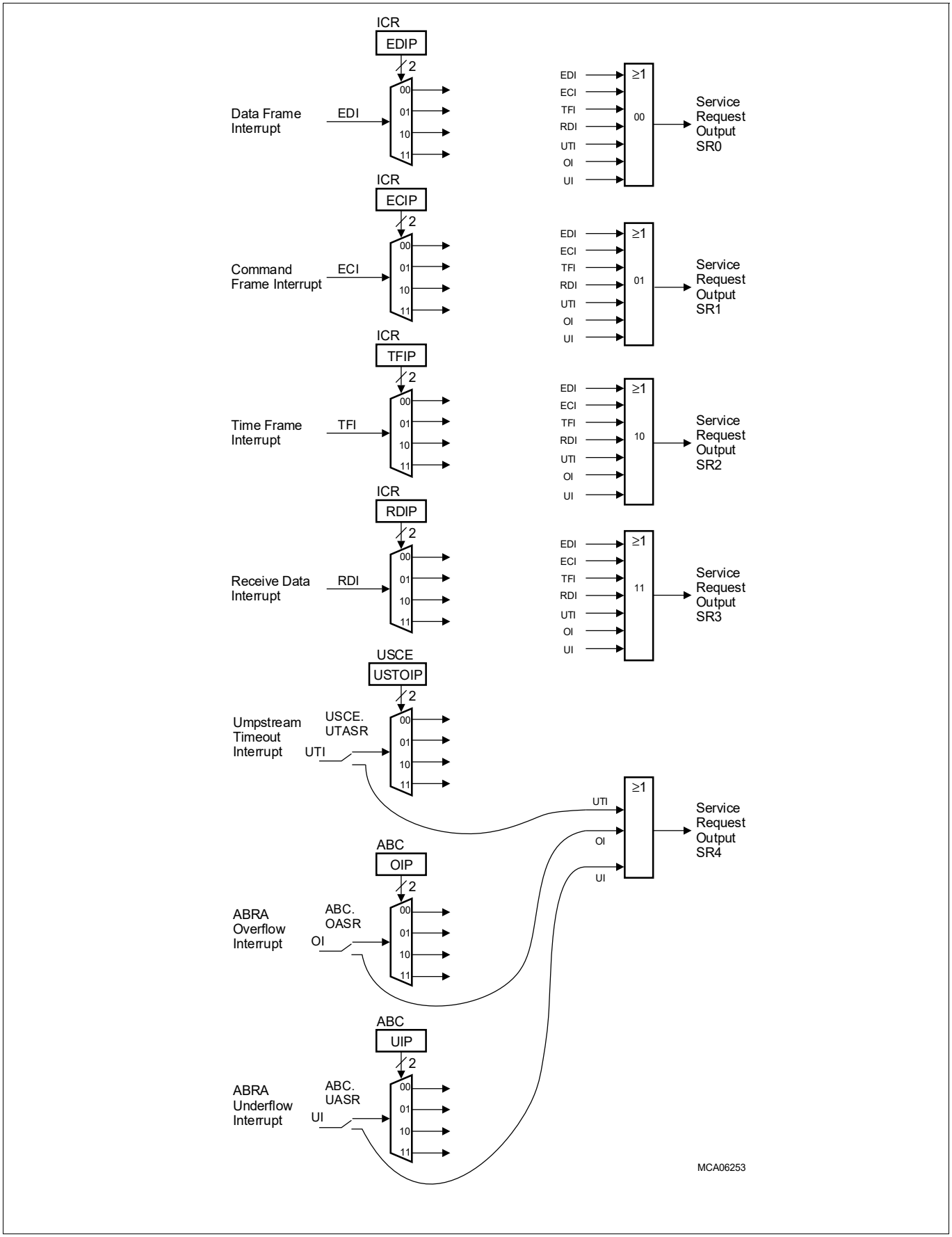


Figure 547 MSC Interrupt Request Compressor

38.3.2 CX (Command Extension) Mode

Command Extension Mode overrides the following standard features of the MSC protocol:

- Command Frames:
  - Instead of 32 bits, Command Frames contain up to 2x32 bits and two select bits set to “1”, one before each 32-bit block. One command Frame now contains two commands, both with same length
  - The bit field **DSC**.NBC is reused as is, and defines the length of both commands
  - The command chip select ENC is active during the whole active phase of the frame
  - The passive phase extension is lenghtened to cover 2 to 127 bit times, see bit field **DSTE**.PPCE and **DSTE**.PPCEM. This is valid only in ABRA active mode. When ABRA bypassed, the command passive phase is constant and fixed to two. The PPCEM bit is needed and extends the command passive phase only in the CX mode (CX=1), where a command can be up to 64 bit long. If CX=0. PPCEM bit is ignored and treated as always 0.
- Data Frames:
  - Instead of activating two enable signals ENL and ENH, Data Frames activate only one Enable Signal, ENL
  - It is responsibility of the user to mandatory activate the select bits in the data frames with **DSC**.ENSELL and ENSELH, in order to ensure the same frame structure of the command and the data frames.
  - the existing data frame length bit fields are reused as they are: **DSC**.NDBL, **DSC**.NDBH, **DSCE**.NDBLE, and **DSCE**.NDBHE. It is the user responsibility to program them symmetrically if required, and program the **DSCE**.EXEN, for example to set the same data frame length as the command frame length, which is the standard use-case.
  - The passive phase of the data frames is reused as is
- Baud Rate: it is possible to configure the MSC module to drive with maximal speed of 80 MBaud. The divider **DSTE**.NDD divides the module kernel frequency. When NDD=0 the module runs with 100MHz, with NDD=2 with 33MHz.

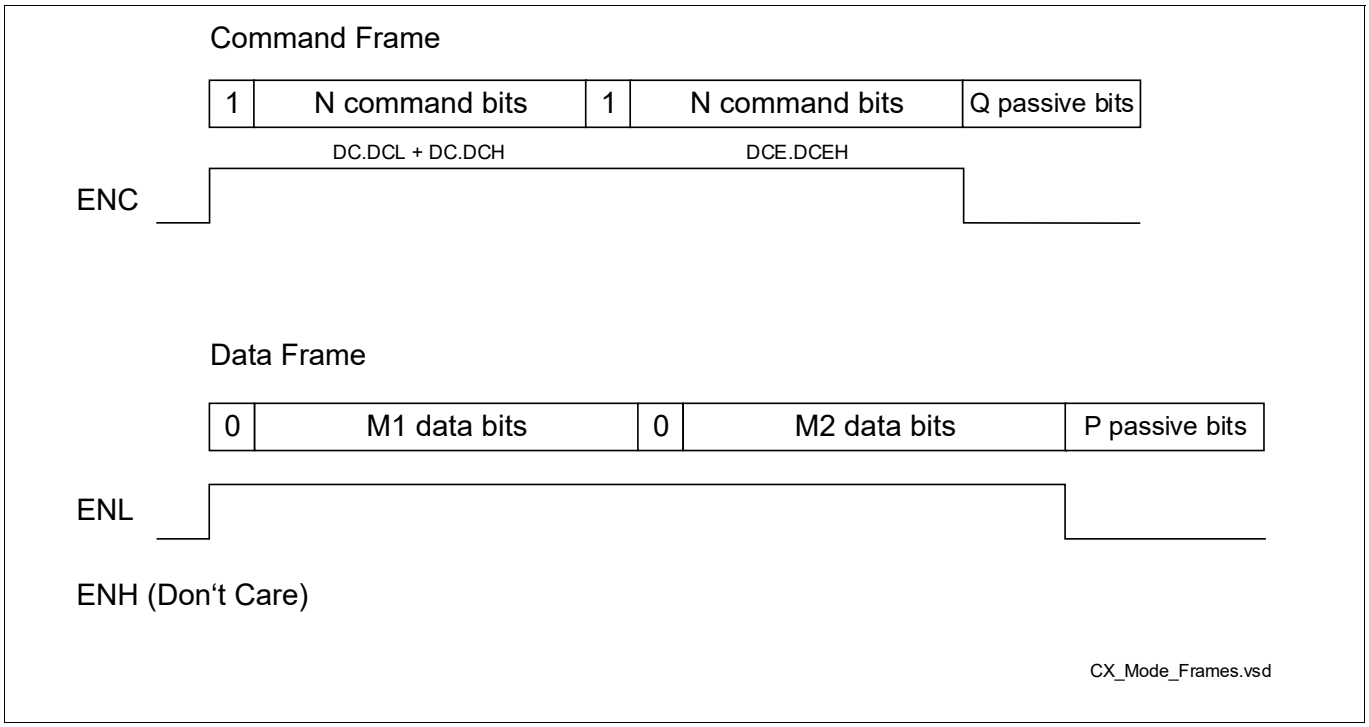


Figure 548 MSC Frames in CX mode

This is the list of the additional bit fields and registers correlated to CX mode.

- **DSTE**.CX bit, activating the CX mode
- **DSTE**.FM bit, activating the fast mode (up to 80MBaud)
- **DSTE**.UL1 protection bit, for unlocking the CX and FM bits for one clock cycle
- **DSTE**.PPCE extended by one bit, in order to lenghten the passive phase of the command frame
- **DCM**, Downstream Command Mirror register, providing convenient way to send more than 32 bit long commands by using DMA with memory address wrap around.
- **DCE**, Downstream Command Extension register, defining the second half of the command frame. In CX mode, always write first the **DCE** register, and then **DC** or **DCM**, because write to the **DC** / **DCM** triggers the command transmission

38.3.3 ABRA (Asynchronous Baud Rate Adjustment Block)

The Asynchronous Baud Rate Adjustment Block (ABRA) takes as an input a serial MSC stream with one baud rate and outputs the same serial stream with another baud rate, asynchronous to the input one. The input signal bundle consists of serial clock, data and select signal, and the output signal bundle consists of the same signals.

Additionally, the output clock of an MSC without ABRA can be only set to 50% duty cycle, while with the ABRA block, other duty cycles are also possible.

38.3.3.1 Overview

The ABRA block is located between the MSC kernel and the output pins.

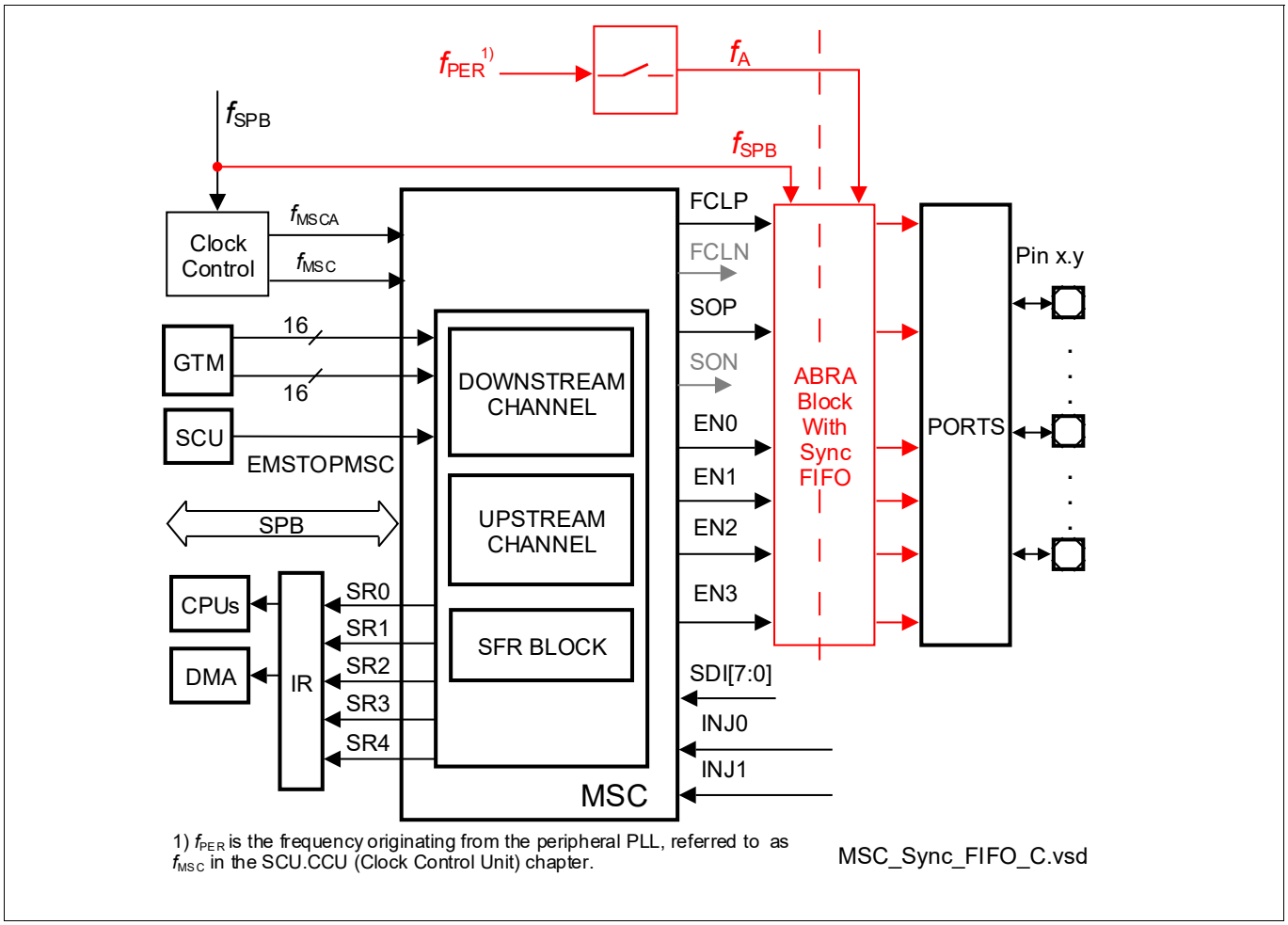


Figure 549 ABRA Overview

The user has two possibilities: either to use the ABRA block, or to bypass it. In case of using ABRA, additional delay in the signal path between the module and the pins of up to three f_A periods is to be taken into account. Additionally, some margin between the input frame length and the output frame length must be guaranteed by the customer, in order to avoid overflow of the ABRA block.

If the ABRA is not used, the behavior of the MSC module is identical with the previous implementations. The user can also choose between using the 64-bit extension or not, which makes 4 combinations in total, as shown in **Figure 550**

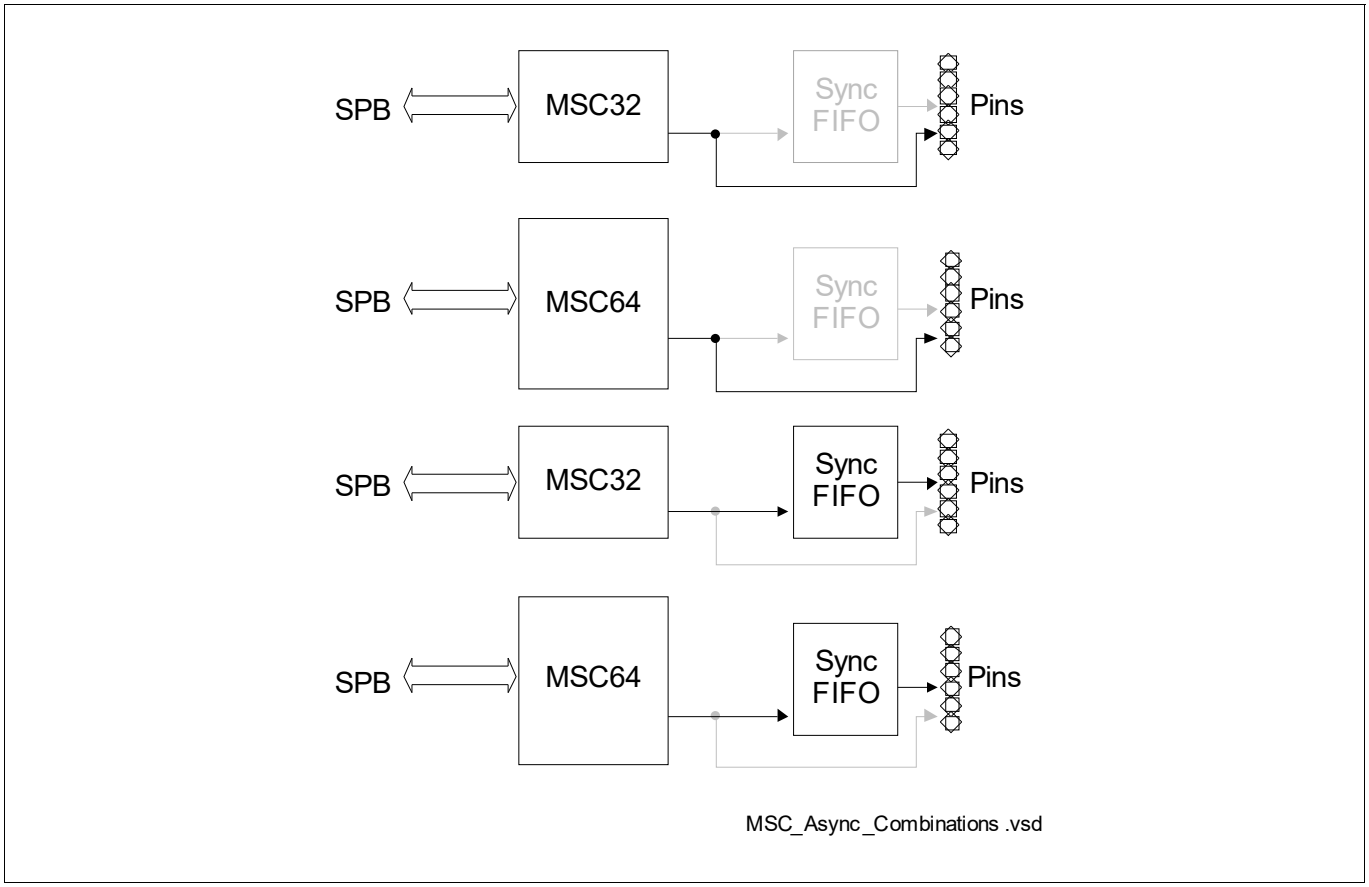


Figure 550 Use-Case Combinations

All configuration parameters related to the ABRA block are located in the ABC (Asynchronous Block Configuration) register.

38.3.3.2 Timing Issues

The ABRA block introduces a delay in the signal path from the MSC module to the pins. However it does not influence significantly the timings between the signals themselves. The output delays between the shift clock, data and enable signals remain approximately the same as if the ABRA block is not used, and are covered in the timings of the MSC module as defined in the data sheet.

38.3.3.3 Adjusting the Passive Phase of a Frame

The main purpose of the Asynchronous Baud Rate Adjustment block is to transform a frame going in with a higher baud rate (for example 50Mbaud), to a frame going out with a lower baud rate (for example 40MBaud). Therefore, the duration of the outgoing frame is longer then the duration of the incoming frame. In order not to overflow the ABRA block with too many incoming frames, there must be a pause between them, which is adjusted by

configuring the passive phase of the frames. This pause plus the incoming frame duration must be longer or equal to the outgoing frame duration plus some margin.

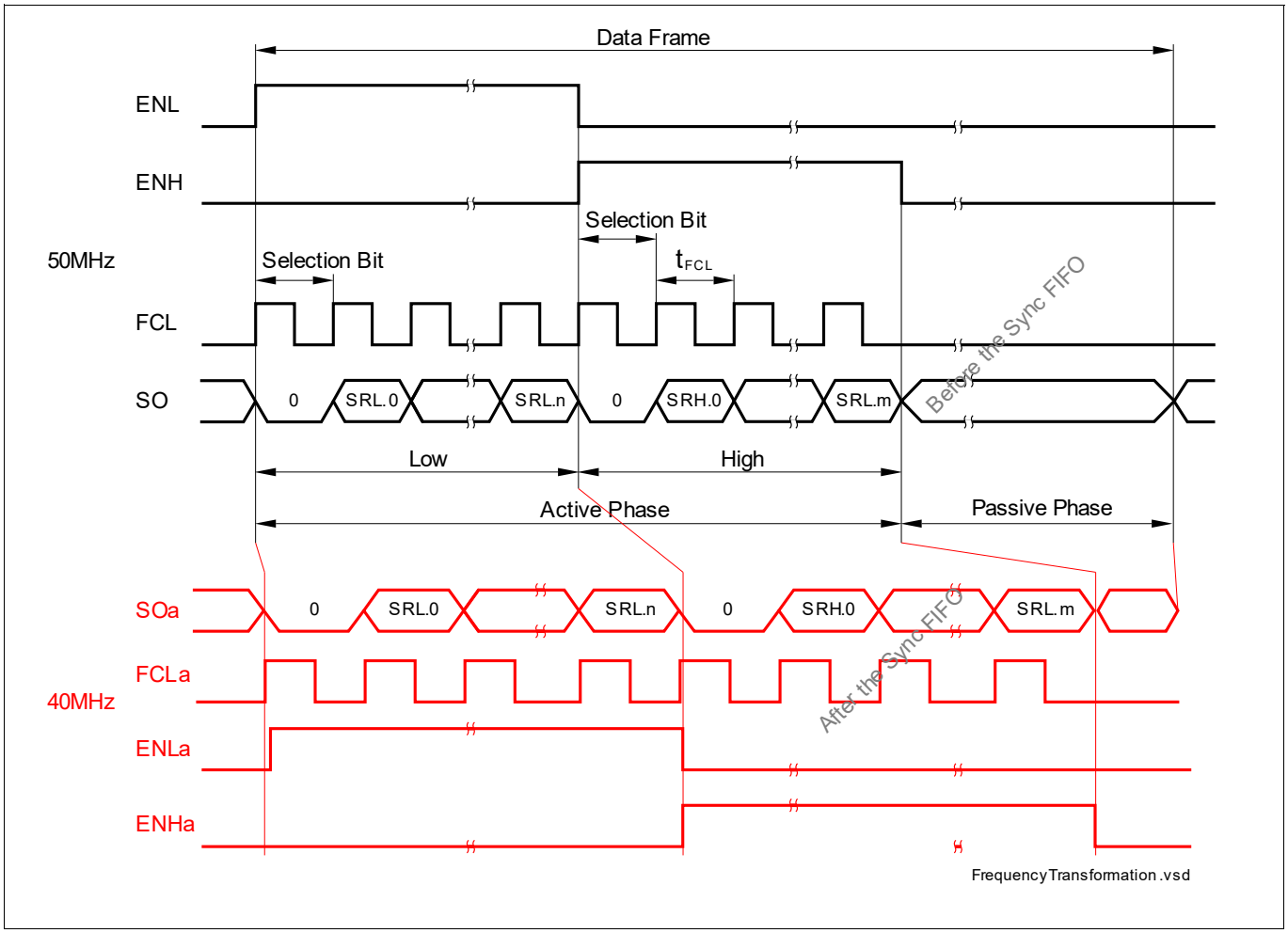


Figure 551 Passive Phase Adjustment

In order to allow for frame length adjustments:

- for command frames, the bit field **DSTE**.PPCE defines a programmable passive phase in a range of 2 to 65 input frequency cycles, and
- for data frames, the bit field **DSTE**.PPDE extends the passive phase from 2 to 31 to 2 to 127 bit times.

The length of the frame in the MSC domain (data and passive bits) must be longer than the length of the frame in the asynchronous domain, in order to prevent overload in case of back to back transmission:

$(N_D + N_P) * T_{FCL} > (N_D + N_{PA}) * T_{FCLA}$

which results to the following inequality for setting the length of the passive phase in the MSC domain:

$N_P > (N_D + N_{PA}) * T_{FCLA} / T_{FCL} - N_D$ or equivalently $N_P > (N_D + N_{PA}) * f_{FCL} / f_{FCLA} - N_D$

for example if f_{FCL} = 50MBaud, f_{FCLA} = 40MBaud, N_D=32 and N_{PA}=3, then $N_P > (32 + 3) * 1.25 - 32$, $N_P > 12$

Where:

T_{FCL}: serial clock period in the MSC domain with activated ABRA: $T_{FCL} = 1 / BaudRate_{MSCA}$,
$BaudRate_{MSCA} = f_{SPB} / (2*m)$, see **Intermediate Downstream Channel Baud Rate** with ABRA block.

T_{FCLA}: serial clock period of the ABRA block $T_{FCLA} = 1 / BaudRate_{ABRA}$,
$BaudRate_{ABRA} = f_A / [($**ABC**. NDA+1) * (**ABC**.LOW+1 + **ABC**.HIGH+1)]

N_P: number of the passive bits in the MSC domain (see **DSTE**.PPDE and **DSC**.PPD)

N_{PA}: number of the targeted passive bits in the asynchronous domain

N_D: number of data bits **DSC**.NDBH+NDBL+SELH+SELL+**DSCE**.NDBLE+NDBHE in both domains.

Note: If the passive phase settings in the MSC module are not correctly configured in such a way that two frames merge, that is, there is no passive phase on the pins at all, an overflow interrupt is triggered. If the passive phase is wrongly shorter, no error signal is generated.

Input/Output Baud Rate Ratio

The ABRA output baud rate going to the pins must be lower than the input baud rate coming from the MSC module and higher than roughly half of the input baud rate (more precisely higher than 57% of the input baud rate in worst-case, due to the fact that in worst-case MSC data frame can contain 64 data bits plus additionally two select bits).

38.3.3.4 Jitter of the Downstream Frames

This section describes the jitter effects affecting the starting point of the downstream frames, depending on the setting of the serial clock - serial clock only during the active phase of a frame, or continuous clock.

38.3.3.4.1 Jitter in Active Phase Clock Mode

The active phase clock mode is set when **OCR**.CLKCTRL = 0. In this mode the serial clock is generated only during the transmission of the data bits. The ABRA outgoing frame passive phase duration jitters for +- one f_A cycle. The example in the **Figure 552** assumes 50MBaud input and 40MBaud output baud rate, f_A = 80MHz, back to back frames.

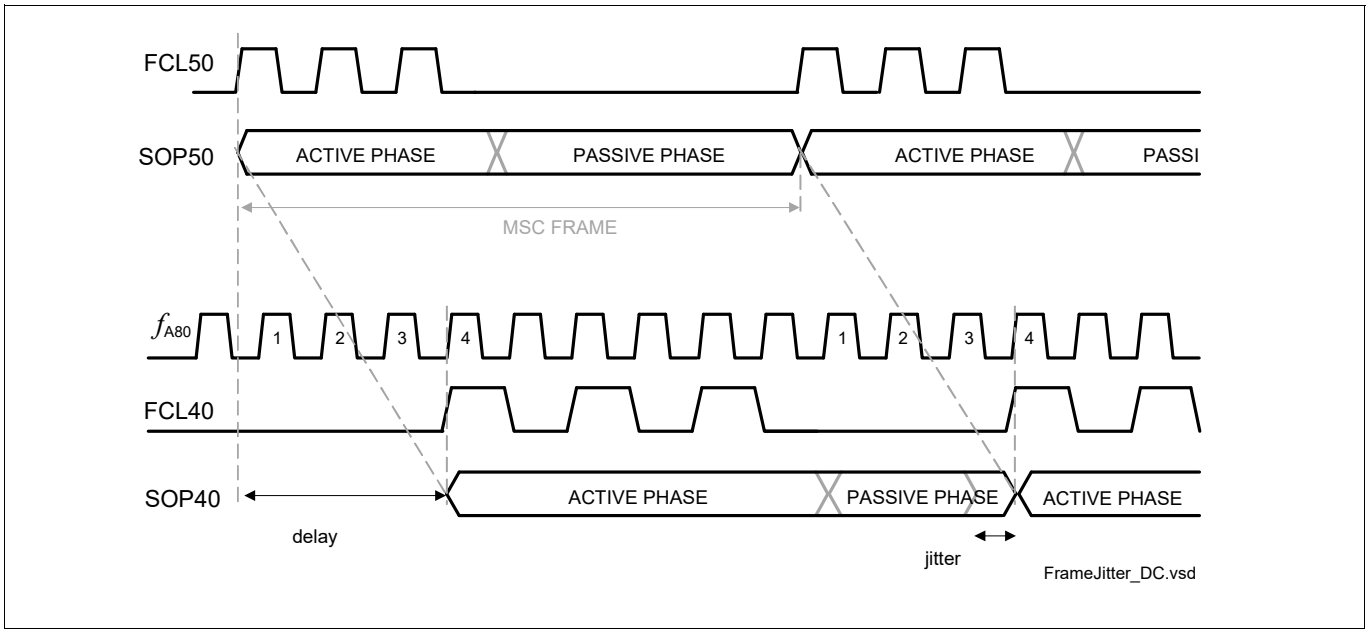


Figure 552 Frame Jitter in Active Phase Clock Mode

38.3.3.4.2 Jitter in Continuous Clock Mode

Continuous clock mode is set when **OCR**.CLKCTRL = 1. In this mode the serial clock is always on. The jitter is one output bit time.

The average length of the output frames is equal to the length of the input frames. The ABRA block generates an averaging effect similar to that of a fractional divider. It introduces occasionally jitter of one bit time in the output frame length, in the passive phase of the frame, in order to adjust the average output time raster to the ideal input time raster. The granularity of the input time raster is one input bit time (for example 20ns at 50MBaud), the granularity of the output time raster is one output bit time (for example 25ns at 40MBaud).

The example in the **Figure 553** assumes 50MBaud input baud rate and 40MBaud output baud rate, f_A = 80MHz, back to back frames.

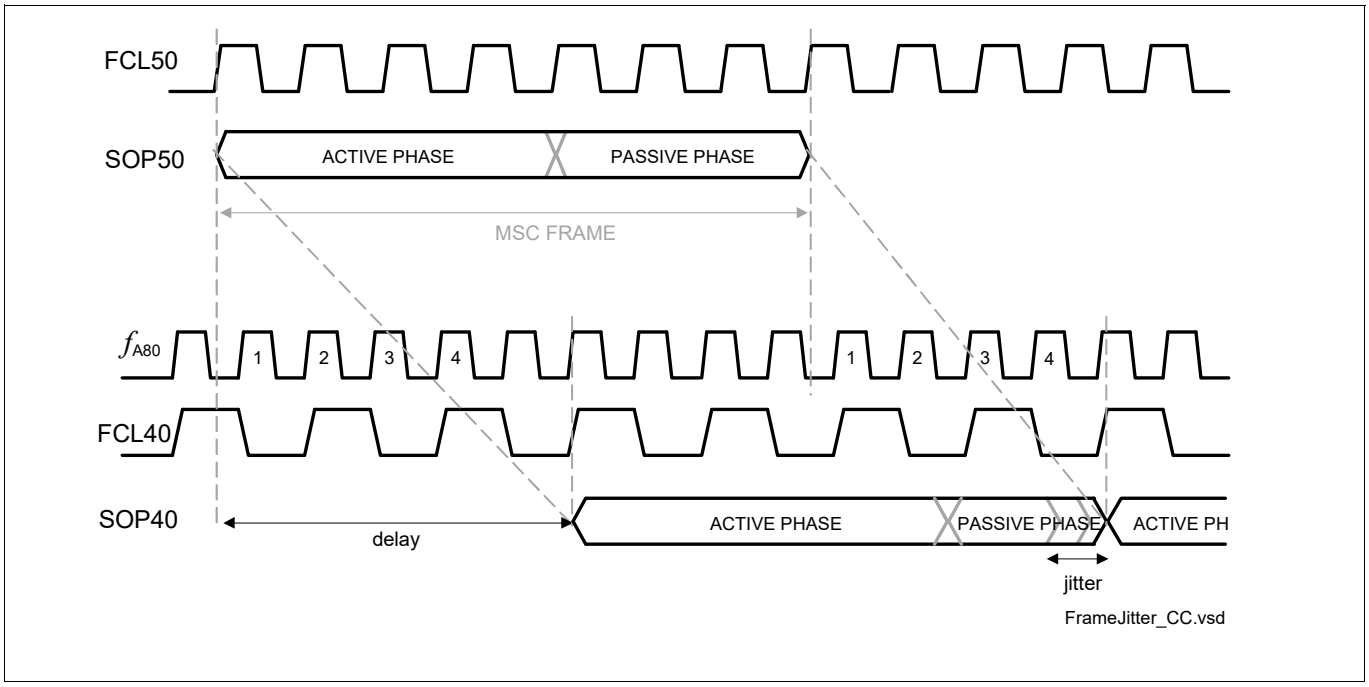


Figure 553 Frame Jitter in Continuous Clock Mode

38.3.3.5 Interrupt Position with the ABRA Block

The position of the MSC interrupts within a frame depends only on the internal time schedule of the MSC module in both use cases with and without ABRA block. Therefore the interrupt position in the internal time raster is constant and is the same as in the previous versions of the MSC module, and does not depend on the asynchronous baud rate which is seen on the pins. This behavior affects only the downstream interrupts, which are:

- Data Frame Interrupt
- Command Frame Interrupt
- Time Frame Finished Interrupt

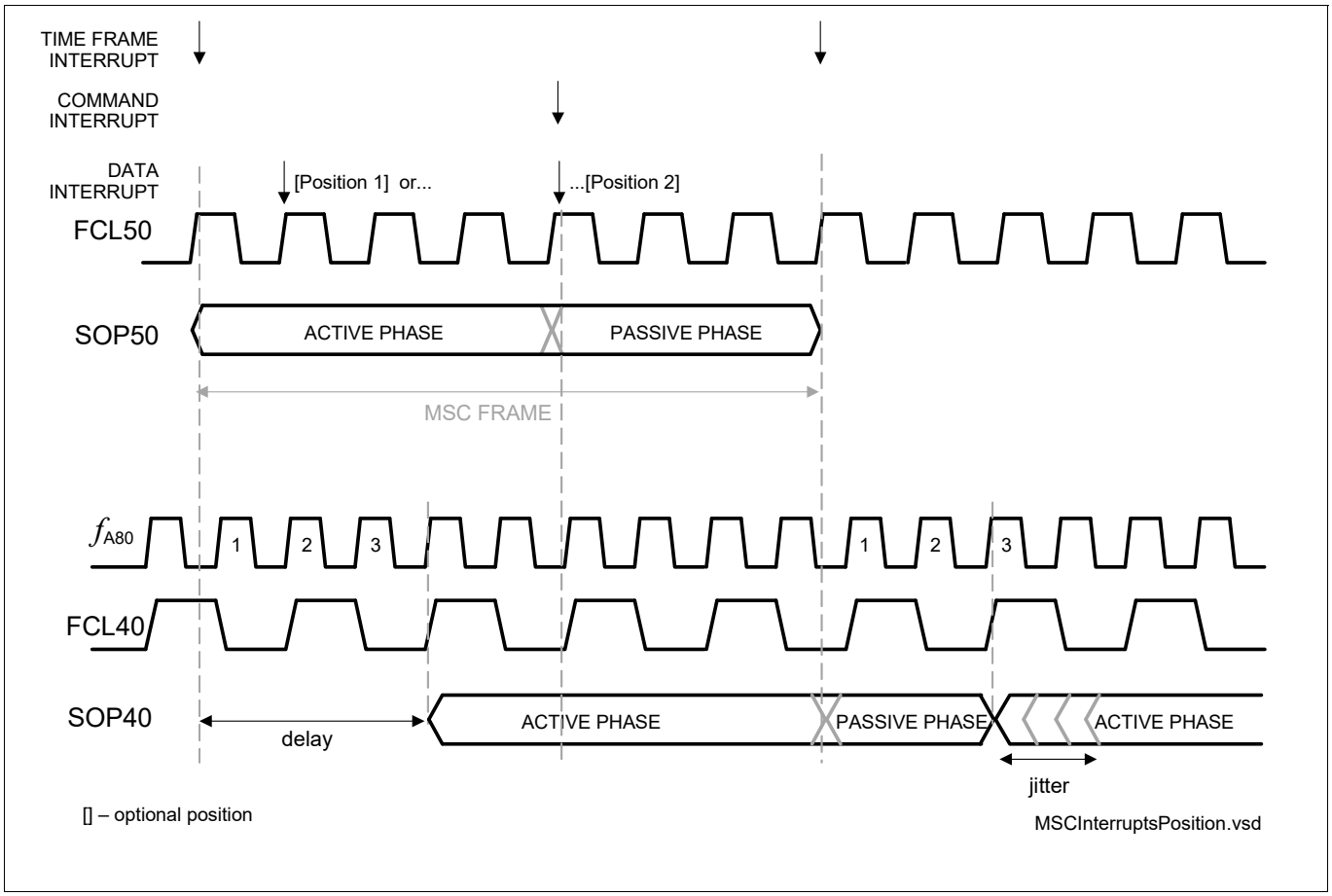


Figure 554 Interrupt Position with the ABRA Block

The downstream interrupts can be used for updates of the shadowed rw bit fields for the next frame transmission. Such bit fields are located in the Data Register **DD**, Command Register **DC**, Data Control **DSC** and Data Status **DSS**. Due to the next-frame-update effect, the small jitter and delay effects of the frames as seen on the pins is not relevant, as well as the position of the interrupt in the asynchronous frame.

38.3.3.6 Configuring the ABRA block

This section describes all the configurable parameters of the ABRA block. They are located in the **ABC** (Asynchronous Block Configuration) register.

Output Baud Rate

The ABRA block shifts out the data with a baud rate generated by dividing the clock f_A by using the bit fields **ABC**.HIGH, **ABC**.LOW and **ABC**.NDA.

The high and the low time of the shift clock are generated by a separate 4-bit n-divider, each operating in the range of 1 to 16. Therefore the duty cycle can be finely configured to be 50% or any other ratio. This may provide some advantages when the communicating devices have asymmetrical timings (output delays or set-up/hold times). See the description of the bit-fields **ABC**.HIGH and **ABC**.LOW.

The output baud rate of the ABRA block is defined with the formula:

Baud Rate = f_A / [(**ABC**. NDA+1) * (**ABC**.LOW+1 + **ABC**.HIGH+1)]

Committed Baud Rates

The following table defines the settings for the baud rates that are mandatory for the ABRA block for various devices. The ABRA **ABC**.LOW and HIGH are equal in the range of 0 to 7, and divide the f_A with 2, 4, 6, 8, 10, 12, 14, 16. The clock pre divider **ABC**.NDA=000_B.

The planned f_A frequencies for the various devices are the following:

- 160 MHz (80 MHz) and 200 MHz without FMPLL, 100 MHz with FMPLL

This results in the following table (* baud rates above 40MBaud are beyond the standard MSC specification):

Table 340 Committed Baud Rates in MBaud

Division ratio	200 MHz	160 MHz	100 MHz	80 MHz
2	-	80,00[1)]	50,00*	40,00
4	50,00*	40,00	25,00[1)]	20,00
6	33,33	26,67[1)]	16,67	13,33
8	25,00[1)]	20,00	12,50	10,00
10	20,00	16,00	10,00	8,00
12	16,67	13,33	8,33	6,67
14	14,29	11,43	7,14	5,71
16	12,50	10,00	6,25	5,00

1) For baud rate of 80 MBaud, FM mode with NDD=0 must be used. For baud rates of 25MBaud and 26,67 MBaud, it is recommended to use FM mode with NDD=2, so that the MSC module fills the ABRA FIFO with baud rate of 33.3 MBaud.

Continuous Shift Clock

As defined in the bit field **OCR**.CLKCTRL, the shift clock signal can be active either:

- only during the data transmission time
- continuously

Overflow and Underflow Interrupts

If the input and the output timings of the ABRA block are not adjusted to each other, an overflow of the ABRA block will occur and an error interrupt will be triggered on line X.

Overflow occurs either if the input data rate is too high or the output data rate is too low.

The corresponding bit fields are **ABC**.OVF, OFM, OIE.

Underflow occurs either if the input data rate is too low or the output data rate is too high.

The corresponding bit fields are **ABC**.UNF, UFM, UIE

See also **Interrupt Request Compressor**.

This feature can not guarantee protection against software errors like reconfigure the baud rates in the middle of a frame.

The configuration of the ABRA block and the MSC kernel is static and the time behavior is deterministic and calculable. Therefore, an overflow/underflow event always signals some configuration error resulting in unadjusted input/output baud rate ratio (see **Input/Output Baud Rate Ratio**). In normal operation and with correctly configured baud rates (clocks and clock dividers) these events do not occur.

In case of underflow/overflow event the output frame will be corrupt. The simplest solution to a sporadic event caused by a transient (noise, alpha particle) is to reset the module, reconfigure and restart the communication.

Bypass Mode

If the ABRA block is not used, it is disabled and bypassed via the bit field **ABC**.ABB.

38.3.3.7 Implementation Issues

The ABRA block operates using synchronization FIFO. The MSC module writes its serial stream to the synchronization FIFO with its serial shift clock, and the same data is read out of the FIFO by using the ABRA serial shift clock, generating out of the f_A.

Three signals are being transformed: the serial shift clock, the serial data stream and the enable (chip select) signals.

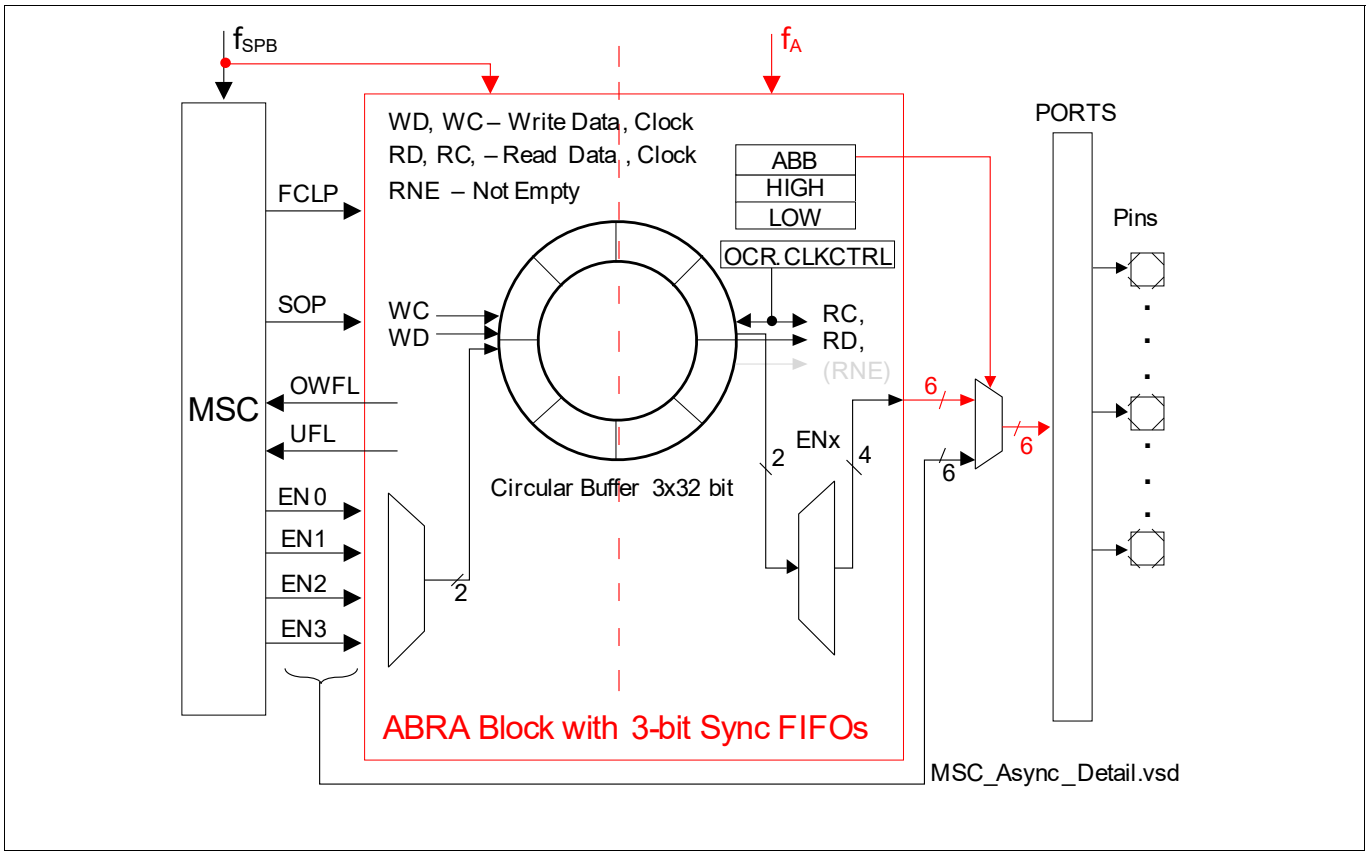


Figure 555 Implementation Details of the ABRA Block

38.3.3.8 ABRA Disable, Sleep and Suspend Behavior

This section describes the behavior of the MSC module in case of:

- switching the module on/off - disable behavior
- power saving - sleep mode
- debugging - OCDS suspend mode

The main concept is to provide behavior as similar as the behavior of the MSC module without the ABRA block, as much as possible.

38.3.3.8.1 Disable and Sleep Behavior

There is no difference in the behavior of the module in case of entering / leaving the disable and sleep state. Disable state is controlled by a bit set / cleared by software, sleep state is controlled by a hardware signal which can be disabled or enabled.

On disable/sleep state request, the module finishes the possibly ongoing frames (downstream or upstream), then acknowledges the request and the module clock is switched off.

On leaving the disable/sleep state, the module continues with the operation at the same point where it entered the corresponding state.

38.3.3.8.2 OCDS Suspend Behavior

In case of Hard Suspend, the clock is switched off immediately. A running frame will not be finished.

In case of Soft Suspend, the state is entered after the following two conditions are met:

- any ongoing upstream reception is completed, and
- any ongoing downstream frame transmission is completed, both from the MSC kernel and the ABRA block side

38.3.4 MSC Module Implementation

This section describes the MSC module interface as implemented in the device. It especially covers clock control, port and on-chip connections, interrupt control, and address decoding.

38.3.4.1 BPI_FPI Module Registers (Single Kernel Configuration)

38.3.4.1.1 System Registers

Figure 556 shows all registers associated with the BPI_FPI module, configured for one kernel.

BPI_FPI Registers Overview

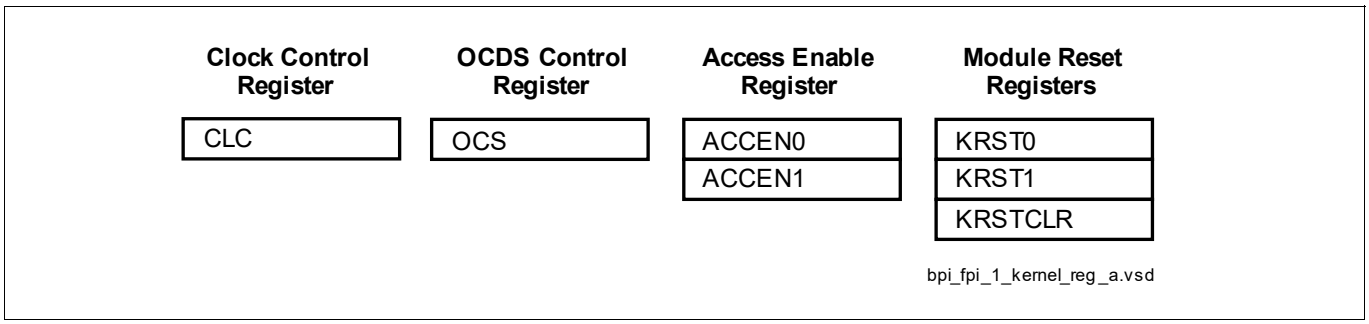


Figure 556 BPI_FPI Registers

The writes of the bus masters to the MSC module is controlled by Access Protection registers ACCENx.

The MSC implements two ACCENx registers, ACCEN0 and ACCEN1.

The ACCENx registers are protected by Safety Endinit mechanism.

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the $f_{<>}$ module clock signal, sleep mode and disable mode for the module.

CLC

Clock Control Register (000_H) Application Reset Value: $0000\ 0003_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.

Field	Bits	Type	Description
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode.
0	2, 31:4	r	**Reserved** Read as 0; shall be written with 0.

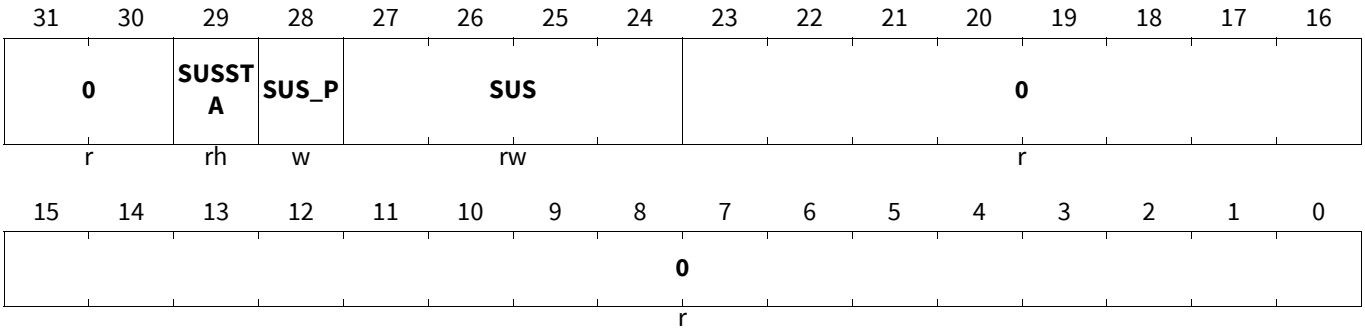
OCDS Control and Status

The OCDS Control and Status (OCS) register is cleared by Debug Reset.

The register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32 bit wide only and requires Supervisor Mode

OCS

OCDS Control and Status (0E8$_H$) Debug Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. 2_H Soft suspend **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:0, 31:30	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is

prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, …, EN31 -> TAG ID 011111B.

ACCEN0

Access Enable Register 0 ($0FC_H$) Application Reset Value: $FFFF\ FFFF_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

The Access Enable Register 1 controls write access[1] for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the T3xx devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, …, EN31 -> TAG ID 111111B.

ACCEN1

Access Enable Register 1 ($0F8_H$) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; shall be written with 0.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be reset by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is reset by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be reset to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 (0F4$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RSTSTAT	RST
r														rh	rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (reset to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; shall be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be reset by the BPI with the end of the BPI kernel reset sequence.

KRST1

Kernel Reset Register 1 **(0F0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															RST
r															rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (reset to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; shall be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **(0EC$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															CLR
r															w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; shall be written with 0.

38.3.4.2 Interface Connections of the MSC Module

The MSC module is supplied with a separate clock control, address decoding, and interrupt control logic. The service request outputs are connected with the interrupt nodes in the Interrupt Router module. Outputs of the GTM module are connected to the alternate input buses ALTINL/ALTINH. The Port Emergency Stop output from the SCU controls the corresponding inputs of each MSC module.

The serial data and clock outputs of the downstream channels of the MSC module are connected to combined GPIO/LVDS differential output drivers. Details about these four pins are defined in the ports chapter.

38.3.4.3 MSC Module-Related External Registers

Figure 557 summarizes the module-related external registers which are required for MSC programming (see also **Figure 561** for the module kernel specific registers). Generally, additionally to programming the MSC kernel and BPI registers, the Fractional Divider, the ports registers located in the Ports module and interrupt registers located in the Interrupt Router module.

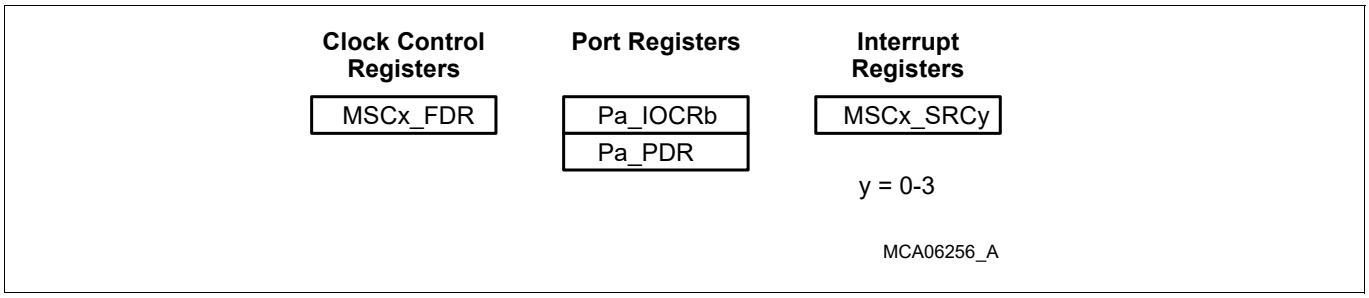


Figure 557 MSC Implementation-specific Special Function Registers

38.3.4.4 Clock Control

The following section describes the clock control and the baud rate generation of the MSC module.

The MSC module is provided with two independent clock signals (**Figure 558**):

- f_{CLC}
  This is the module clock that is used inside the MSC kernel for control purposes such as clocking of control logic and register operations. The frequency of f_{CLC} is always identical to the system clock frequency f_{SPB}. The clock control register MSC_CLC makes it possible to enable/disable f_{CLC} under certain conditions.
- f_{MSC}
  This clock is the module clock that is used inside the MSC for baud rate generation of the serial upstream and downstream channel. The fractional divider register MSC_FDR controls the frequency of f_{MSC} and makes it possible to enable/disable it independent of f_{CLC}.
- f_{MSCA}
  This clock is the downstream clock which is used together with the ABRA block. In this case, the upstream baud rate can be fine-tuned with the fractional divider, and the downstream channel frequency is configured with an N-Divider.

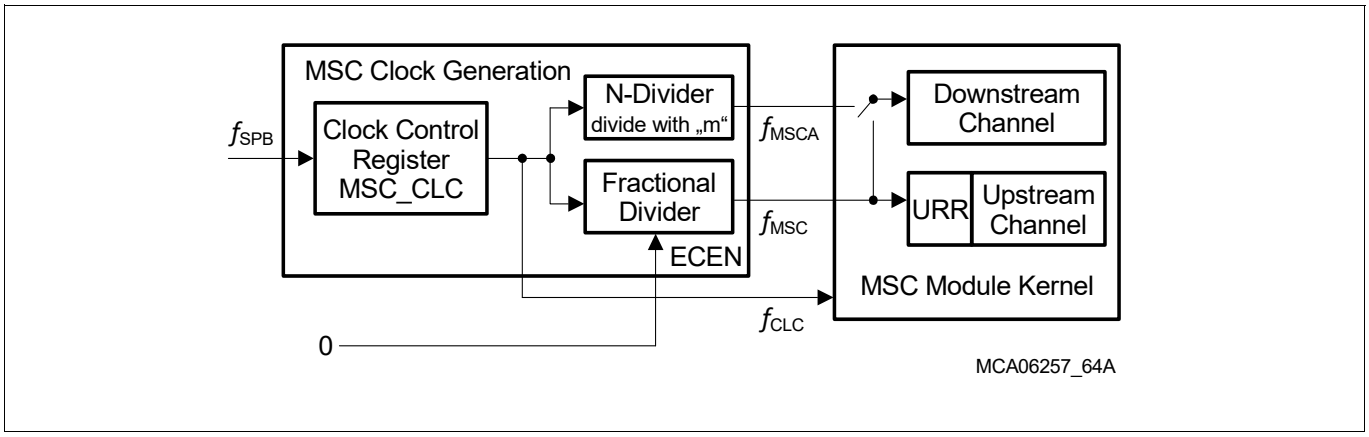


Figure 558 MSC Module Clock Generation

38.3.4.4.1 Clock Control Without the ABRA Block

Not using the ABRA block is the standard, compatibility use-case, which guarantees the full compatibility with the standard MSC module not containing the ABRA block. The clock generation is therefore identical with the standard MSC clock generation. The additional N-Divider (dividing with the division factor named “m”) is not used. Only the fractional divider is used (using the factor named “n” in the formulas below).

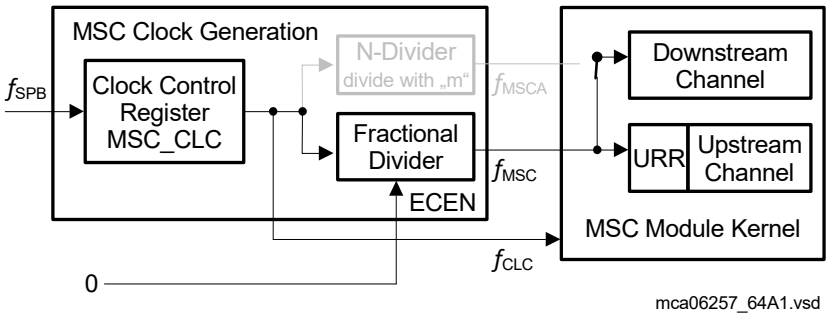


Figure 559 MSC Module Clock Generation

The following two formulas define the frequency of f_{MSC}:

$$f_{MSC} = f_{SPB} \times \frac{1}{n} \quad \text{with n = 1024 - MSC.FDR.STEP} \tag{38.4}$$

$$f_{MSC} = f_{SPB} \times \frac{n}{1024} \quad \text{with n = 0-1023} \tag{38.5}$$

Downstream Channel Baud Rate

As the clock signal FCL of the synchronous downstream channel is always half the frequency of f_{MSC}, the resulting downstream channel baud rate is defined by:

$$\text{Baud rate}_{MSC} = f_{SPB} \times \frac{1}{2 \times (1024 - \text{MSC.FDR.STEP})} \tag{38.6}$$

$$\text{Baud rate}_{MSC} = f_{SPB} \times \frac{\text{MSC.FDR.STEP}}{2 \times 1024} \tag{38.7}$$

Upstream Channel Baud Rate

The baud rate of the asynchronous upstream channel is derived from the module clock f_{MSC} by a programmable clock divider selected by bit field USR.URR (see also **Equation (38.3)** on **Page 24**). The divide factor DF can be at minimum 4 and at maximum 256.

$$\text{Baud rate}_{MSC} = f_{SPB} \times \frac{1}{DF \times (1024 - \text{MSC.FDR.STEP})} \tag{38.8}$$

$$\text{Baud rate}_{MSC} = f_{SPB} \times \frac{\text{MSC.FDR.STEP}}{DF \times 1024} \tag{38.9}$$

Equation (38.4), **Equation (38.6)**, and **Equation (38.8)** are valid for normal divider mode (FDR.DM = 01_B). **Equation (38.5)**, **Equation (38.7)**, and **Equation (38.9)** are valid for fractional divider mode (FDR.DM = 10_B).

38.3.4.4.2 Clock Control when using the ABRA Block

Using the ABRA block makes additional considerations regarding the baud-rate adjustments necessary.

An additional N-Divider **DSTE**.NDD is used for configuring the intermediate downstream baud rate the MSC module is using for writing to the ABRA block (dividing the f_{SPB} with the integer division factor named “m” in the formulas below). The final output downstream baud rate is configured in the ABRA block itself. The following restriction applies: the ABRA output downstream baud rate is between half and whole intermediate baud rate.

When ABRA block is active, the fractional divider **FDR**.STEP is used only for adjusting the upstream baud rate (using the factor named “n” in the formulas below).

Attention: ***The following restriction applies: the upstream baud rate must be less than one tenth of the f_{SPB}. See also Upstream Channel.***

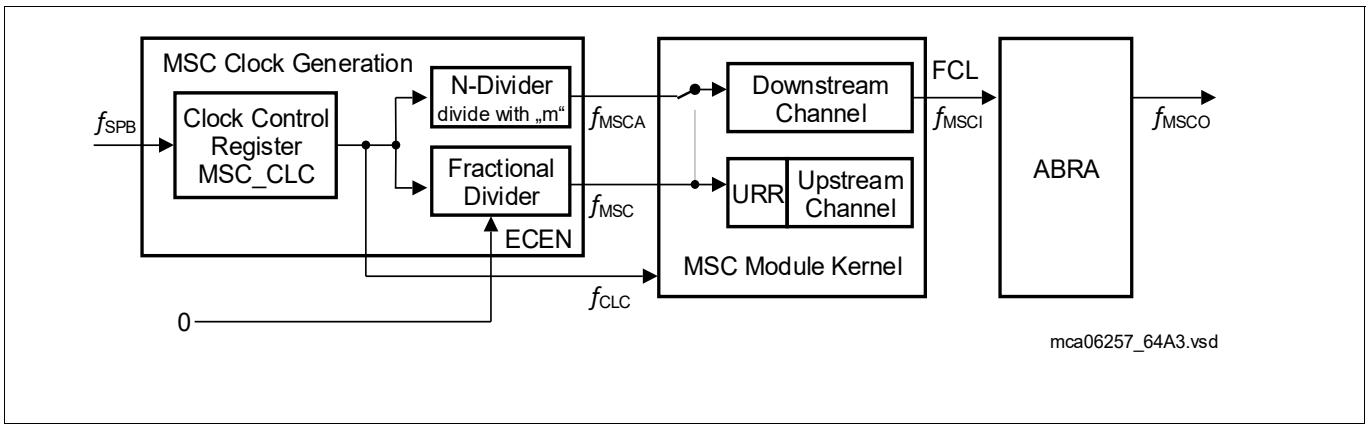


Figure 560 MSC Module Clock Generation

The following two formulas define the frequency of f_{MSC}:

$$f_{MSC} = f_{SPB} \times \frac{1}{n} \text{ with n = 1024 - MSC.FDR.STEP} \quad (38.10)$$

$$f_{MSC} = f_{SPB} \times \frac{n}{1024} \text{ with n = 0-1023} \quad (38.11)$$

The following formula defines the frequency of f_{MSCA}:

$$f_{MSCA} = f_{SPB} \times \frac{1}{m} \text{ with m = 1 - 16} \quad (38.12)$$

Intermediate Downstream Channel Baud Rate

When the ABRA block and the downstream N-divider are enabled with **ABC**.ABB, the intermediate clock signal FCL with frequency f_{MSCI} of the synchronous downstream channel going into the ABRA block is always half the frequency of f_{MSCA}:

$$\text{Baud rate}_{MSCI} = f_{SPB} \times \frac{1}{2 \times m} \quad (38.13)$$

The final output baud rate f_{MSCO} can be calculated as described in **Output Baud Rate**.

Upstream Channel Baud Rate

The baud rate of the asynchronous upstream channel is derived from the module clock f_{MSC} by a programmable clock divider selected by bit field USR.URR (see also **Equation (38.3)** on **Page 24**). The divide factor DF can be at minimum 4 and at maximum 256. Nevertheless, the minimum DF factor when using ABRA block should be 8. See the note on **Page 20**.

$$\text{Baud rate}_{MSC} = f_{SPB} \times \frac{1}{DF \times (1024 - MSC.FDR.STEP)} \quad (38.14)$$

$$\text{Baud rate}_{MSC} = f_{SPB} \times \frac{MSC.FDR.STEP}{DF \times 1024} \quad (38.15)$$

Equation (38.10), **Equation (38.11)**, and **Equation (38.14)** are valid for normal divider mode (FDR.DM = 01_B). **Equation (38.11)** and **Equation (38.15)** are valid for fractional divider mode (FDR.DM = 10_B).

Example for Setting the Fractional Divider

This example shows how to receive upstream frames generated from 40MHz downstream clock, when the module baud-rate frequency is 100MHz/2=50MHz. The fractional divider is used to adjust the FCL clock of 50MHz to FCL clock of 40MHz, that is, it multiplies with 4/5. The best approximation of 4/5 with x/1024 is for x=819, giving an error of 0.025%. So in this case, the optimal setting for FDR.STEP is 819. In the next stage of the divider cascade, the DF power-of-two-divider divides the clock further to the final baud rate, matching the baud rate generated by the MSC slave device.

38.3.4.4.3 Fractional Divider Register

Fractional Divider Register

The Fractional Divider Register controls the clock rate of the shift clock f_{MSC}.

FDR
Fractional Divider Register **(00C$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DISCLK	ENHW	0	0	0		RESULT									
rwh	rw	r	r	r		rh									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		0		0	0	STEP									
rw		r		r	r	rw									

Field	Bits	Type	Description
STEP	9:0	rw	**Step Value** Reload or addition value for RESULT.
DM	15:14	rw	**Divider Mode** DM selects normal or fractional divider mode.
RESULT	25:16	rh	**Result Value** Bit field for the addition result.
ENHW	30	rw	**Enable Hardware Clock Control** Controls operation of ECEN input and DISCLK bit.
DISCLK	31	rwh	**Disable Clock** Hardware controlled disable for f_{MSC} signal.
0	10, 11, 13:12, 27:26, 28, 29	r	**Reserved** Read as 0; shall be written with 0.

38.3.4.5 Port Control

Clock and data output lines are connected to dedicated LVDS differential output drivers. Some of the MSC module I/O lines are connected to I/O ports and therefore controlled in the port logic.

The following port control operations selections must be executed for these I/O lines:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for the outputs (PDR registers)

38.4 Registers

This section describes the kernel registers of the MSC module. All MSC kernel register names described in this section will be referenced in other parts of the User's Manual by the module name prefix "MSCx_".

All registers in the MSC address spaces are reset with the application reset (definition see SCU section "Reset Operation").

MSC Kernel Register Overview

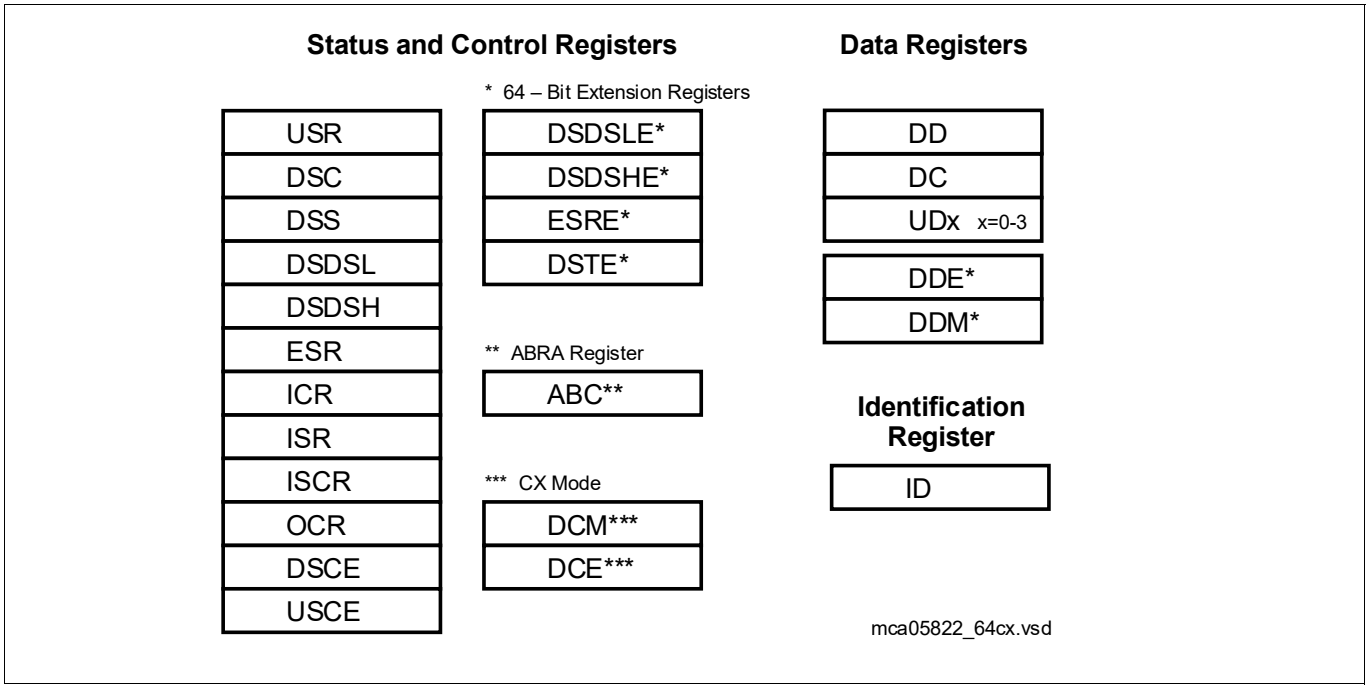


Figure 561 MSC Kernel Registers

Table 341 Register Address Space - MSC

Module	Base Address	End Address	Note
MSC0	$F0002600_H$	$F00026FF_H$	FPI slave interface
MSC1	$F0002700_H$	$F00027FF_H$	FPI slave interface
MSC2	$F0002800_H$	$F00028FF_H$	FPI slave interface
MSC3	$F0002900_H$	$F00029FF_H$	FPI slave interface

Table 342 Register Overview - MSC (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	000_H	U,SV	SV,E,P	Application Reset	43
ID	Module Identification Register	008_H	U,SV	BE	Application Reset	57
FDR	Fractional Divider Register	$00C_H$	U,SV	SV,P,E	Application Reset	53
USR	Upstream Status Register	010_H	U,SV	U,SV,P	Application Reset	58
DSC	Downstream Control Register	014_H	U,SV	U,SV,P	Application Reset	59
DSS	Downstream Status Register	018_H	U,SV	U,SV,P	Application Reset	64
DD	Downstream Data Register	$01C_H$	U,SV	U,SV,P	Application Reset	77
DC	Downstream Command Register	020_H	U,SV	U,SV,P	Application Reset	77
DSDSL	Downstream Select Data Source Low Register	024_H	U,SV	U,SV,P	Application Reset	65
DSDSH	Downstream Select Data Source High Register	028_H	U,SV	U,SV,P	Application Reset	66
ESR	Emergency Stop Register	$02C_H$	U,SV	U,SV,P	Application Reset	66
UDx	Upstream Data Register x	030_H+x* 4	U,SV	U,SV,P	Application Reset	78
ICR	Interrupt Control Register	040_H	U,SV	U,SV,P	Application Reset	67
ISR	Interrupt Status Register	044_H	U,SV	U,SV,P	Application Reset	68
ISC	Interrupt Set Clear Register	048_H	U,SV	U,SV,P	Application Reset	69
OCR	Output Control Register	$04C_H$	U,SV	U,SV,P	Application Reset	71
DSCE	Downstream Control Enhanced Register 1	058_H	U,SV	U,SV,P	Application Reset	61
USCE	Upstream Control Enhanced Register 1	$05C_H$	U,SV	U,SV,P	Application Reset	62
DSDSLE	Downstream Select Data Source Low Extension Register	060_H	U,SV	U,SV,P	Application Reset	73

Table 342 Register Overview - MSC (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
DSDSHE	Downstream Select Data Source High Extension Register	064_H	U,SV	U,SV,P	Application Reset	74
ESRE	Emergency Stop Extension Register	068_H	U,SV	U,SV,P	Application Reset	79
DSTE	Downstream Timing Extension Register	$06C_H$	U,SV	U,SV,P	Application Reset	74
DDM	Downstream Data Mirror Register	070_H	U,SV	U,SV,P	Application Reset	80
DDE	Downstream Data Extension Register	074_H	U,SV	U,SV,P	Application Reset	79
DCM	Downstream Command Mirror Register	078_H	U,SV	U,SV,P	Application Reset	80
DCE	Downstream Command Extension Register	$07C_H$	U,SV	U,SV,P	Application Reset	81
ABC	Asynchronous Block Configuration Register	080_H	U,SV	U,SV,P	Application Reset	82
OCS	OCDS Control and Status	$0E8_H$	U,SV	SV,P	Debug Reset	44
KRSTCLR	Kernel Reset Status Clear Register	$0EC_H$	U,SV	SV,E,P	Application Reset	47
KRST1	Kernel Reset Register 1	$0F0_H$	U,SV	SV,E,P	Application Reset	46
KRST0	Kernel Reset Register 0	$0F4_H$	U,SV	SV,E,P	Application Reset	46
ACCEN1	Access Enable Register 1	$0F8_H$	U,SV	SV,SE	Application Reset	45
ACCEN0	Access Enable Register 0	$0FC_H$	U,SV	SV,SE	Application Reset	44

Note:

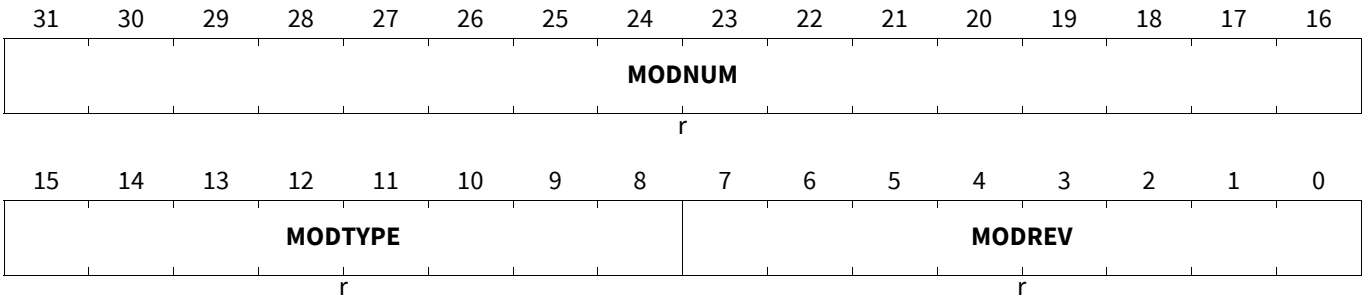
7. *All registers belong to Application Reset except OCS, which belongs to Debug Reset.*
8. *Read access to reserved addresses delivers bus error (BE) except 50H and 54H.*
9. *Write access to reserved addresses delivers bus error (BE).*

38.4.1 Module Identification Register

Module Identification Register

The MSC Module Identification Register ID contains read-only information about the module version.

ID
Module Identification Register (008$_H$) Application Reset Value: 0028 C01X$_H$



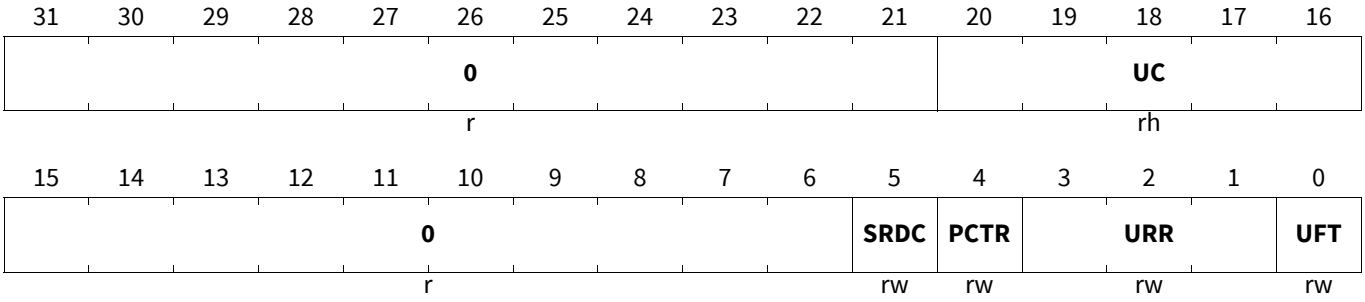
Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01_H (first revision).
MODTYPE	15:8	r	**Module Type** This bit field defines the module as a 32-bit module: $C0_H$
MODNUM	31:16	r	**Module Number Value** This bit field defines the module identification number for the MSC: 0028_H

38.4.2 Status and Control Registers

Upstream Status Register

The Upstream Status Register is used to configure the upstream channel data format, baud rate, and parity type. It also provides the status information of the upstream counter (UC).

USR
Upstream Status Register (010$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
UFT	0	rw	**Upstream Channel Frame Type** This bit determines the frame type used by the upstream channel for data reception. 0_B 12-bit upstream frame selected 1_B 16-bit upstream frame selected (with 4-bit address field)
URR	3:1	rw	**Upstream Channel Receiving Rate** This bit field determines the baud rate for the upstream channel. 000_B Upstream channel disabled; no reception is possible 001_B Baud rate = $f_{MSC}/4$ 010_B Baud rate = $f_{MSC}/8$ 011_B Baud rate = $f_{MSC}/16$ 100_B Baud rate = $f_{MSC}/32$ 101_B Baud rate = $f_{MSC}/64$ 110_B Baud rate = $f_{MSC}/128$ 111_B Baud rate = $f_{MSC}/256$
PCTR	4	rw	**Parity Control** This bit determines the parity mode used by the upstream channel for data reception. 0_B Even parity mode is selected. A parity bit is set on an odd number of 1s in the serial address/data stream. 1_B Odd parity mode is selected. A parity bit is set on an even number of 1s in the serial address/data stream.
SRDC	5	rw	**Service Request Delay Control** This bit determines the additional delay inserted by the upstream channel when triggering the receive interrupt. Only the hardware generated interrupt can be delayed. The software generated interrupt by writing the ISC.SURDI bit is never delayed. 0_B No delay 1_B Delay of 1 bit time

Field	Bits	Type	Description
UC	20:16	rh	**Upstream Counter** This bit field indicates the content of the upstream counter that counts the bits during upstream channel reception.
0	15:6, 31:21	r	**Reserved** Read as 0; shall be written with 0.

Downstream Control Register

The Downstream Control Register is used to control the operation mode and frame layout of the downstream channel transmission. It also contains the two pending status bits.

Note: The "rw" bits in the DSC register are buffered in a shadow buffer at the start of a corresponding frame transmission.

Note: If the asynchronous mode (ABRA=1) is enabled and no data-bit (zero data-bits) in a data frame is configured (DSC.NDBL=DSC.NDBH=00000), then the first select-bit must be set to active (DSC.ENSELL=1).

DSC
Downstream Control Register (014$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**PPD**					**0**		**NBC**					
r			rw					r		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSDIS	**ENSELH**	**ENSELL**	**NDBH**					**NDBL**					**DP**	**CP**	**TM**
rh	rw	rw	rw					rw					rh	rh	rw

Field	Bits	Type	Description
TM	0	rw	**Transmission Mode** This bit selects the transmission mode of the downstream channel. 0_B Triggered Mode selected 1_B Data Repetition Mode selected
CP	1	rh	**Command Pending** This bit is set when the downstream command register DC is written. CP is cleared when the first bit of the related command frame is sent out.
DP	2	rh	**Data Pending** In Triggered Mode, this bit is set when the set data pending bit ISC.SDP is set by software. In Data Repetition Mode, this bit is set by hardware at the last passive time frame. At the start of the data frame, DP is cleared by hardware.

Field	Bits	Type	Description
NDBL	7:3	rw	**Number of SRL Bits Shifted at Data Frames** NDBL determines the number of shift register low part (SRL) bits that are shifted out on SO during a data frame. Other bit combinations are reserved; do not use these bit combinations. 00_H No SRL bit shifted 01_H SRL[0:0] shifted … 10_H SRL[15:0] shifted **others**, reserved
NDBH	12:8	rw	**Number of SRH Bits Shifted at Data Frames** NDBH determines the number of shift register high part (SRH) bits that are shifted out on SO during a data frame. Other bit combinations are reserved; do not use these bit combinations. 00_H No SRH bit shifted; no selection bit is generated, the SRH active phase is completely skipped. 01_H SRH[0:0] shifted … 10_H SRH[15:0] shifted **others**, reserved
ENSELL	13	rw	**Enable SRL Active Phase Selection Bit** This bit determines whether a low level selection bit is inserted at the beginning of a data frame's SRL active phase. 0_B No selection bit inserted. 1_B Low level selection bit inserted.
ENSELH	14	rw	**Enable SRH Active Phase Selection Bit** This bit determines whether a low level selection bit is inserted at the beginning of a data frame's SRH active phase. 0_B No selection bit inserted. 1_B Low level selection bit inserted.
DSDIS	15	rh	**Downstream Disable** This bit indicates the state of the downstream channel operation. 0_B The downstream channel is enabled. A frame transmission can take place (Triggered Mode) or takes place (Data Repetition Mode). 1_B Downstream Counter becomes disabled. No new frame transmission is started. A running frame transmission is always completed.
NBC	21:16	rw	**Number of Bits Shifted at Command Frames** This bit field determines how many bits of the SRL/SRH shift registers are shifted out during transmission of a command frame. 00_H No bit shifted 01_H SRL[0:0] shifted … 10_H SRL[15:0] shifted 11_H SRL[15:0] and SRH[0:0] shifted … 20_H SRL[15:0] and SRH[15:0] shifted **others**, reserved

Field	Bits	Type	Description
PPD	28:24	rw	**Passive Phase Length at Data Frames** This bit field determines the length of the passive phase of a data frame. 00_H Passive phase length is 2 x t_FCL 01_H Passive phase length is 2 x t_FCL 02_H Passive phase length is 2 x t_FCL … $1F_H$ Passive phase length is 31 x t_FCL
0	23:22, 31:29	r	**Reserved** Read as 0; shall be written with 0.

Downstream Control Enhanced Register 1

The Downstream Control Enhanced Register is used to control the enhanced operation mode and frame layout of the downstream channel transmission.

DSCE

Downstream Control Enhanced Register 1 (058_H) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CDCM	**INJPOSP1**						**INJENP1**	**0**	**INJPOSP0**						**INJENP0**
rw	rw						rw	r	rw						rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCF	**EXEN**	**0**												**NDBLE**	**NDBHE**
rh	rw	r												rw	rw

Field	Bits	Type	Description
NDBHE	0	rw	**Number of SRH Bits Shifted at Data Frames Extension** Additional MSB bit extension of the NDBH bit field. Adds 16 to the resulting NDBH value if set.
NDBLE	1	rw	**Number of SRL Bits Shifted at Data Frames Extension** Additional MSB bit extension of the NDBL bit field. Adds 16 to the resulting NDBL value if set.
EXEN	14	rw	**Extension Enable** Enables the extension bit fields. 0_B disabled 1_B enabled
CCF	15	rh	**Command-Command Flag** This bit flags that a second command frame has been written without data frame to be sent in between. It is active only if CDCM=1. Otherwise it is always low. 0_B no second command 1_B second command frame pending

Field	Bits	Type	Description
INJENP0	16	rw	**Injection Enable of the Pin 0 Signal** This bit selects if an external signal is injected in a data frame. 0_B Disabled 1_B Enabled
INJPOSP0	22:17	rw	**Injection Position of the Pin 0 Signal** This bit selects the position of the injected external one bit signal, at the bit position in range of 0 to 63 in the data frame. If both PIN0POS and PIN1POS point to a same bit, PIN0POS has the higher priority, that is PIN0 level will be visible in the data frame.
INJENP1	24	rw	**Injection Enable of the Pin 1 Signal** This bit selects if an external signal is injected in a data frame. 0_B Disabled 1_B Enabled
INJPOSP1	30:25	rw	**Injection Position of the Pin 1 Signal** This bit selects the position of the injected external one bit signal, at the bit position in range of 0 to 63 in the data frame.
CDCM	31	rw	**Command-Data-Command in Data Repetition Mode** This bit selects if a data frame is automatically inserted between two consecutive command frame requests. 0_B Automatic data insertion disabled 1_B Automatic data insertion enabled
0	13:2, 23	r	**Reserved** Read as 0; shall be written with 0.

Upstream Control Enhanced Register 1

The Upstream Control Enhanced Register is used to control the upstream channel time-out feature.

USCE

Upstream Control Enhanced Register 1 ($05C_H$) Application Reset Value: 0000 00FF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USTOIP		UTASR	0	USTS	USTC	USTF	USTO EN	USTOVAL				USTOPRE			
rw		rw	r	w	w	rh	rw	rw				rw			

Field	Bits	Type	Description
USTOPRE	3:0	rw	**Upstream Time-out Prescaler** Prescaler for the upstream time-out limit. Expressed in upstream bit times. 2^n divider in the range of 1 to 16384. Write to this bit field triggers new start of the watchdog timer. 0_H 1 … E_H 16384 F_H reserved
USTOVAL	7:4	rw	**Upstream Time-out Value** Upstream time-out value for the N-divider in the range of 1 to 16, expressed in number of upstream bit time lengths (the upstream bit time is the reciprocal value of the upstream baud rate). Time-out = BitTime * $2^{(USTOPRE+1)}$ * (USTOVAL+1). Write to this bit field triggers new start of the watchdog timer.
USTOEN	8	rw	**Upstream Time-out Interrupt Enable** Enable bit for the upstream time-out interrupt. The Time-out counter runs continuously independently of the USTOEN bit. The USTOEN enables only the interrupt generation. 0_B Disabled 1_B Enabled
USTF	9	rh	**Upstream Time-out Flag** Signals an upstream timer event. If set by software, the interrupt is also triggered, if enabled. The watchdog timer runs continuously and sets the USTF flag at overflow, independently from the enable bit. Therefore this bit must be cleared with the same write access which enables the interrupt. 0_B Disabled 1_B Enabled
USTC	10	w	**Upstream Time-out Clear** Clears the USTF flag. Write only bit, reads as 0. 0_B No action 1_B Clear
USTS	11	w	**Upstream Time-out Set** Sets the USTF flag. Write only bit, reads as 0. 0_B No action 1_B Set
UTASR	13	rw	**Upstream Time-out Alternate Service Request** Selects if the interrupt signal is routed to the alternate service request node. See **Figure 547**. 0_B SR Multiplexer selected 1_B Alternate Service Request Node (SR4) selected

Field	Bits	Type	Description
USTOIP	15:14	rw	**Upstream Time-out Interrupt Node Pointer** USTOIP selects the service request output line SRn (n = 0-3) for the time-out interrupt. 00_B Service request output SR0 selected 01_B Service request output SR1 selected 10_B Service request output SR2 selected 11_B Service request output SR3 selected
0	12, 31:16	r	**Reserved** Read as 0; shall be written with 0.

Downstream Status Register

The Downstream Status Register DSS contains counter bit fields, status bits, and indicates the number of passive time frames.

DSS

Downstream Status Register (018$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						CFA	DFA	DC							
r						rh	rh	rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				NPTF				0				PFC			
r				rw				r				rh			

Field	Bits	Type	Description
PFC	3:0	rh	**Passive Time Frame Counter** In Data Repetition Mode, this bit field indicates the count of passive time frames that are currently transmitted. In Triggered Mode PFC remains at 0000_B. 0_H Data frame is transmitted. 1_H First passive time frame is transmitted. 2_H 2th passive time frame is transmitted. … F_H 15th passive time frame is transmitted.
NPTF	11:8	rw	**Number Of Passive Time Frames** This bit field indicates the number of passive time frames that are inserted in Data Repetition Mode between two data frames. 0_H No passive time frame inserted. 1_H 1 passive time frame inserted. 2_H 2 passive time frames inserted. … F_H 15 passive time frames inserted.

Field	Bits	Type	Description
DC	23:16	rh	**Downstream Counter** This bit field indicates the number of downstream shift clock periods that have been elapsed since the start of the current frame. DC is reset at the end of a downstream frame. 00_H No shift clock elapsed (after counter reset). 01_H 1 shift clock elapsed. 02_H 2 shift clocks elapsed. … $7F_H$ 127 shift clocks elapsed.
DFA	24	rh	**Data Frame Active** This bit indicates if a data frame is currently sent out, active phase only. 0_B No data frame is currently sent out. 1_B A data frame is currently sent out.
CFA	25	rh	**Command Frame Active** This bit indicates if a command frame is currently sent out, active phase only. 0_B No command frame is currently sent out. 1_B A command frame is currently sent out.
0	7:4, 15:12, 31:26	r	**Reserved** Read as 0; shall be written with 0.

Downstream Select Data Source Low Register

The bit fields of the Downstream Select Data Source Low Register DSDSL determine the data source for each bit in shift register SRL.

DSDSL

Downstream Select Data Source Low Register (024$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SL15		**SL14**		**SL13**		**SL12**		**SL11**		**SL10**		**SL9**		**SL8**	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SL7		**SL6**		**SL5**		**SL4**		**SL3**		**SL2**		**SL1**		**SL0**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
SLx (x=0-15)	2*x+1:2*x	rw	**Select Source for SRL** SLx determines which data source is used for the shift register bit SRL[x] for data frame transmission. 00_B SRL[x] is taken from data register DD.DDL[x]. 01_B Reserved. 10_B SRL[x] is taken from the ALTINL input line x. 11_B SRL[x] is taken from the ALTINL input line x in inverted state.

Downstream Select Data Source High Register

The bit fields of the Downstream Select Data Source High Register DSDSH determine the data source for each bit in shift register SRH.

DSDSH

Downstream Select Data Source High Register (028$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SH15		SH14		SH13		SH12		SH11		SH10		SH9		SH8	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SH7		SH6		SH5		SH4		SH3		SH2		SH1		SH0	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
SHx (x=0-15)	2*x+1:2*x	rw	**Select Source for SRH** SHx determines which data source is used for the shift register bit SRH[x] during data frame transmission. 00_B SRH[x] is taken from data register DD.DDH[x]. 01_B Reserved. 10_B SRH[x] is taken from the ALTINH input line x. 11_B SRH[x] is taken from the ALTINH input line x in inverted state.

Emergency Stop Register

The Emergency Stop Register ESR determines which bits of SRL and SRH are enabled for emergency operation.

ESR

Emergency Stop Register (02C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENH15	ENH14	ENH13	ENH12	ENH11	ENH10	ENH9	ENH8	ENH7	ENH6	ENH5	ENH4	ENH3	ENH2	ENH1	ENH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENL15	ENL14	ENL13	ENL12	ENL11	ENL10	ENL9	ENL8	ENL7	ENL6	ENL5	ENL4	ENL3	ENL2	ENL1	ENL0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENLx (x=0-15)	x	rw	**Emergency Stop Enable for Bit x in SRL** This bit enables the emergency stop feature selectively for each SRL bit. If the emergency stop condition is met and enabled (ENLx = 1), the SRL[x] bit of the data register DD.DDL[x] is used for the shift register load operation. 0_B Emergency stop feature for bit SRL[x] is disabled. 1_B The emergency stop feature for bit SRL[x] is enabled.

Field	Bits	Type	Description
ENHx (x=0-15)	x+16	rw	**Emergency Stop Enable for Bit x in SRH** This bit enables the emergency stop feature selectively for each SRH bit. If the emergency stop condition is met and enabled (ENHx = 1), the SRH[x] bit of the data register DD.DDH[x] is used for the shift register load operation. 0_B Emergency stop feature for bit SRH[x] is disabled. 1_B The emergency stop feature for bit SRH[x] is enabled.

Interrupt Control Register

The Interrupt Control Register ICR holds the interrupt enable bits and interrupt pointers of all four MSC interrupts.

ICR
Interrupt Control Register **(040_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDIE		RDIP		TFIE	0	TFIP		ECIE	0	ECIP		EDIE		EDIP	
rw		rw		rw	r	rw		rw	r	rw		rw		rw	

Field	Bits	Type	Description
EDIP	1:0	rw	**Data Frame Interrupt Node Pointer** EDIP selects the service request output line SRn (n = 0-3) for the data frame interrupt. 00_B Service request output SR0 selected 01_B Service request output SR1 selected 10_B Service request output SR2 selected 11_B Service request output SR3 selected
EDIE	3:2	rw	**Data Frame Interrupt Enable** This bit field determines the enable conditions for the data frame interrupt. 00_B Interrupt generation disabled 01_B An interrupt is generated when the last data bit has been shifted out. 10_B An interrupt is generated when the first data bit has been shifted out, but only if DSC.NDBL is not equal 00000_B. This means, at least one SRL bit must be shifted out for the first data bit shifted interrupt to become active. 11_B Interrupt generation disabled

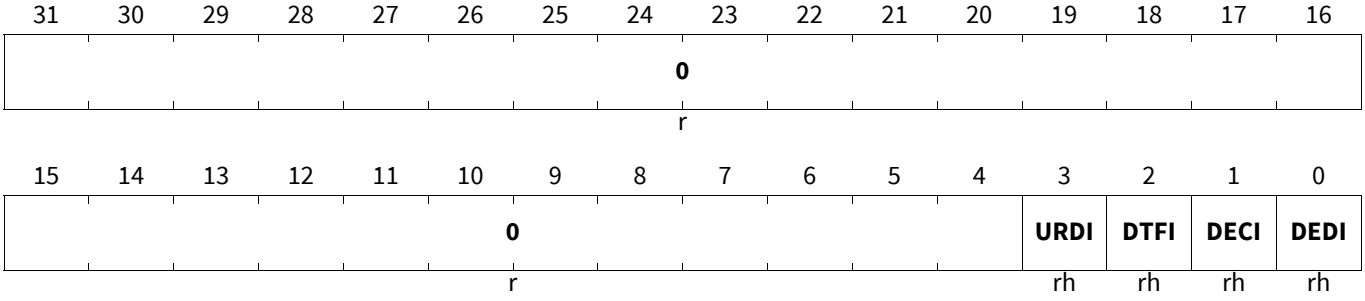
Field	Bits	Type	Description
ECIP	5:4	rw	**Command Frame Interrupt Node Pointer** ECIP selects the service request output line SRn (n = 0-3) for the command frame interrupt. 00_B Service request output SR0 selected 01_B Service request output SR1 selected 10_B Service request output SR2 selected 11_B Service request output SR3 selected
ECIE	7	rw	**Command Frame Interrupt Enable** This bit enables the command frame interrupt. 0_B Interrupt generation disabled. 1_B Interrupt generation enabled.
TFIP	9:8	rw	**Time Frame Interrupt Pointer** TFIP selects the service request output line SRn (n = 0-3) for the time frame interrupt. 00_B Service request output SR0 selected 01_B Service request output SR1 selected 10_B Service request output SR2 selected 11_B Service request output SR3 selected
TFIE	11	rw	**Time Frame Interrupt Enable** This bit enables the time frame interrupt. 0_B Interrupt generation disabled. 1_B Interrupt generation enabled.
RDIP	13:12	rw	**Receive Data Interrupt Pointer** RDIP selects the service request output line SRn (n = 0-3) for the receive data interrupt. 00_B Service request output SR0 selected 01_B Service request output SR1 selected 10_B Service request output SR2 selected 11_B Service request output SR3 selected
RDIE	15:14	rw	**Receive Data Interrupt Enable** This bit field determines the enable conditions for the receive data interrupt. 00_B Interrupt generation disabled. 01_B An interrupt is generated when data is received and written into the upstream data registers UDx (x = 0-3). 10_B An interrupt is generated as with RDIE = 01_B but only if the received data is not equal to 00_H. 11_B An interrupt is generated when data is received and written into register UD3.
0	6, 10, 31:16	r	**Reserved** Read as 0; shall be written with 0.

Interrupt Status Register

The Interrupt Status Register ISR holds the interrupt status flags that indicate an interrupt occurrence in downstream and upstream channels.

ISR

Interrupt Status Register **(044_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DEDI	0	rh	**Data Frame Interrupt Flag** This flag is always set by hardware when a downstream channel data frame interrupt is generated. DEDI can be set or cleared by software when writing to register ISC with the appropriate bits ISC.SDEDI or ISC.CDEDI set.
DECI	1	rh	**Command Frame Interrupt Flag** This flag is always set by hardware when a downstream channel command frame interrupt is generated, whether or not it is enabled. DECI can be set or cleared by software when writing to register ISC with the appropriate bits SDECI or CDECI set.
DTFI	2	rh	**Time Frame Interrupt Flag** This flag is always set by hardware when a downstream channel time frame interrupt is generated, whether or not it is enabled. DTFI can be set or cleared by software when writing to register ISC with the appropriate bits SDTFI or CDTFI set.
URDI	3	rh	**Receive Data Interrupt Flag** This flag is always set by hardware when an upstream channel receive data interrupt is generated, whether or not it is enabled. URDI can be set or cleared by software when writing to register ISC with the appropriate bits SURDI or CURDI set.
0	31:4	r	**Reserved** Read as 0; shall be written with 0.

Interrupt Set Clear Register

The Interrupt Set Clear Register ISC is used to set or clear the MSC interrupt flags located in the Interrupt Status Register ISR. Reading ISC always returns $0000\ 0000_H$.

Note: When the ISC register is written with both bits (set and clear bit) for a specific interrupt flag, the clear operation takes place and the set operation is ignored.

ISC

Interrupt Set Clear Register **(048_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									SDDIS	SCP	SDP	SURDI	SDTFI	SDECI	SDEDI
r									w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									CDDIS	CCP	CDP	CURDI	CDTFI	CDECI	CDEDI
r									w	w	w	w	w	w	w

Field	Bits	Type	Description
CDEDI	0	w	**Clear DEDI Flag** 0_B No operation 1_B Bit ISR.DEDI is cleared.
CDECI	1	w	**Clear DECI Flag** 0_B No operation 1_B Bit ISR.DECI is cleared.
CDTFI	2	w	**Clear DTFI Flag** 0_B No operation 1_B Bit ISR.DTFI is cleared.
CURDI	3	w	**Clear URDI Flag** 0_B No operation 1_B Bit ISR.URDI is cleared.
CDP	4	w	**Clear DP Flag** 0_B No operation 1_B Bit DSC.DP is cleared.
CCP	5	w	**Clear CP Flag** 0_B No operation 1_B Bit DSC.CP is cleared.
CDDIS	6	w	**Clear DSDIS Flag** 0_B No operation 1_B Bit DSC.DSDIS is cleared.
SDEDI	16	w	**Set DEDI Flag** 0_B No operation 1_B Bit ISR.DEDI is set.

Field	Bits	Type	Description
SDECI	17	w	**Set DECI Flag** 0_B No operation 1_B Bit ISR.DECI is set.
SDTFI	18	w	**Set DTFI Flag** 0_B No operation 1_B Bit ISR.DTFI is set.
SURDI	19	w	**Set URDI Flag** 0_B No operation 1_B Bit ISR.URDI is set.
SDP	20	w	**Set DP Bit** 0_B No effect 1_B Bit DSC.DP is set.
SCP	21	w	**Set CP Flag** 0_B No operation 1_B Bit DSC.CP is set.
SDDIS	22	w	**Set DSDIS Flag** 0_B No operation 1_B Bit DSC.DSDIS is set.
0	15:7, 31:23	r	**Reserved** Read as 0; shall be written with 0.

Output Control Register

The Output Control Register OCR determines the MSC input/output signal polarities, the chip select output signal assignment, and the serial output clock generation.

OCR

Output Control Register **($04C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													SDISEL		
r													rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CSC		CSH		CSL		CLKCTRL	0				ILP	CSLP	SLP	CLP
r	rw		rw		rw		rw	r				rw	rw	rw	rw

Field	Bits	Type	Description
CLP	0	rw	**FCLP Line Polarity** 0_B FCLP and FCL signal polarity is identical. FCLN signal has inverted FCL signal polarity. 1_B FCLP signal has inverted FCL signal polarity. FCLN and FCL signal polarities are identical.
SLP	1	rw	**SOP Line Polarity** 0_B SOP and SO signal polarity is identical. SON signal has inverted SO signal polarity. 1_B SOP signal has inverted SO signal polarity. SON and SO signal polarities are identical.
CSLP	2	rw	**Chip Selection Lines Polarity** Bit CSLP is buffered during a frame transmission. This means that any change of CSLP becomes valid first with the start of the next frame transmission. 0_B EN[3:0] and ENL, ENH, ENC signal polarities are identical (high active). 1_B EN[3:0] signal polarities are inverted (low active) to the ENL, ENH, ENC signal polarities.
ILP	3	rw	**SDI Line Polarity** 0_B SDI and SI signal polarities are identical. 1_B SDI and SI signal polarities are inverted.
CLKCTRL	8	rw	**Clock Control** This bit determines the activation of clock output FCL. 0_B FCL is activated only during the active phases of data or command frames (not during passive time frames). 1_B FCL is always active whether or not a downstream frame is currently transmitted.
CSL	10:9	rw	**Chip Enable Selection for ENL** This bit field selects the chip enable output ENx that becomes active during the SRL active phase (ENL = 1) of a data frame. The active level of ENx is defined by bit CSLP. 00_B EN0 line is selected for ENL. 01_B EN1 line is selected for ENL. 10_B EN2 line is selected for ENL. 11_B EN3 line is selected for ENL.

Field	Bits	Type	Description
CSH	12:11	rw	**Chip Enable Selection for ENH** This bit field selects the chip enable output ENx that becomes active during the SRH active phase (ENH = 1) of a data frame. The active level of ENx is defined by bit CSLP. 00_B EN0 line is selected for ENH. 01_B EN1 line is selected for ENH. 10_B EN2 line is selected for ENH. 11_B EN3 line is selected for ENH.
CSC	14:13	rw	**Chip Enable Selection for ENC** This bit field selects the chip enable output ENx that becomes active during the active phase (ENC = 1) of a command frame. The active level of ENx is defined by bit CSLP. 00_B EN0 line is selected for ENC. 01_B EN1 line is selected for ENC. 10_B EN2 line is selected for ENC. 11_B EN3 line is selected for ENC.
SDISEL	18:16	rw	**Serial Data Input Selection** This bit field selects the source for the serial data input SDI of the upstream channel. 000_B SDI0 input is selected for SDI. 001_B SDI1 input is selected for SDI. 010_B SDI2 input is selected for SDI. 011_B SDI3 input is selected for SDI. 100_B SDI4 input is selected for SDI. 101_B SDI5 input is selected for SDI. 110_B SDI6 input is selected for SDI. 111_B SDI7 input is selected for SDI.
0	7:4, 15, 31:19	r	**Reserved** Read as 0; shall be written with 0.

Downstream Select Data Source Low Extension Register

The bit fields of the Downstream Select Data Source Low Extension Register DSDSLE determine the data source for the upper 16 bits in the shift register SRL.

DSDSLE

Downstream Select Data Source Low Extension Register(060_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SL31		**SL30**		**SL29**		**SL28**		**SL27**		**SL26**		**SL25**		**SL24**	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SL23		**SL22**		**SL21**		**SL20**		**SL19**		**SL18**		**SL17**		**SL16**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
SLx (x=16-31)	2*x-31:2*x-32	rw	**Select Source for SRL** SLx determines which data source is used for the shift register bit SRL[x] during data frame transmission. 00_B SRL[x] is taken from data register DD.DDL[x]. 01_B Reserved. 10_B SRL[x] is taken from the ALTINLE[15:0] input line x-16. 11_B SRL[x] is taken from the ALTINLE[15:0] input line x-16 in inverted state.

Downstream Select Data Source High Extension Register

The bit fields of the Downstream Select Data Source High Extension Register DSDSHE determine the data source for the upper 16 bit in the shift register SRH.

DSDSHE

Downstream Select Data Source High Extension Register(064_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SH31		**SH30**		**SH29**		**SH28**		**SH27**		**SH26**		**SH25**		**SH24**	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SH23		**SH22**		**SH21**		**SH20**		**SH19**		**SH18**		**SH17**		**SH16**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
SHx (x=16-31)	2*x-31:2*x-32	rw	**Select Source for SRH** SHx determines which data source is used for the shift register bit SRH[x] during data frame transmission. 00_B SRH[x] is taken from data register DD.DDH[x]. 01_B Reserved. 10_B SRH[x] is taken from the ALTINHE[15:0] input line x-16. 11_B SRH[x] is taken from the ALTINHE[15:0] input line x-16 in inverted state.

Downstream Timing Extension Register

The Downstream Timing Extension Register is used to control the enhanced operation mode and frame layout of the downstream channel transmission.

DSTE

Downstream Timing Extension Register (06C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UL1	0		CX	0											FM
w	r		rw	r											rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			PPCEM	NDD				PPCE						PPDE	
r			rw	rw				rw						rw	

Field	Bits	Type	Description
PPDE	1:0	rw	**Passive Phase Length at Data Frames Extension** Additional MSB bits extension of the PPD bit field.
PPCE	7:2	rw	**Passive Phase Length at Control Frames Extension** Additional MSB bits extension of the fixed length of 2 of the command frames passive phase. The final length is 2 + PPCE resulting in a range of 2 to 65. 00_H 2 … $3F_H$ 65
NDD	11:8	rw	**N Divider Downstream** Defines the division ratio in the range of 1 to 16. 0_H 1 … F_H 16
PPCEM	12	rw	**PPCE Extension Bit on the MSB Side** This bit is the MSB extension bit for the PPCE bit field, extending command frame passive phase to the value of 127. The values of 128 and 129 are not allowed. 0_B MSB value of 0 1_B MSB value of 1
FM	16	rw	**Fast Mode** Activates the fast mode and writing to the ABRA FIFO with 100 MBaud input baud rate, in order to provide 80MBaud output baud rate. It is also recommended for baud rate of 26,67 MBaud, as defined in the **Table 340**. FM=0 is the compatibility setting with the previous generations of the MSC. 0_B Fast Mode deactivated 1_B Fast Mode activated
CX	28	rw	**Command Extension Mode** Activates 64 bit command frame feature. 0_B CX mode deactivated 1_B CX mode activated

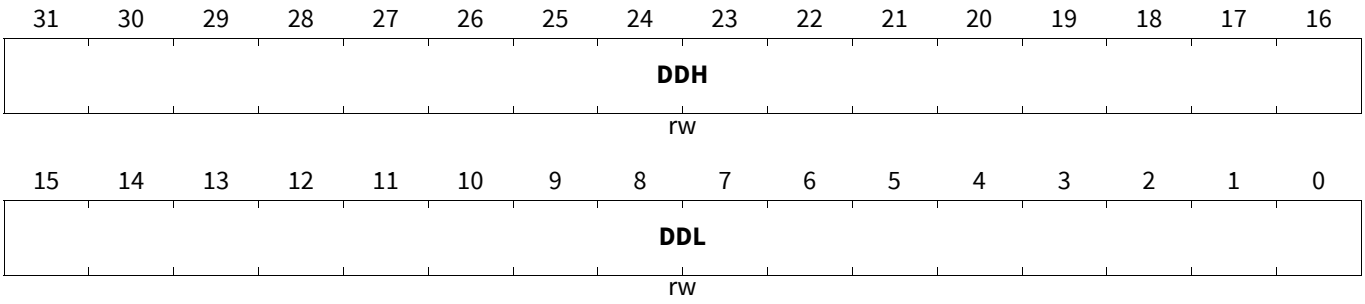
Field	Bits	Type	Description
UL1	31	w	**Unlock CX and FM for one write access** Write access with UL1 = 1 unlocks the CX and FM bit fields for the current write access. Write accesses to these bit fields with UL1 = 0 do not have any effect. Returns zero on read.
0	15:13, 27:17, 30:29	r	**Reserved** Read as 0; shall be written with 0.

38.4.3 Data Registers

Downstream Data Register

The Downstream Data Register DD contains data to be transmitted during data frames.

DD
Downstream Data Register **($01C_H$)** **Application Reset Value: $0000\ 0000_H$**

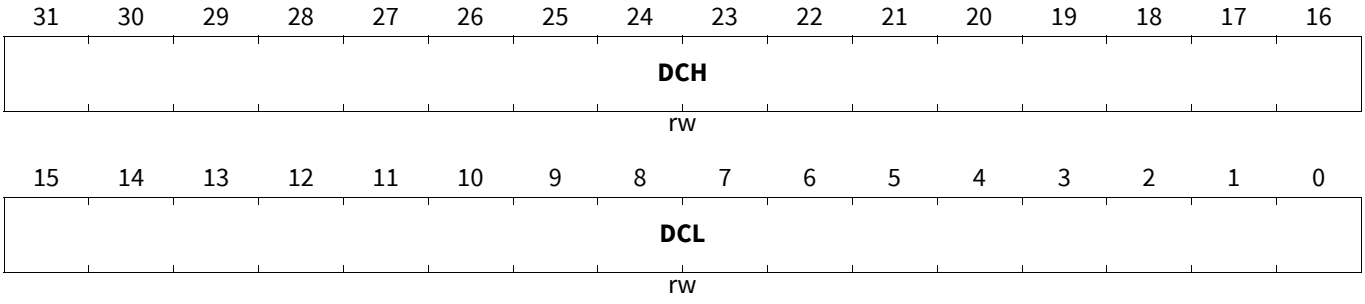


Field	Bits	Type	Description
DDL	15:0	rw	**Downstream Data for SRL Shift Register** Contains the data bits to be transmitted during the SRL active phase of a data frame.
DDH	31:16	rw	**Downstream Data for SRH Shift Register** Contains the data bits to be transmitted during the SRH active phase of a data frame.

Downstream Command Register

The Downstream Command Register DC contains command information to be transmitted during command frames. Every write access to the DC register triggers a transmission of a command frame.

DC
Downstream Command Register **(020_H)** **Application Reset Value: $0000\ 0000_H$**



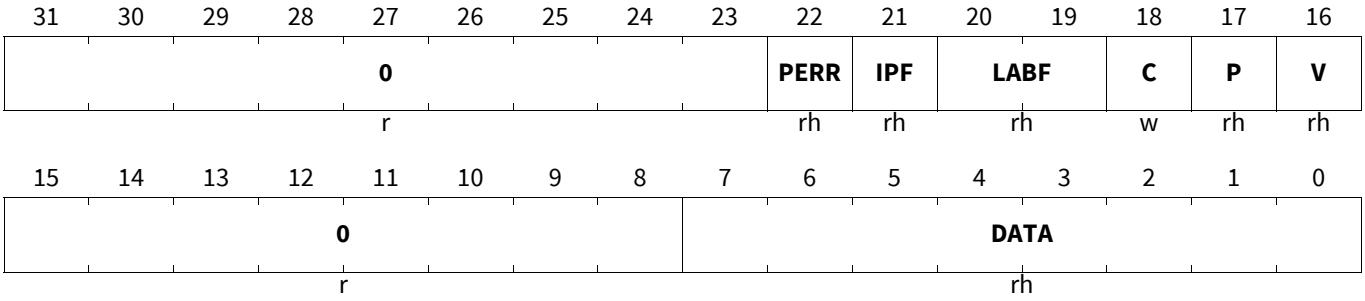
Field	Bits	Type	Description
DCL	15:0	rw	**Downstream Command for SRL Shift Register** Contains the data bits to be transmitted during the SRL active phase of a command frame.
DCH	31:16	rw	**Downstream Command for SRH Shift Register** Contains the data bits to be transmitted during the SRH active phase of a command frame.

Upstream Data Register x

The four Upstream Data Registers UDx store the content (data, addresses, received and calculated parity bit, parity error bit) of a received upstream channel data frame.

UDx (x=0-3)

Upstream Data Register x **(030_H+x*4)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DATA	7:0	rh	**Received Data** This bit field contains the 8-bit receive data.
V	16	rh	**Valid Bit** This bit is set by hardware when the received data is written to UDx. Writing bit C = 1 clears V. If hardware setting and software clearing of the valid bit occur simultaneously, bit V will be cleared.
P	17	rh	**Parity Bit** This flag contains the parity bit that has been received with the data frame.
C	18	w	**Clear Bit** C is always read as 0. 0_B No operation. 1_B Bit V is cleared.
LABF	20:19	rh	**Lower Address Bit Field** This bit field contains the two address bits A[1:0] of the 4-bit address field (16-bit data frame). If 12-bit data frame is selected, LABF is always set to 00_B.
IPF	21	rh	**Internal Parity Flag** This bit contains the parity bit that has been calculated in the MSC during data frame reception.
PERR	22	rh	**Parity Error** This bit indicates if a start bit error, parity error, or stop bit error occurred during frame reception. 0_B No error detected. 1_B Error detected.
0	15:8, 31:23	r	**Reserved** Read as 0; shall be written with 0.

38.4.4 Extension Registers

This sub-chapter describes the registers related to 64-bit extension of the MSC module.

Emergency Stop Extension Register

The Emergency Stop Extension Register ESRE determines which bits of SRL and SRH are enabled for emergency operation.

ESRE

Emergency Stop Extension Register (068$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENH31	**ENH30**	**ENH29**	**ENH28**	**ENH27**	**ENH26**	**ENH25**	**ENH24**	**ENH23**	**ENH22**	**ENH21**	**ENH20**	**ENH19**	**ENH18**	**ENH17**	**ENH16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENL31	**ENL30**	**ENL29**	**ENL28**	**ENL27**	**ENL26**	**ENL25**	**ENL24**	**ENL23**	**ENL22**	**ENL21**	**ENL20**	**ENL19**	**ENL18**	**ENL17**	**ENL16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENLx (x=16-31)	x-16	rw	**Emergency Stop Enable for Bit x in SRL** This bit enables the emergency stop feature selectively for each SRL bit. If the emergency stop condition is met and enabled (ENLx = 1), the SRL[x] bit is of the data register DD.DDL[x] is used for the shift register load operation. 0_B Emergency stop feature for bit SRL[x] is disabled. 1_B The emergency stop feature for bit SRL[x] is enabled.
ENHx (x=16-31)	x	rw	**Emergency Stop Enable for Bit x in SRH** This bit enables the emergency stop feature selectively for each SRH bit. If the emergency stop condition is met and enabled (ENHx = 1), the SRH[x] bit of the data register DD.DDH[x] is used for the shift register load operation. 0_B Emergency stop feature for bit SRH[x] is disabled. 1_B The emergency stop feature for bit SRH[x] is enabled.

Downstream Data Extension Register

The Downstream Data Extension Register DDE contains data to be transmitted during data frames.

DDE

Downstream Data Extension Register (074$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DDHE															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDLE															
rw															

Field	Bits	Type	Description
DDLE	15:0	rw	**Downstream Data Extension for SRL Shift Register** Contains the data bits to be transmitted during the SRL active phase of a data frame.
DDHE	31:16	rw	**Downstream Data Extension for SRH Shift Register** Contains the data bits to be transmitted during the SRH active phase of a data frame.

Downstream Data Mirror Register

The Downstream Data Mirror Register DDM provides alternative address for the DD register. The purpose of this alternative address is to provide convenient way to write up to 64 bit data blocks to the MSC module by using DMA with automatic address increment and wrap around. The DMA writes first to the DDE register, and then to the DDM register, which behaves in the same way as a write to the DD register.

DDM
Downstream Data Mirror Register (070$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DDHM															
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDLM															
w															

Field	Bits	Type	Description
DDLM	15:0	w	**Downstream Data Mirror for SRL Shift Register** Contains the data bits to be transmitted during the SRL active phase of a data frame.
DDHM	31:16	w	**Downstream Data Mirror for SRH Shift Register** Contains the data bits to be transmitted during the SRH active phase of a data frame.

Downstream Command Mirror Register

The Downstream Command Mirror Register DCM provides alternative address for the DC register. This alternative address provides convenient way to write up to 64 bit data blocks to the MSC module by using DMA with automatic address increment and wrap around. The DMA writes first to the DCE register, and then to the DCM register. Every write to DCE triggers a command transmission the same way as a write to the DC register.

DCM

Downstream Command Mirror Register (078$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DCHM															
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCLM															
w															

Field	Bits	Type	Description
DCLM	15:0	w	**Downstream Command Mirror of the DC.DCL Bit Field** Contains alternate write location for the command bits to be transmitted during the SRL active phase of a command frame.
DCHM	31:16	w	**Downstream Command Mirror of the DC.DCH Bit Field** Contains the alternate write location for the command bits to be transmitted during the SRH active phase of a command frame.

Downstream Command Extension Register

The Downstream Data Extension Register DDE contains data to be transmitted during data frames, which are concatenated to the standard 32-bit commands, after the second select bit.

DCE

Downstream Command Extension Register (07C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DCEH															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCEH															
rw															

Field	Bits	Type	Description
DCEH	31:0	rw	**Downstream Command Extension for SRH Shift Register** Contains the data bits to be transmitted during the second half of the command frame in CX mode.

38.4.5 Asynchronous Block Registers

Asynchronous Block Configuration Register

The Asynchronous Block Configuration register ABC contains all the bit fields related to the ABRA block.

>> **Configuring the ABRA block**

*Note: Switching the peripheral clock off and on is done by using two writes: first a write of zero to the CLKSEL bitfield, and then a second write switching on the clock source. Between the first and the second write a delay of minimum 4 * (1/f_{PER}) + 2 * (1/f_{CLC}) must be inserted by software, where f_{PER} is the frequency being switched off with the first write. This also applies if in-between the writes the clock frequency is changed in the CCU.*

ABC

Asynchronous Block Configuration Register (080_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ABB	**0**	**CLKSEL**			**UIE**	**UFM**		**UNF**	**0**	**UASR**	**UIP**		**NDA**		
rw	r	rw			rw	w		r	r	rw	rw		rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIE	**OFM**		**OVF**	**0**	**OASR**	**OIP**		**HIGH**				**LOW**			
rw	w		r	r	rw	rw		rw				rw			

Field	Bits	Type	Description
LOW	3:0	rw	**Duration of the Low Phase of the Shift Clock** Defines the duration of the low phase of the shift clock in periods of f_A in the range of 1 to 16. 0_H 1 … F_H 16
HIGH	7:4	rw	**Duration of the High Phase of the Shift Clock** Defines the duration of the high phase of the shift clock in periods of f_A in the range of 1 to 16. 0_H 1 … F_H 16
OIP	9:8	rw	**Overflow Interrupt Node Pointer** OIP selects the service request output line SRn (n = 0-3) for the overflow interrupt. 00_B Service request output SR0 selected 01_B Service request output SR1 selected 10_B Service request output SR2 selected 11_B Service request output SR3 selected

Field	Bits	Type	Description
OASR	10	rw	**Overflow Alternate Service Request** Selects if the interrupt signal is routed to the alternate service request node. See **Figure 547**. 0_B SR Multiplexer selected 1_B Alternate Service Request Node (SR4) selected
OVF	12	r	**Overflow Flag** Indicates if overflow error has occurred. 0_B No overflow error 1_B Overflow error
OFM	14:13	w	**Overflow Flag Modify** Sets or clears the overflow flag OVF. 00_B No action 01_B Set (and triggers the overflow interrupt if enabled) 10_B Clear (additionally flushes the ABRA FIFO and resets the bit counter) 11_B No action
OIE	15	rw	**Overflow Interrupt Enable** Enables or disables the path of the interrupt signal towards the interrupt node. If enabled, an overflow event triggers an interrupt, and if disabled then not. The OVF flag always indicates the occurrence of an overflow event, independently of OIE. 0_B Overflow interrupt disabled 1_B Overflow interrupt enabled
NDA	18:16	rw	**N Divider ABRA** Defines the division ratio in the range of 1 to 8. 000_B 1 … 111_B 8
UIP	20:19	rw	**Underflow Interrupt Node Pointer** UIP selects the service request output line SRn (n = 0-3) for the underflow interrupt. 00_B Service request output SR0 selected 01_B Service request output SR1 selected 10_B Service request output SR2 selected 11_B Service request output SR3 selected
UASR	21	rw	**Underflow Alternate Service Request** Selects if the interrupt signal is routed to the alternate service request node. See **Figure 547**. 0_B SR Multiplexer selected 1_B Alternate Service Request Node (SR4) selected
UNF	23	r	**Underflow Flag** Indicates if underflow error has occurred. 0_B No underflow error 1_B Underflow error

Field	Bits	Type	Description
UFM	25:24	w	**Underflow Flag Modify** Sets or clears the underflow flag UNF. 00_B No action 01_B Set (also triggers the underflow interrupt if enabled) 10_B Clear (additionally flushes the ABRA FIFO and resets the bit counter) 11_B No action
UIE	26	rw	**Underflow Interrupt Enable** Enables or disables the path of the interrupt signal towards the interrupt node. If enabled, an underflow event triggers an interrupt, and if disabled then not. The UNF flag always indicates the occurrence of an underflow event, independently of UIE. 0_B Underflow interrupt disabled 1_B Underflow interrupt enabled
CLKSEL	29:27	rw	**Clock Select** Selects the clock source for the ABRA block. 000_B no clock 001_B f_{PER} **others**, reserved (no clock)
ABB	31	rw	**Asynchronous Block Bypass** Defines if the asynchronous block and the n-divider of the MSC downstream path (located parallel to the fractional divider) are used or not. If bypassed, then also disabled. 0_B Bypassed and disabled 1_B Not bypassed and active
0	11, 22, 30	r	**Reserved** Read as 0; shall be written with 0.

38.5 Use Cases

This section provides a code example for the MSC module to give an overview about the core functionality. It describes how to send data frames continuously without any software interaction, called data repetition mode. This examples uses interface MSC 0.

Each interface can send command and data frames (see also **Frame Formats and Definitions**) via downstream channel. Between each data frame one or more passive frames can be inserted (see also **Figure 529**). This example realizes sending out only data frames in repetition mode with one passive frame in between. The module has more features than the example shows. For further details please see user's manual.

The MSC module can also receive frames with the upstream channel. These frames are usually status frames from the external devices. The example keeps the focus on sending out via downstream channel so the upstream channel is not part of this chapter. The initialization process follows the following order:

1. Enable MSC 0
2. Initialize MSC 0 for down streaming
3. Start sending

Step description to initialize the MSC module:

(Line 1) reset ENDINIT to get access to ENDINIT protected registers. (see also ENDINIT protection chapter).

(Line 2) enable control of the module, in clock control register **CLC**.

(Line 3) read back (dummy variable has to be defined). The reading process ensures that the write process from line 2 is done.

(Line 4) This line loads the fractional divider register **FDR**. DM=01 sets the module to normal divider mode while STEP defines the Baudrate.

Note: Assuming fclc=100MHz and STEP=3FE --> Baudrate =25 Mbaud

(Line 5) set ENDINIT to lock the protected register again.

(Line 6) This line loads the downstream control register **DSC**. TM=1 sets the module to data repetition mode, NDBL and NDBH are defining the length of the SRL and SRH active phase, here set to 15 bit (see also **Figure 525**). PPD=2 sets the length of the passive frame to 2 x tFCL and DSDIS=1 disables sending. Sending will be enabled after initialization.

(Line 7) This line set the number of passive time frames in the downstream status register **DSS**. NPTF=1 --> one passive time frame gets inserted.

(Line 8) The data gets loaded in the downstream data register **DD**. The data itself is here just an example.

(Line 9) This line resets DSDIS, so the module starts sending out data and passive time frames.

Initialization of the MSC module:

```
// enable MSC 0
(1)     SCU_vResetENDINIT (0);              // enable ENDINIT reg
(2)     MSC0_CLC.U = 0x000;                 // disable module control
(3)     dummy = MSC0_CLC.U;                 // read to check the made changes
(4)     MSC0_FDR.U = (1 << 14) | 0x3FE;     // DM=01, STEP=3FE
(5)     SCU_vSetENDINIT (0);                // lock ENDINIT reg
// Initialize MSC 0 for down streaming
(6a)    MSC0_DSC.U = 0x02001081;            // PPD=2,NDBH=16,NDBL=16,TM=1
(6b)    MSC0_ISC.U = (0x1 << 22);           // DSDIS=1
(7)     MSC0_DSS.U = (1 << 8);              // NPTF=1
(8)     MSC0_DD.U = 0x5555AAAA;             // downstream data (example)
```

```
// Start sending
(9)     MSC0_ISC.U = (0x1 << 6);            // DSDIS=0
```

38.6 IO Interfaces

This section describes the on-chip connections of the MSCx module.

The table below lists the interfaces of the MSC to other modules in the device.

Table 343 List of MSC64 Interface Signals

Interface Signals	I/O	Description
clocks		**Clock socket**
sx_dft_clocks		**DfT Clock socket**
reset		**Reset socket**
bus_fpi_slave		**FPI slave socket**
protect		**Protection socket**
sx_sff		
debug		**OCDS socket**
standard_spll		**Scan socket**
standard_ppll		**Scan socket**
sx_irq_msc		**Interrupt socket**
sx_gtm2msc		**GTM signal socket**
SR0_INT	out	**MSC Service Request 0**
SR1_INT	out	**MSC Service Request 1**
SR2_INT	out	**MSC Service Request 2**
SR3_INT	out	**MSC Service Request 3**
SR4_INT	out	**MSC Service Request 4**
ALTINL(15:0)	in	**GTM timer output vector - low part**
ALTINLEXT(15:0)	in	**GTM timer output vector - low extension part**
ALTINH(15:0)	in	**GTM timer output vector - high part**
ALTINHEXT(15:0)	in	**GTM timer output vector - high extension part**
prdcfg	in	**product configuration**
INJ0	in	**Injection signal from port**
INJ1	in	**Injection signal from port**
EN(3:0)	out	**Chip Select**
EMGSTOPMSC	in	**Emergency stop signal from SCU**
FCLN	out	**Shift-clock inverted part of the differential signal**
FCLP	out	**Shift-clock direct part of the differential signal**
SDI(7:0)	in	**Upstream asynchronous input signal**
SON	out	**Data output - inverted part of the differential signal**
SOP	out	**Data output - direct part of the differential signal**

38.6.1 Port Emergency Stop Signal (from SCU)

The emergency stop input signal Port Emergency Stop of the MSCx modules is connected to the output signal of the emergency stop input control logic. This logic is located in the SCU. Its functionality is controlled by the SCU emergency stop register.

Note: If the Port Emergency Stop signal is used by the MSC module, setting the SCU.EMSR.MODE=0 is mandatory. MODE=1 is not allowed to be used with the MSC module.

38.6.2 Interrupt Service Requests

All five service request outputs SR[4:0] are connected to the Interrupt Router Module. There they are routed further to a CPU or a DMA. The MSCx_SRCy registers are also located in the IR (Interrupt Router) Module.

38.6.3 GTM Connections

Each MSC module can be connected to up to 64 TOM or ATOM outputs of the GTM module. For description of the connections between the GTM and MSC modules, see the section “MSC Connections” in the GTM chapter, which contains detailed description of the signals sets and the multiplexing options.

38.7 Revision History

Table 344 Revision History

Reference	Change to Previous Version	Comment
V5.0.10		
Page 87	Clean up revision history, no functional changes.	
V5.0.11		
Page 85	Updated initialization code.	

39 Single Edge Nibble Transmission (SENT)

The SENT module communicates with the external world via one I/O line for each channel. The STx lines are the receive data input signals. They can overlay ADC inputs. If the optional SPC mode is used, they can be used on a port configured with an open drain transistor. This way the optional SPC data can be transmitted and the line is used bidirectionally. In case of an external transceiver, receive and transmit path can be routed to two different ports.

Figure 562 shows a global view of the SENT interface.

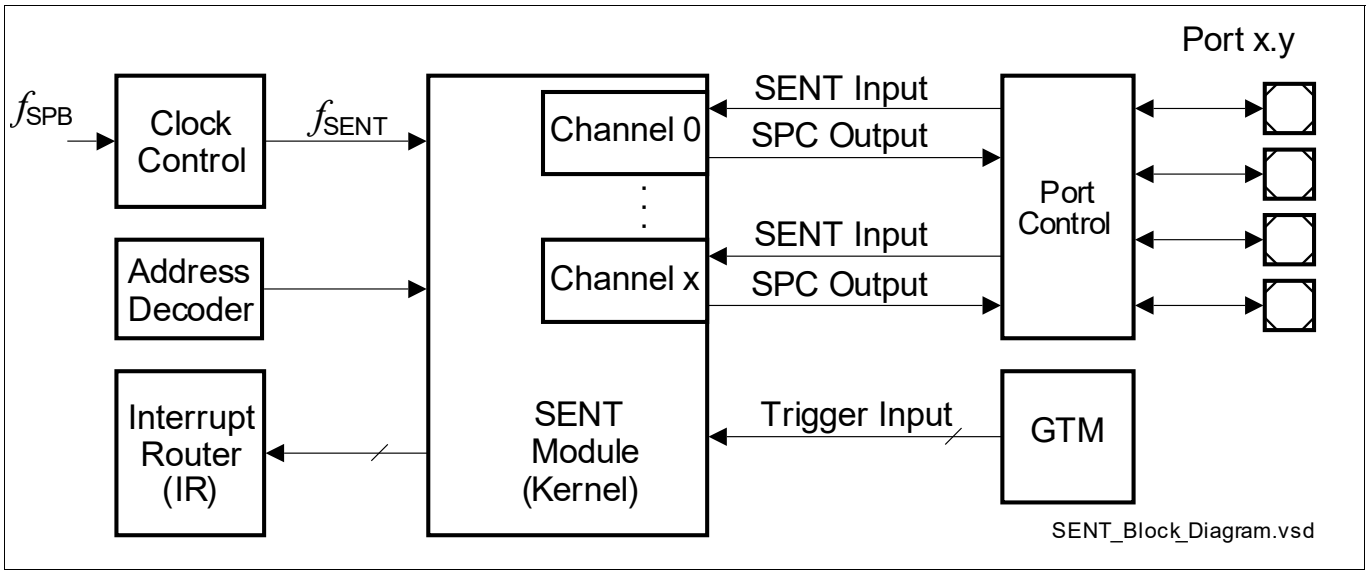


Figure 562 General Block Diagram of the SENT Interface

39.1 Feature List

FEATURE LIST

- Conformance with SENT SAE J2716 042016 (fully backward compatible with SAE J2716 2010 standard specifications)
- Data rates of up to 65,8 kbit/s at 3 µs tick length and 6 data nibbles on each channel
- Support of standard tick times (3 µs through 90 µs)
- 25 SENT channels implemented, all working independently in parallel
- Message tick time programmable between 0.2 µs and 1024 µs, see restrictions in **Chapter 39.3.4.1**
- Status nibble optionally included in the checksum (default not included)
- Sticky interrupt flags, error interrupt optional (default disabled)
- Configurable frame length (default is 24 bit), max data size is 32 bits
- Serial data processing optional (default: disabled)
- Option for bigger frame lengths (must still be fix for each application)
- transparent mode (nibble CRCs are written to the receive control register for SW processing)
- Support of SPC
- Support of trailing Pause Nibble of any length (even longer than 70 ticks)
- Indication of system status: STOP, INITIALIZED, RUNNING, SYNCHRONIZED
- The receiver module will monitor the message for the following error conditions:
- Calibration pulse length deviates more than +/-25% from the nominal 56 ticks

- Too many or too few nibbles between calibration pulses.
- Checksum error.
- Successive calibration pulse differ by more than 1.5625%
- Any nibble data values measured as < 0 or >15.
- When any of those errors is detected, the receiver module shall declare that a message error has occurred and ignore the entire message.
- Any of those errors shall cause the receiver to begin searching for a valid calibration pulse to re synchronize.
- Option to enable/disable the check of the next calibration pulse before validation of received data
- Digital Glitch filter suppressing noise
- Buffer overrun detection
- Optional output inversion for use of external open drain transistor
- Optional input inversion for use of external open transistor for level shifting
- Interrupt on status nibble violation
- Programmable Nibble sorting to relief CPU
- Time stamp generation
- Watch Dog on incoming frames
- Support frequency drift analysis based on frame length for frames with pause pulse

REFERENCES

This Internal Target Specification is based on: SAE Standard "**SAE J2716 042016**".

39.2 Overview

The SENT interface provides a serial communication link typically used to connect sensors or other peripheral devices.

Clock control, address decoding, and service request control are managed by the SENT module kernel.

The SENT IP-module performs communication according to the SENT specification SAE J2716 042016.
While staying compliant to this standard, it is able to cover as well the Short PWM Code (SPC) protocol extensions. This enhances the standardized SENT protocol defined by SAE J2716 042016. SPC enables the use of enhanced protocol functionality like "synchronous", "range selection" and "ID selection" protocol mode.

Receive data on a SENT channel can be set up according to the underlying application. In particular the number of nibbles forming one value is configurable.

In SPC mode, maintaining the sample and transmission schedule as well as providing message status information is support.

The register set of the SENT module can be accessed directly by the CPU for configuration, data read out and status query.

39.3 Functional Description

39.3.1 General Operation

The Single Edge Nibble Transmission encoding scheme (SENT) is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/D converters and PWM and as a simpler low cost

alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

Figure 563 shows a typical application in which a SENT interface reads a sensor device.

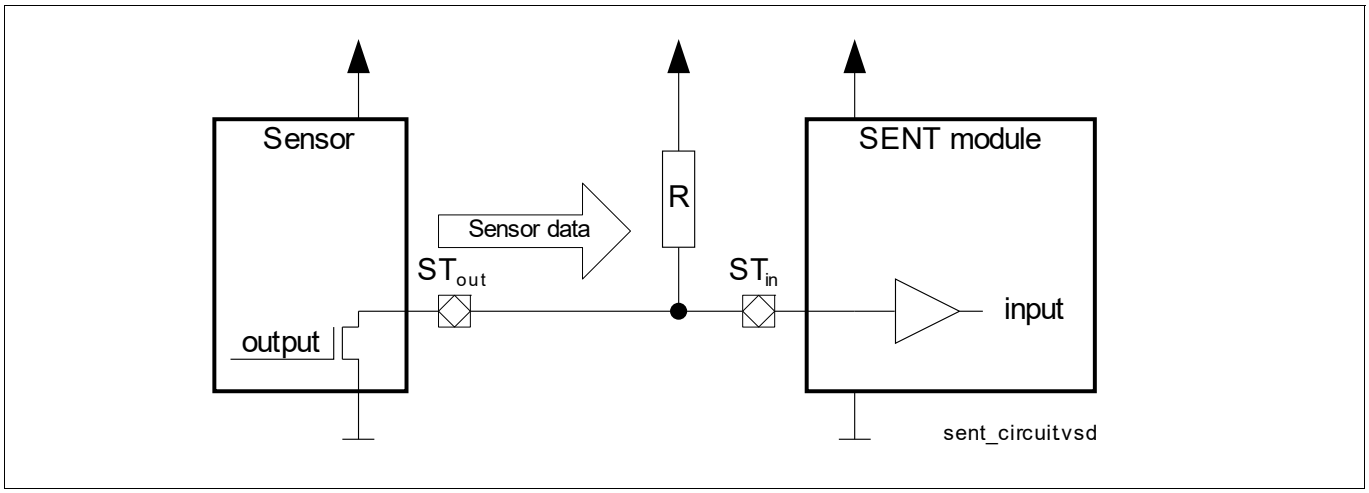


Figure 563 SENT to External Device Connection

SENT communication is unidirectional from sensor to controller without any synchronization. The sensor signal is transmitted as a series of PWM blocks measured as falling to falling edge times.

The Short PWM Code (SPC) extension overcomes the drawback of unidirectionality as said above while keeping conformance to the standard.

Figure 564 shows a typical application in which an SPC enabled SENT ECU reads an SPC enabled SENT sensor device.

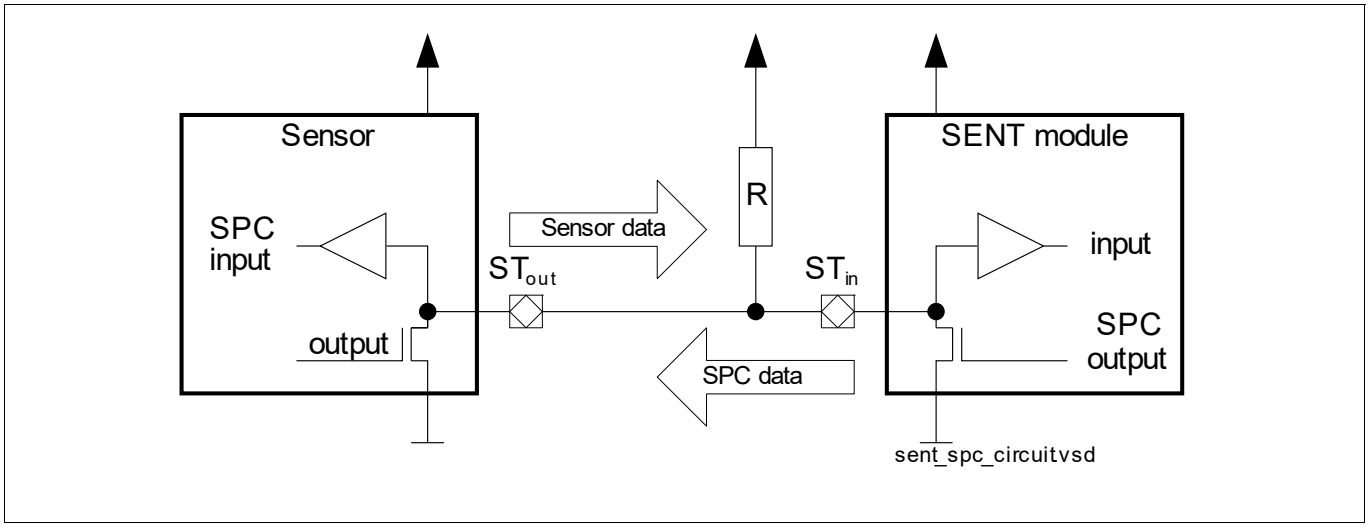


Figure 564 SENT SPC to External Device Connection

Some applications are:

- Read out of the external throttle position sensor (e.g. SENT enabled linear hall sensor)
- Receiving pedal position
- Synchronize sampling and read out of up to four sensors on one single line (SPC)
- Selection of 1 out of 4 sensors connected to a single SENT channel.
- Serial connections to other peripheral devices

39.3.1.1 Definitions

SENT: Single Edge Nibble Transmission

Nibble: Four Bit value between 0 and 15 = half a Byte = one character in hex (0 to F)

SPC: Short PWM Code

PWM: Pulse Width Modulation

ASIC: Application Specific Integrated Circuit

CAN: Controller Area Network

LIN: Local Interconnect Network

ISO: International Organization for Standardization

ECU: Electronic Control Unit

FSM: Finite State Machine

39.3.2 Standard SENT Operation

Standard SENT Mode supports communication fully compliant to SAE J2716 042016, in which only the external device is sending and the ECU is receiving. In this unidirectional communication, both transmitter and receiver use the same data frame format and have the same baud rate. These settings are defined at system integration time and do not change for a given application. Data is received on pin ST. **Figure 565** shows the block diagram of the SENT Module when operating in Standard SENT Mode.

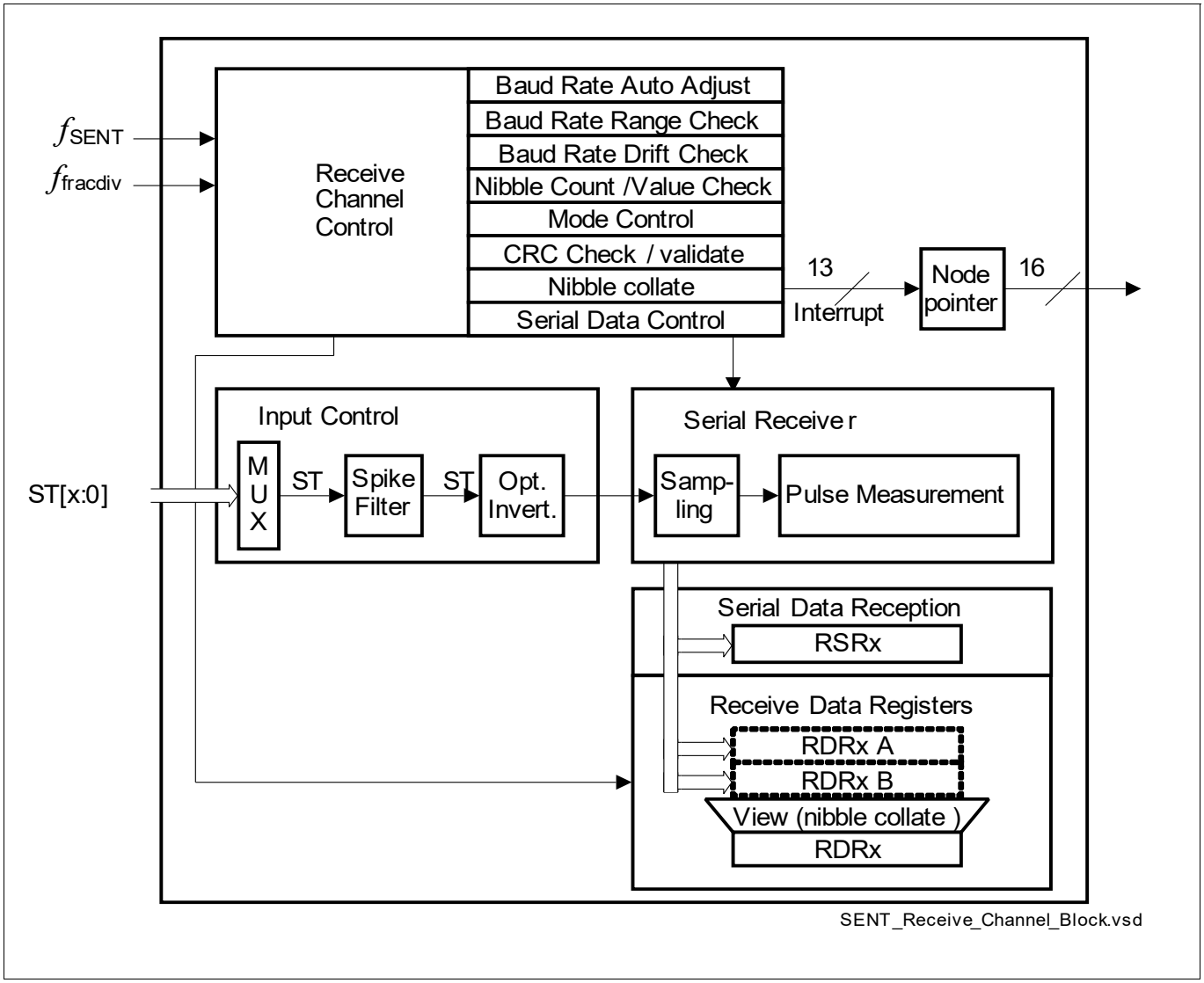


Figure 565 Standard SENT Mode Operation

39.3.2.1 Frame Formats and Definitions

This section describes the frame formats and definitions of the SENT protocol.

Basic Definitions

Figure 566 shows the layout and definitions of a standard SENT frame. Note that the SENT standard does not specify whether the most significant nibble or the least significant nibble is sent out first.

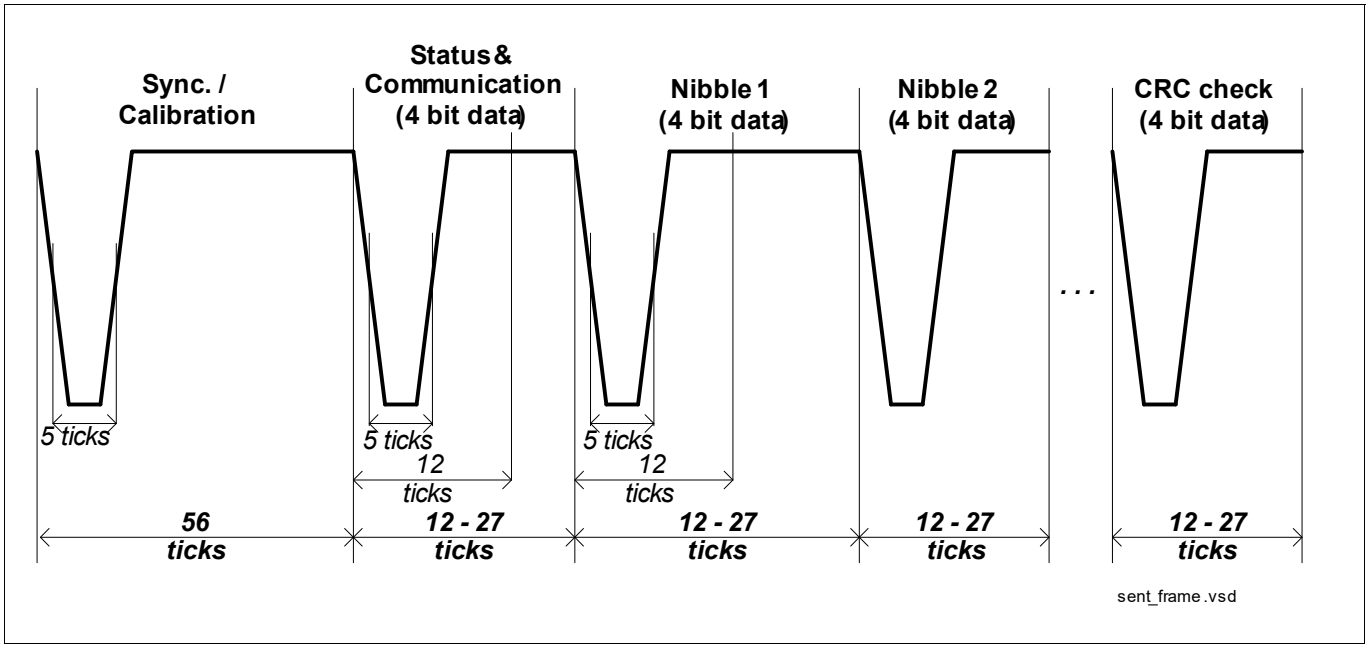


Figure 566 Standard Encoding Frame

Serial Data Encoding in Status and Communication Nibble

The Status and Communication nibble is used by sensors that transmit extra serial data such as part numbers or diagnostic information. Data bits from this nibble are constructed across multiple SENT messages to form a Short Serial Message or an Enhanced Serial Message. These messages are alternately referred to as “Slow Channel” data in the SAE J2716 042016.

Table 345 Short Serial Data Encoding

Bit Number	Bit Function
0	Reserved for specific application
1	Reserved for specific application
2	Serial Data message bits
3	Message start = 1, otherwise = 0

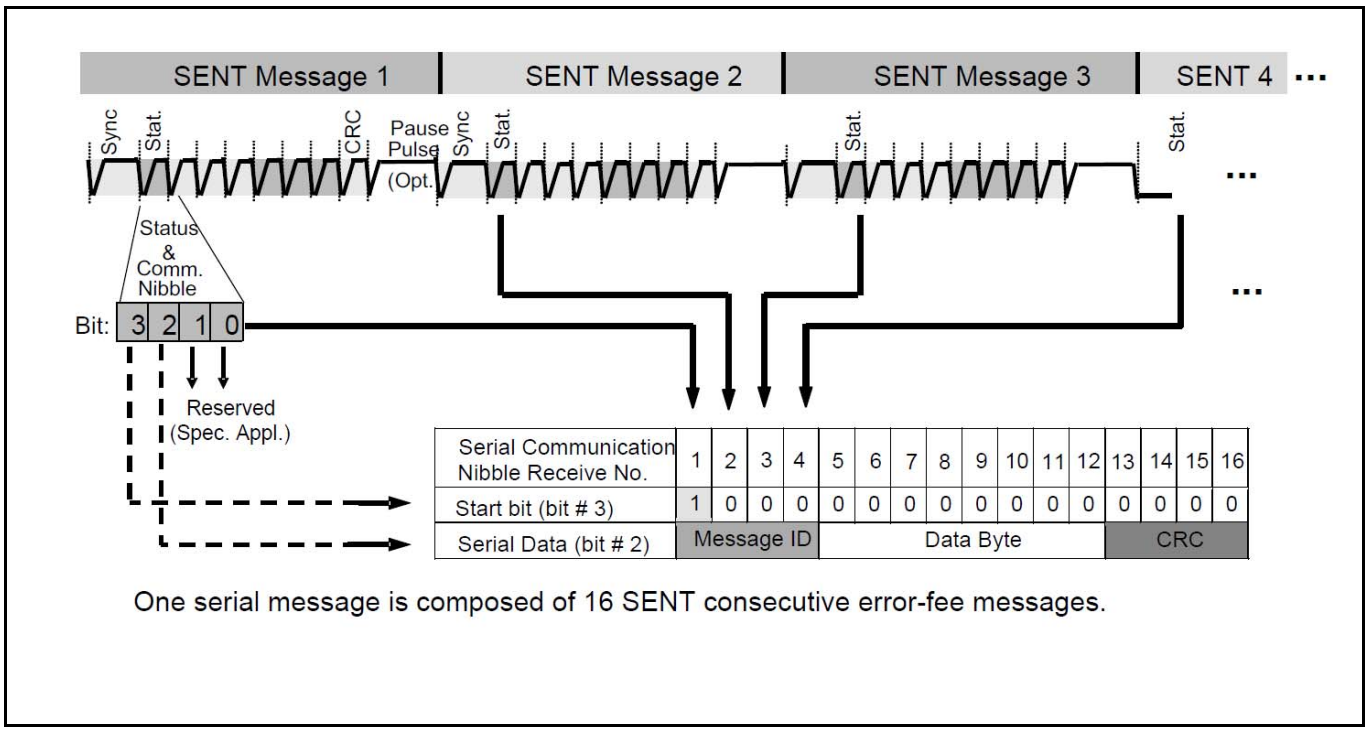


Figure 567 Short Serial Message, Serial Data Encoding over 16 messages

Short Serial Message Format

The SENT kernel supports the Short Serial Message Format defined in the SAE J2716 042016. The Short Serial Message Format provides 16 bits of data, constructed across 16 SENT messages. Each of the 16 messages contains one bit of Short Serial Message data in Status and Communication nibble bit 2.

Serial data is communicated in a 16-bit sequence as shown in **Table 346**. The start of a Short Serial Message is indicated by a 1 in bit 3 of the Status and Communication nibble. That SENT message and the next 15 must be successfully received (no errors) for the Short Serial Message data to be received. The CRC generation is identical to the CRC generation on the data nibbles. Short Serial Message data is provided in the SDSx register.

Table 346 Short Serial Message Format

Serial Communication Nibble Receive No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Startbit (bit # 3)	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Serial Data (bit # 2)	Message ID				Data Byte								CRC			

CRC Calculation

SENT signal data and Short Serial Message data is secured with a 4 bit CRC. The standard CRC calculation method is defined in the SAE J2716 042016.

RCR.CRZ defines, if a zero nibble is added for calculation or not.

The alternative checksum nibble is a 4-bit CRC of the data nibbles (including the status nibble if RCR.SNI is set, as used in sensor TLE4998). The CRC is calculated using a polynomial x^4 +x^3 + x^2 + 1 with a seed value of 0101.

In order to facilitate CRC implementations and to avoid ambiguities, an example implementation of the alternative 4-bit CRC is given below. This is used e.g. in the sensor TLE4998 for the signal data. See **Figure 568** for details.

// Fast way for any µC with low memory and compute capabilities

```
// contains input data (status nibble, 6 data nibble, CRC)
char Data[8] = {…};
// required variables and LUT
char CheckSum, i;
char CrcLookup[16] = {0, 13, 7, 10, 14, 3, 9, 4, 1, 12, 6, 11, 15, 2, 8, 5};
CheckSum= 5; // initialize checksum with seed “0101”
for (i=0; i<7; i++) {
CheckSum = CheckSum ^ Data[i];
CheckSum = CrcLookup[CheckSum];
};
// finally check if Data[7] is equal to CheckSum
```

For the “Enhanced Serial Message Frame Format”, an own 6-bit CRC calculation method is defined by the standard.

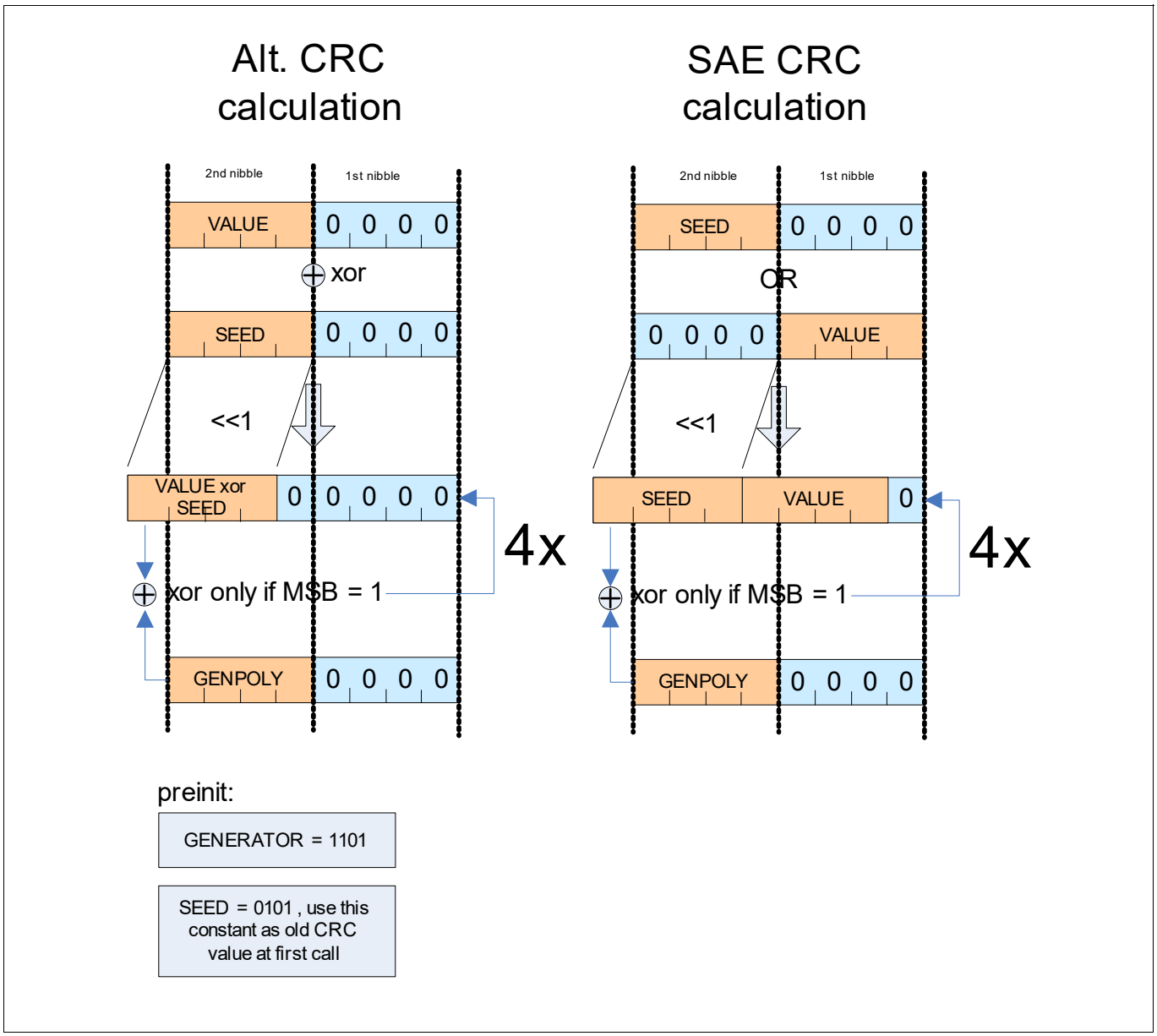


Figure 568 Alternate vs. SAE CRC Calculation

Enhanced Serial Message Format

The SENT kernel supports the Enhanced Serial Message Format defined in the SAE J2716 042016. The Enhanced Serial Message Format provides 21 bits of message data and a 6-bit CRC, constructed across 18 SENT messages. The serial data is transmitted bit wise per frame in bits [3:2] of the Status and Communication nibbles of 18 consecutive messages from the transmitter.

Table 347 Enhanced Serial Data Encoding

Bit Number	Bit Function
0	Reserved for specific application
1	Reserved for specific application
2	Serial Data message bits
3	Message start = 1 1 1 1 1 1 0

Enhanced Serial Message Frame

Serial data is communicated in an 18 frame sequence as shown in **Table 348**. The start of an Enhanced Serial Message is indicated by the unique pattern “1111110” in bit #3 of the status and communication nibble, constructed across 7 SENT messages. In message 1 - 6 they are set to “1”. In message 7, 13 and 18 they are set to “0”. 18 consecutive SENT messages must be successfully received (no errors) for the Enhanced Serial Message data to be received. The 6-bit Enhanced Serial Message CRC is different from the 4-bit CRC on SENT (fast channel) messages. (See SENT Standard)

An Enhanced Serial Message contains 21 bits of message data and a 6-bit frame-check sequence. A configuration bit (serial data bit #3 of the 8th SENT message's Serial Communication Nibble) determines how the SENT module interprets the serial data.

- Configuration bit = 0: 12-bit data and 8-bit message ID, see **Table 349**
- Configuration bit = 1: 16-bit data and 4-bit message ID, see **Table 350**

Enhanced Serial Message data is provided in the SDSx register.

Table 348 Enhanced Serial Message Frame

Serial Communication Nibble Receive No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Serial Data (bit # 3)	1	1	1	1	1	1	0	C	8-bit ID (7-4)				0	8-bit ID (3-0) or 4-bit data				0
Serial Data (bit # 2)	6-bit CRC						12-bit data field											

Note:

10. Message 8 Status Nibble bit # 3 is Configuration Bit C that defines the Enhanced Serial Data format:
*0: 12-bit data, 8-bit ID, see **Table 349***
*1: 16-bit data, 4-bit ID, see **Table 350***
11. Messages 17-14 Status Nibble bit # 3 contain either the lower 4 bits (Bit 3-0) of 8bit ID or 4 additional data bits.

Table 349 shows in detail the frame structure for Enhanced Serial Data Frames with Configuration Bit = 0.

Table 349 Enhanced Serial Message Configuration - Bit = 0

Serial Communication Nibble Receive No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Serial Data (bit # 3)	1	1	1	1	1	1	0	0	8-bit ID (7-4)				0	8-bit ID (3-0)				0
Serial Data (bit # 2)	6-bit CRC						12-bit data field											

Table 350 shows in detail the frame structure for Enhanced Serial Data Frames with Configuration Bit = 1.

Table 350 Enhanced Serial Message Configuration Bit = 1

Serial Communication Nibble Receive No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Serial Data (bit # 3)	1	1	1	1	1	1	0	1	4-bit ID (3-0)				0	4-bit data (15-12)				0
Serial Data (bit # 2)	6-bit CRC						12-bit data field											

39.3.3 SPC Operation

The module supports a SPC (Short PWM Code) protocol, which enhances the standardized SENT protocol (Single Edge Nibble Transmission) defined by SAE J2716 042016. SPC enables the use of enhanced protocol functionality due to the ability to select between “synchronous”, “range selection” and “ID selection” protocol mode or even “bidirectional transmit mode”.

39.3.3.1 Synchronous Transmission

In the “synchronous” mode, the sensor (slave) starts to transfer a complete data frame only after a low pulse is forced by the master on the OUT pin. This means that the data line is bidirectional - an open drain output of the micro controller (master) sends the trigger pulse. The sensor then initiates a sync pulse and starts to calculate the new output data value. After the synchronization period, the data follows in form of a standard SENT frame, starting with the status, data and CRC nibbles. At the end, an end pulse allows the CRC nibble decoding and indicates that the data line is idle again. The timing diagram in **Figure 569** visualizes a synchronous transmission

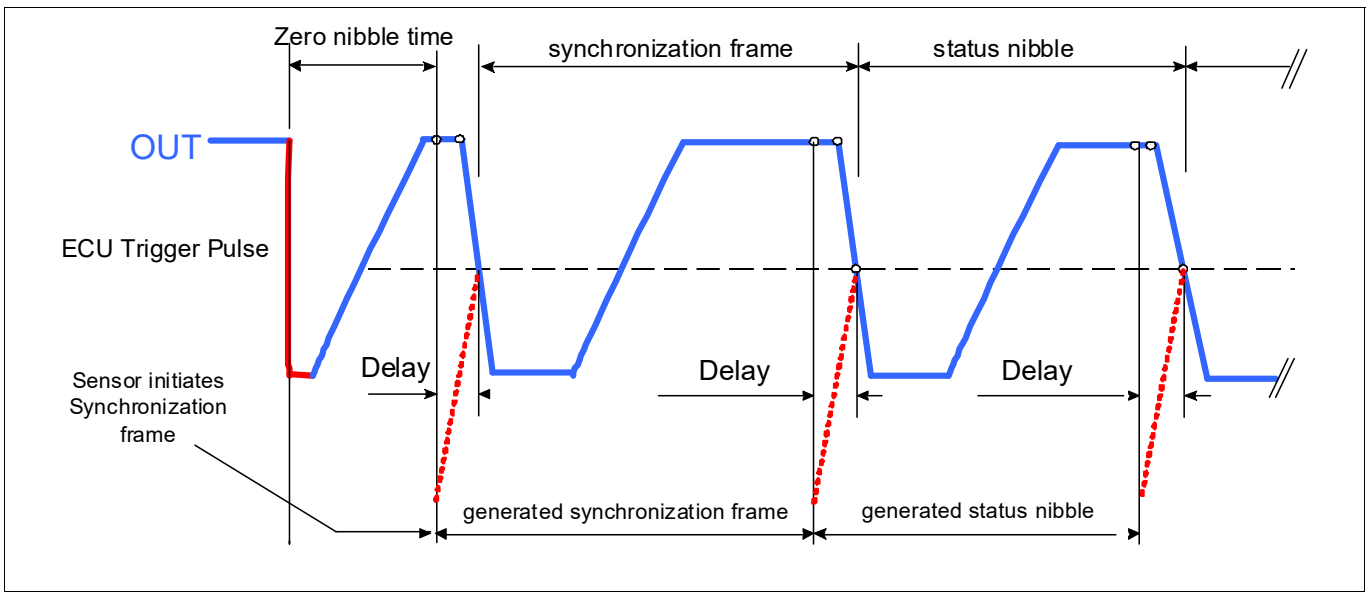


Figure 569 Synchronous Transmission (SPC)

39.3.3.2 Range Selection

The low time duration of the master can be used to select the range of the sensor in SPC dynamic range selection mode.

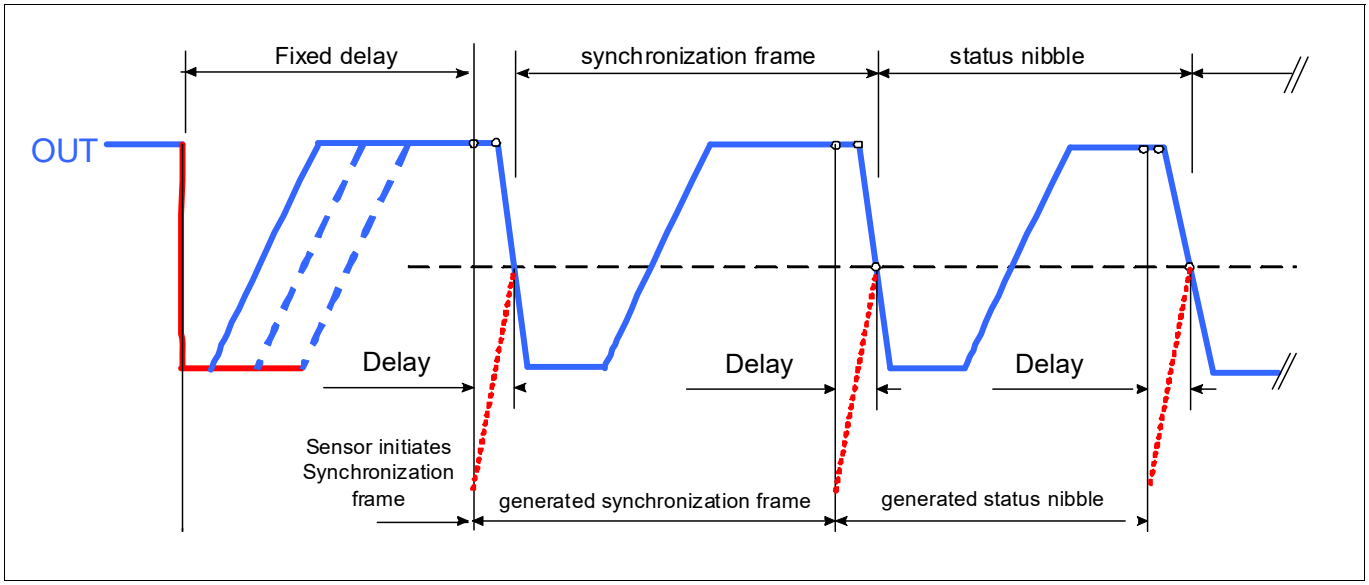


Figure 570 Range (SPC)

39.3.3.3 ID Selection

This functionality is similar to the previous mode, but instead of switching the range of one sensor, one of up to four sensors are selectable on a bus (bus mode, 1 master with up to 4 slaves). This allows parallel connection of up to 4 sensors using only three lines (VDD, GND, OUT), as illustrated in **Figure 571**.

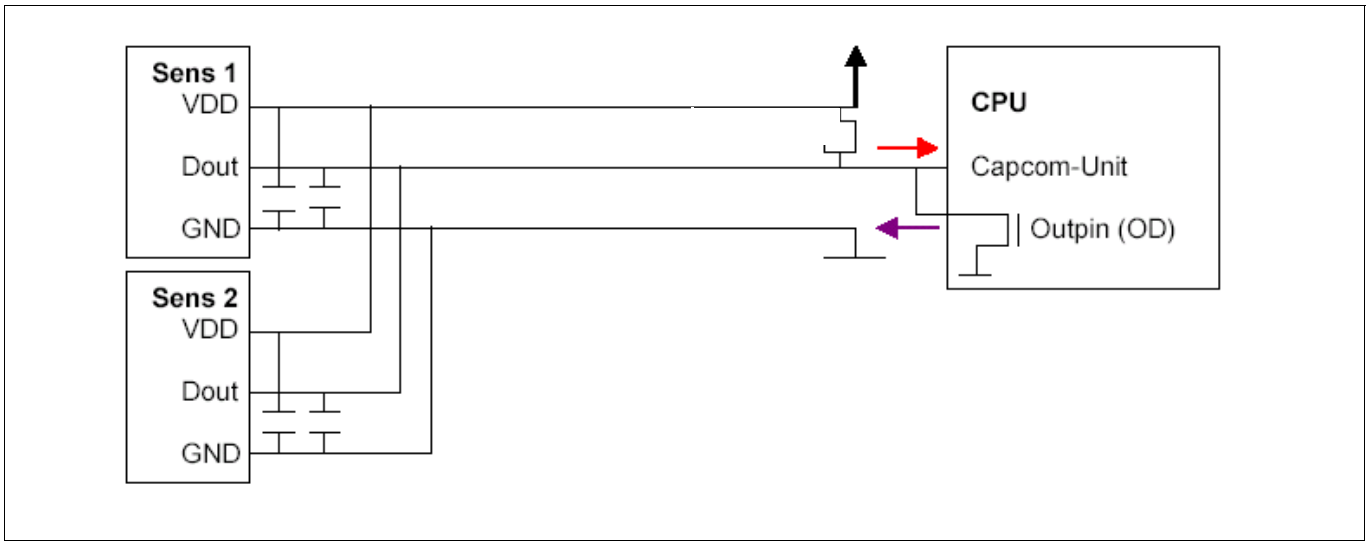


Figure 571 ID Selection (SPC)

39.3.3.4 Bidirectional Transmit Mode

After addressing the sensor the ECU interrupts the sensor sync pulse by pulling down the line. The ECU starts to transmit own nibbles on the time base of the selected sensor. The ECU has to adapt the sensors timing from an earlier received cycle.

The sensor detects the falling edge that infringes the protocol and switches to receive mode as illustrated in **Figure 572**.

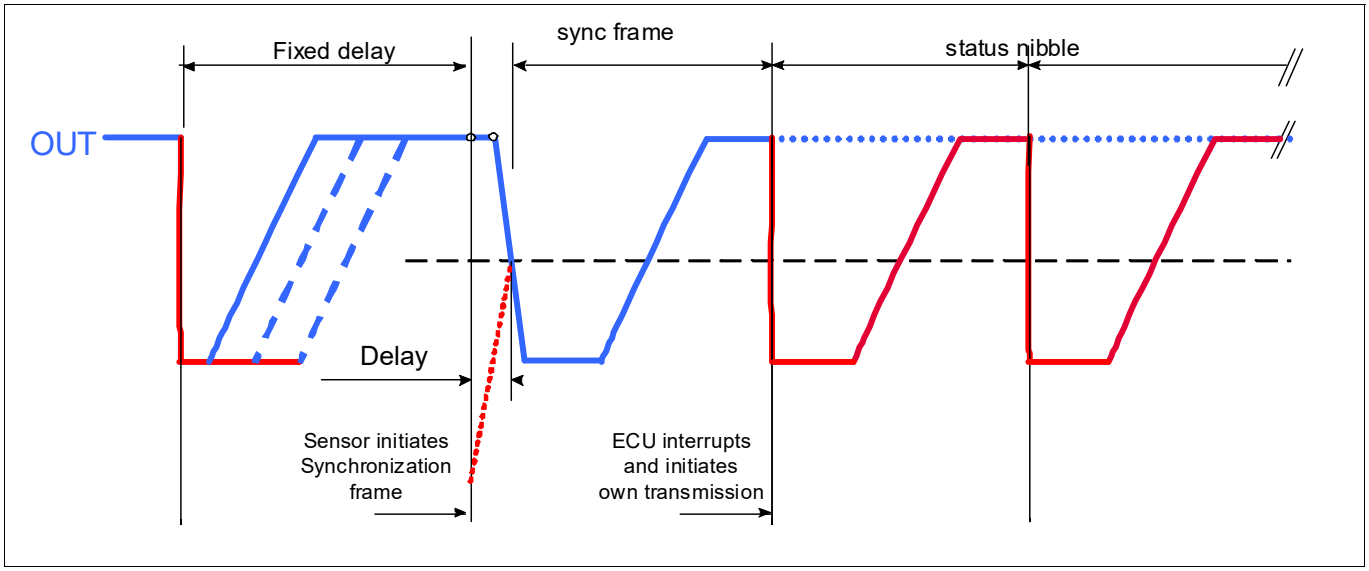


Figure 572 Bidirectional Transmit Mode (SPC)

39.3.3.5 SPC Timing

An SPC transmission is initiated by a Master pulse on the OUT pin. To detect a low level on the OUT pin, the voltage must be below a threshold Vthf. The sensor detects that the OUT line has been released as soon as Vthr is crossed. **Figure 573** shows the timing definitions for the master pulse. The master low time tmlow as well as the total trigger time tmtr are individual for the different SPC modes and are given in the sensors specifications.It is recommended to chose the typical master low time exactly between the minimum and the maximum possible time given by the connected sensor: tmlow,typ = (tmlow, min + tmlow, max) / 2.

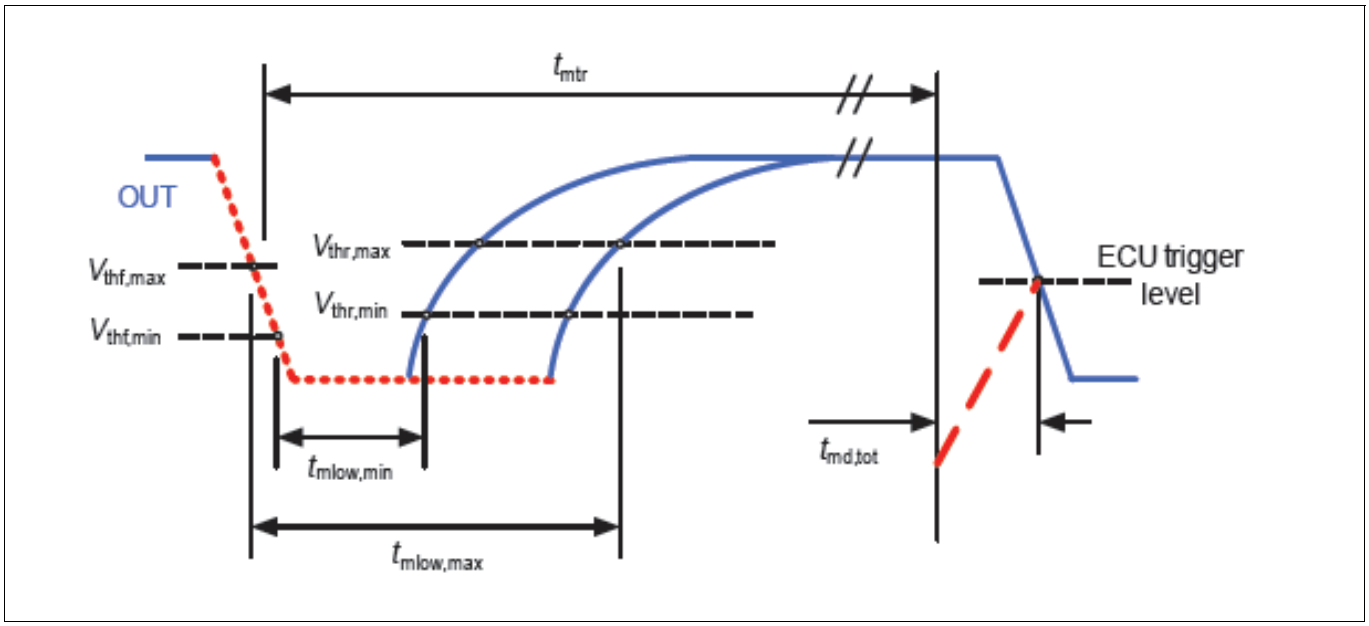


Figure 573 Timing (SPC)

39.3.3.6 Abort of Frames

Only a reset condition of the device, a hard suspend or choosing Mode 0 by clearing bit field SCRx.TRIG can abort a current SPC transmission immediately. During hard suspend, the port output drives its current value until suspend is terminated. If the receive channel becomes disabled, the soft suspend mode is requested, or the sleep mode is entered, the SENT module still does start a new frame transmission. If one of these three conditions becomes active during a running frame transmission, the frame transmission is completely finished before the requested abort state is entered. Note that in this case no frame finished interrupt is generated any more.

Received frames are aborted immediately on a disable, suspend or sleep request.

39.3.4 Baud Rate Generation

The nominal baud rate of each channel is individually adjustable. This is required as it is depending from the connected sensor and its current deviation from nominal frequency.

The actual baud rate of the channel follows automatically the baud rate of the connected device and its current deviation from nominal frequency. The adaptation range is +/- 25% as specified in SAE J2716 042016.

Chapter 39.3.4 shows in detail how the baud rate for each channel is adjusted. 1 / f_{tick_x} defines the resulting tick length.

In the first stage, a global fractional divider serves as pre divider. Its intermediate frequency $f_{fracdiv}$ can be set up so that it is handy as input frequency for the different SENT channels. The clock signal f_{pdiv} of a channel must always be at least 20 times the nominal tick frequency of the channel. This is to allow for the highly flexible and big tolerance of +/-25% versus the sending device. In addition the module must be able to detect a deviation in the length of 2 consecutive Synchronization / Calibration pulses of 1.5625% (1/64) and adapt the own baud rate.

The tick time is typically 3 µs but can vary by standard and take values up to 90 µs. To achieve this each channel has its own fractional divider. This allows to downscale the frequency of f_{tick} precisely to the required tick frequency.

The SENT module provides two clock signals to the channels (**Figure 574**):

- f_{SENT}
  This is the module clock that is used inside the SENT kernel for control purposes such as clocking of control logic and register operations. The frequency of f_{SENT} is always identical to the system clock frequency f_{SPB}. The clock control register SENT_CLC makes it possible to enable/disable f_{SENT} under certain conditions.
- $f_{fracdiv}$
  This clock is the module clock that is used inside the SENT kernel for baud rate generation of the serial channels. The fractional divider register SENT_FDR controls the frequency of $f_{fracdiv}$. Usually not required and set to 1.

The channels generate two local clock signals:

- f_{pdiv_x}
  This clock is the channel clock that is used inside the SENT channel x for baud rate generation of the serial channel. The divider register SENT_CPDRx controls the frequency of f_{pdiv_x}.
- f_{tick_x}
  This clock is the channel clock that is used inside the SENT channel as the baud rate frequency. The fractional divider register SENT_CFDRx controls the frequency of f_{tick_x}.The higher the value of DIV the higher the precision with which the Synchronization / Calibration pulse and the deviation in the length of 2 consecutive Synchronization / Calibration pulses (drift) can be measured.
  DIV must be set in the range of [2200, 49100].

Each time a Synchronization / Calibration pulse starts DIVM is measured. DIVM is the result of measuring the calibration pulse duration with f_{pdiv_x}. At the end of the Synchronization / Calibration pulse this value is taken as new divider of the CFDR.

Each time a pulse starts, the internal accumulator of the CFDR is preset with DIVM / 2. This is required to center the data eye and to make the margin symmetrical.

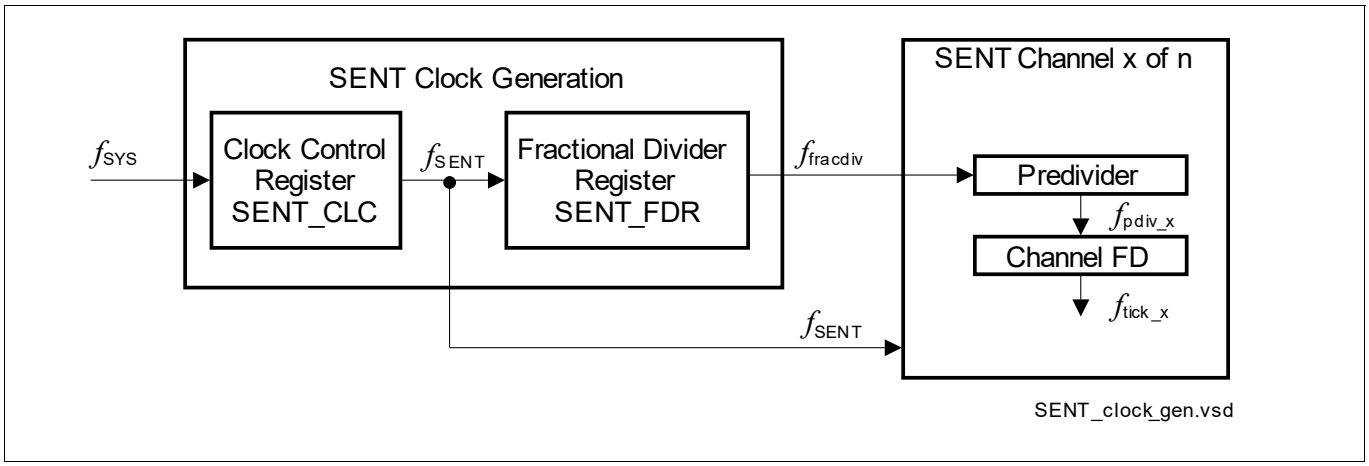


Figure 574 SENT Module Clock Generation

The following two formulas define the frequency of $f_{fracdiv}$:

$f_{fracdiv} = f_{SENT}$ / (1024 - SENT_FDR.STEP); FDR.DM = 01_B (39.1)

$f_{fracdiv} = f_{SENT}$ x SENT_FDR.STEP / 1024 with STEP = 0 … 1023; FDR.DM = 10_B (39.2)

The following formula defines the frequency of f_{pdiv_x}:

$f_{pdiv\_x} = f_{fracdiv}$ / (SENT_CPDRx.PDIV + 1) (39.3)

The following formula defines the nominal frequency of f_{tick_x}. For the actual frequency of f_{tick_x} DIV is replaced by DIVM after the current sensor frequency was validly measured.

$f_{tick\_x} = f_{pdiv\_x}$ x 56 / SENT_CFDRx.DIV with DIV = 2200… 49100 (39.4)

39.3.4.1 Operation outside the granted Standard Limits

The SENT module is granted to comply SAE J2716 042016. The tolerances given in this standard are valid for tick times of 3 ... 90 µs. Nevertheless, the SENT module can operate outside this standard. A minimum tick time of 0.2 µs is programmable. In these use cases outside the standard it is in the responsibility of the system integrator to do proper jitter calculation and system design. Same holds true for tick times of > 90 µs. Note that the low pass filter as given in SAE J2716 042016 needs to be replaced by a circuit fit for the intended use case as well. Such use is not recommended, as a standard is missing for the system.

39.3.5 Error Detection Capabilities

Each SENT channel can detect and signal the following error conditions:

Protocol Level:

- Calibration pulse length deviates more than +/-25% from the nominal 56 ticks
- Too many or too few nibbles between calibration pulses
- Checksum error
- Successive calibration pulse differ by more than 1.5625%
- Any nibble data values measured as < 0 or >15
- Wrong Status and Communication nibbles
- Serial Communication CRC error

Transfer Management Level:

- Receive Data Buffer Overrun
- SPC Data Buffer Underrun

39.3.6 Support for Frequency Drift Analysis in Frames with Pause Pulse (FDFL)

Each SENT channel supports FDFL mode, i.e. monitoring the following condition for messages with pause pulse and fixed message length:

Ratio of calibration pulse to message length varies by > 1/64 or < -1/64 from one message to another.

The FDFL mode is entered if RCR.FDFL, RCR.IEP and RCR.IDE are set. RCR.CFC is ignored in this mode.

There is no automatic calculation or check done by HW in this mode nor is FDI issued. Frames that have an invalid frequency drift and would cause an FDI are still accepted and not dropped by HW. (FRI is not affected by this mode. I.e. the HW still checks the sensor frequency range.) The calculation of the ratio and the comparison with

the ratio of the preceding frame needs to be done in SW. The SW needs to decide whether or not to accept the frame based on this calculation.

The value for the calibration pulse length can be read by SW from CFDR.DIVM.

The value for the message length including pause pulse can be read by SW from RSR.FRLEN only if RCR.FDFL, RCR.IEP and RCR.IDE are set.Both values are valid after RSI. RSI indicates the Frame Buffer valid immediately after the Pause Pulse belonging to the current frame is received if no other error occurs.

FRLEN is a 16 bit counter with saturation (stops at 0xFFFF / 65.535) and is driven by f_{pdiv}.These factors limit the maximum supported length of frames including pause pulses. For the typical use case 1 / f_{tick} = 3 µs a good choice for f_{pdiv} is 16.6 MHz (PDIV = 5; f_{SENT} = 100 MHz). Thus, the frame length can be counted up to 3.93 ms.

Note that the nominal frame length must be 20% shorter to allow the system to compensate a 20% frequency drift. This is just an example calculation and not all products may support all frequencies.

Figure 575 shows the two times that are measured and that need to be set in relation to each other:

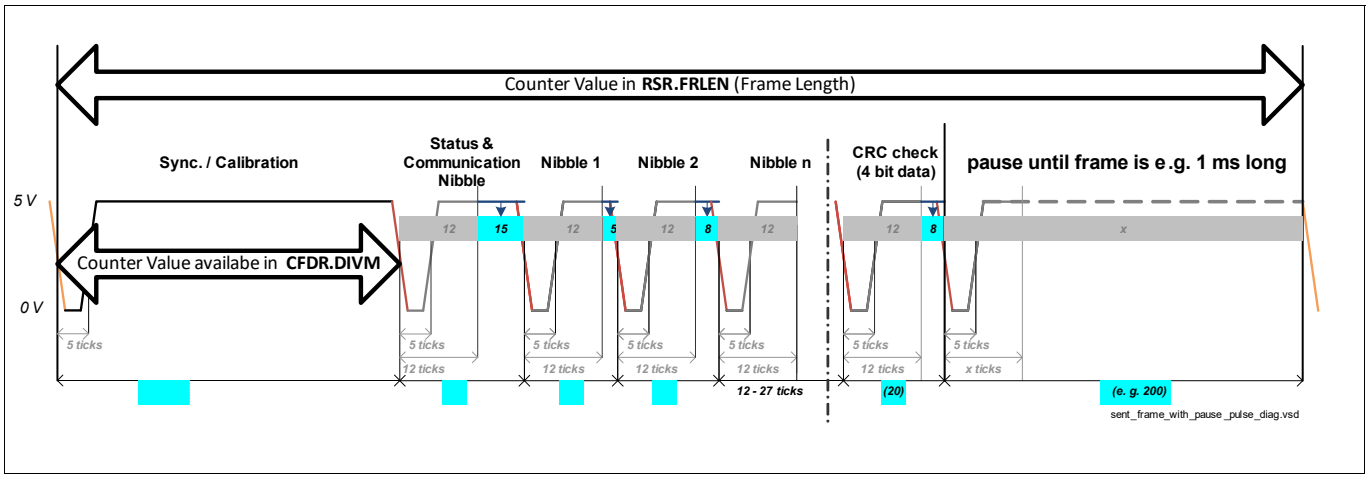


Figure 575 Frequency Drift Analysis in Frames with Pause Pulse

39.3.7 Digital Glitch Filter

Very slow slopes and signal noise can lead to fast transitions of the input signal. These unwanted transitions are suppressed by a digital glitch filter similar to a Filter and Prescaler Cell (FPC) in Delayed Debounce Filter Mode with up and down (no reset).

It is built for filtering very fast transitions only. The filter calculates the integral of the signal. If the integral reaches a programmable saturation point, the signal change is notified to the pulse measurement unit. Thus it helps to find the exact pulse length for said slow slopes in noisy environment.

The glitch filter is clocked with f_{pdiv}. If the state of the input sample differs from the current output signal value, the internal counter is incremented by one. When the state of the input sample matches the current output signal value and the timer is not in idle, the timer is decremented by one. When the timer matches the compare value stored in IOCRx.DEPTH, the level of the output signal line is inverted. The timer will be reset and set to idle state again.

The depth of the filter can be programmed to a value between 1 and 15. Typically a depth of 3 to 5 T_{pdiv} is sufficient. By default it is cleared. If IOCRx.DEPTH is cleared the filter is inactive. Nevertheless, the input signal is sampled with f_{pdiv}.

The internal signal after the filter will change value not before the new value was sampled DEPTH times. If during this period a spike occurs, it takes 2 x T_{pdiv} times longer for the internal signal to change value. The filters principal implies a delay of the signal by (DEPTH x T_{pdiv}).

Upon detection of glitch during rising or falling edge, IOCRx.REG or IOCRx.FEG is set. The rising / falling edge glitch flags must be reset by software.

Figure 576 shows the digital glitch filter:

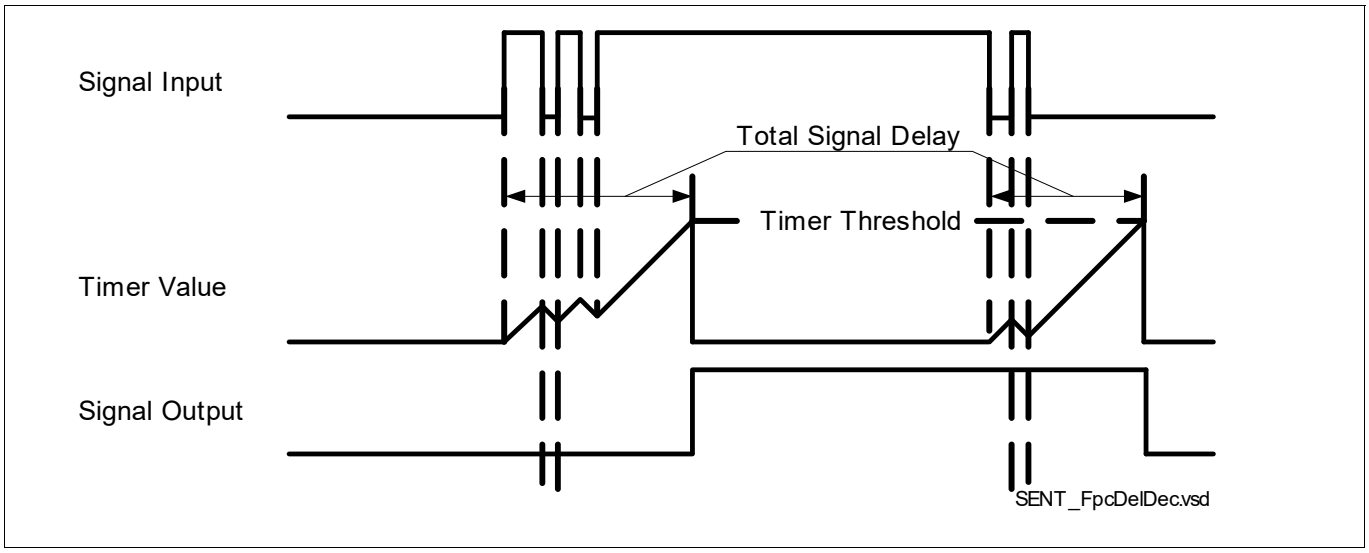


Figure 576 Digital Glitch Filter

39.3.8 Interrupts

10 Interrupt request lines are available for the SENT module.

RDI indicates a receive data interrupt. It is activated when a received frame is moved to a Receive Data Register RDR. RSI indicates a receive frame success interrupt, i.e. the CRC was successful. Both RDI and RSI will be issued together in normal use cases where the frame size is not bigger than 8 nibbles and CRC is correct. RBI indicates a receive buffer overrun interrupt. It is activated when a new frame is transferred to a Receive Data Register RDR while the old value was still not read by the host (“overwrite”), i.e. the kernel wants to set any of the two interrupts RSI and RDI and finds any of these two interrupts already set. TDI indicates a transmit interrupt. It is activated when data is moved from a SCR to a transmit shift register. TBI indicates a transfer buffer under run interrupt. It is set after data has been completely transferred (PLEN exceeded) and no new data was written to SCRx. In addition the protocol error interrupts are available: FRI, FDI, NNI, NVI, CRCI. If one of the protocol interrupts is activated, data is to be treated as invalid according to SAE J2716 042016. WSI, SDI SCRI treat the interrupts referring to the Status and Communication nibble. WDI is the Watch Dog Error Interrupt. It is issued if the time between two frames is too long.

For acceleration of the interrupt service routine, a Register INTOV is implemented that holds a flag for each channel. This flag is automatically set if there is an interrupt pending for the channel which is enabled. It is automatically reset, if no more enabled interrupt is pending for this channel.

The interrupt request or the corresponding interrupt set bit (in register INTSET) can trigger the interrupt generation at the selected interrupt node. The service request pulse is generated independently from the interrupt flag in register INTSTATx. The interrupt flag can be cleared by software by writing to the corresponding bit in register INTCLR.
If more than one interrupt source is connected to the same interrupt node pointer (in register INPx), the requests are combined to one common line.

39.3.9 Interface Connections of the SENT Module

Figure 577 shows a more specific implementation details and interconnections of the SENT module.

The SENT module is supplied with a separate clock control, address decoding, and interrupt control logic. Outputs of the GTM module are connected to the timer inputs.

The inputs of the receive channels of the SENT module as well as the SPC outputs (SPCn) are connected to GPIO lines. If SPC outputs are mapped to the same port pin like the referring SENT data input line on a specific product, they can be used in a quasi bidirectional mode (i.e. open drain out and input on the same pin).

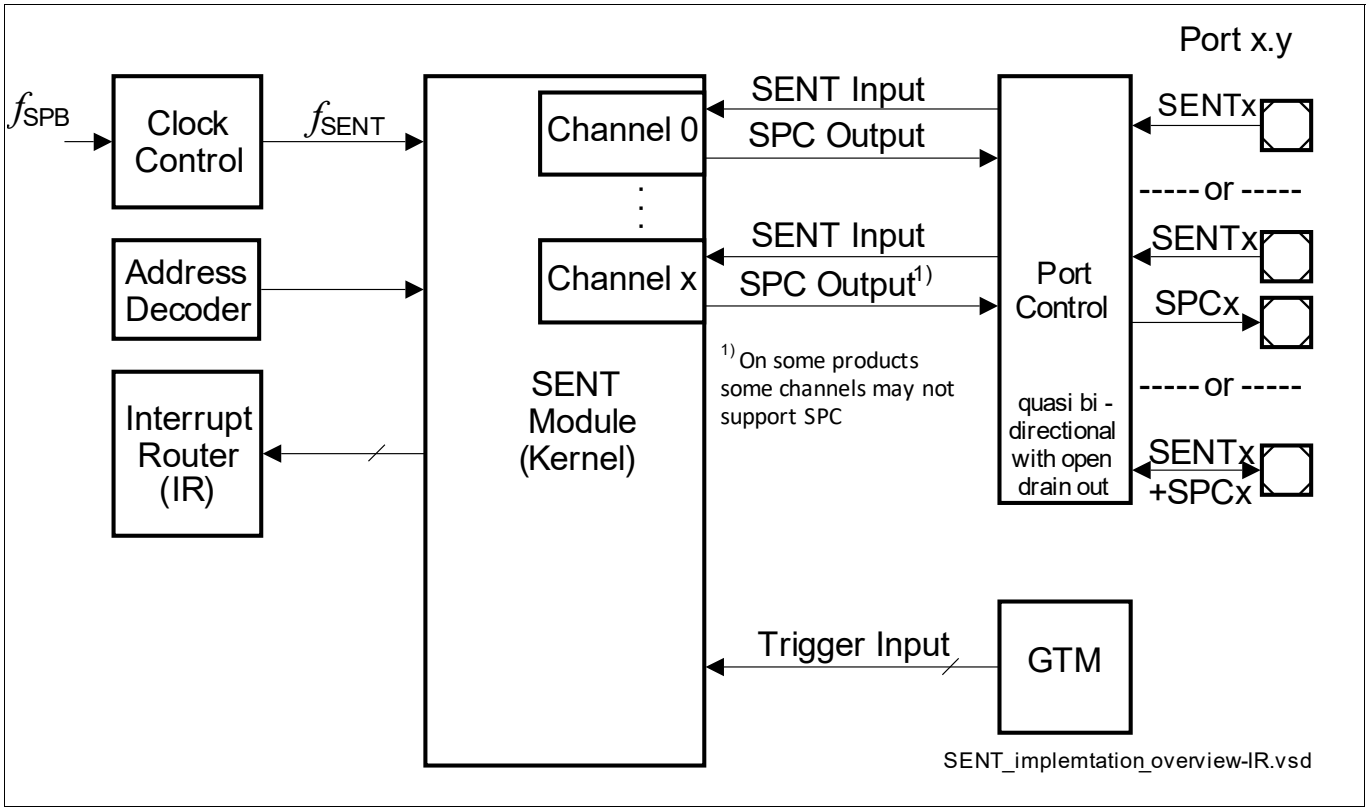


Figure 577 SENT Module Implementation and Interconnections

39.3.9.1 Trigger Inputs

The module has 25 SENT channels and the same number of trigger inputs but not more than n+1 = 16. They can be randomly chosen by programming IOCRx.ETS. The trigger inputs (TRIG[n:0]) of the SENT module are connected to the GTM as shown in Device Specific Appendix.

39.3.10 Port Control

The SENT input channels are overlain with standard ADC channels as they are replacing former analog signals. In addition each channel is connected to general purpose I/O pads as well. The selection from one of up to 4 alternate inputs per channel is done by application SW by programming SENT_IOCRx.ALTI.

Depending on the implementation some products support SPC output as well on some of the channels.

The following port control operations must be executed for the SPC output I/O pads:

- Input/output function selection (Port IOCR registers)
- Pad driver characteristics selection for the outputs (Port PDR registers)

39.3.10.1 Input/Output Function Selection

SENT can be used in three different ways:

- ADC input (dedicated for ADC) overlain with SENT digital input and no output option on this pad.
- Standard general purpose input/output function on two different I/O pads per channel. I.e. input on ADC or on I/O pad while the output is chosen to be on a different I/O pad. This allows the use of external transceivers.
- SPC open drain output on a digital I/O pad and SENT input from same digital I/O pad ("quasi bidirectional SENT/SPC mode")

The SENT module overlays dedicated analog to digital converter (ADC) pads as digital input pad.

SENT physical layer uses 5V signal voltage levels. All SENT inputs are mapped to 5 V compatible ADC pads. Vddm must be supplied with 5V for SENT operation.

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as port direction (input/output), alternate output selection, pull-up/down devices (for pure inputs only), and open-drain selections. The I/O pads for the SENT module are controlled by the Port input/output control registers shown below. Please refer to the Port chapter to make conscious decisions here!

Quasi bidirectional settings are possible only, where an SPC output and a SENT input of the same channel are connected to the same pad. This configuration is not available on all SENT inputs pads and varies by product.

For some channels SPC outputs and SENT input are connected to the same I/O pad. This allows for quasi bidirectional operation: I.e. the output is configured as open drain (e.g. P00_IOCR0.PC1 = 11110_B) while the same pad is used as input (e.g. P00_IOCR0.PC1 = $0XXXX_B$). The port in the example may not be present in all products.

Note that in the examples, figures and tables of this chapter, no quasi bidirectional operation is shown. Only pure inputs and pure outputs are shown for the sake of simplicity.

Alternate inputs that are not connected to any pin are connected to GND internally. E.g. the input selected by SENT_IOCRx.ALTI = 0011_B (aka: "input D") is frequently not in use.

39.4 Registers

This section describes the registers of the SENT module. All SENT register names described in this section will be referenced in other parts of the User's Manual by the module name prefix "SENT_" for the SENT interface.

All registers in the SENT address spaces are reset with the application reset with one exception: OCS is reset with debug reset only.

x can take the values 0 … 24

Table 351 Register Overview - SENT (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	000_H	SV,U	SV,E,P	Application Reset	53
ID	Module Identification Register	008_H	SV,U	BE	Application Reset	22
FDR	SENT Fractional Divider Register	$00C_H$	SV,U	E,P	Application Reset	22
INTOV	Interrupt Overview Register	014_H	SV,U	BE	Application Reset	42
TSR	Module Time Stamp Register	018_H	SV,U	BE	Application Reset	23
TPD	Time Stamp Predivider Register	$01C_H$	SV,U	E,P	Application Reset	23
RDRx	Receive Data Register x	080_H+x* 4	SV,U	BE	Application Reset	37
OCS	OCDS Control and Status	$0E8_H$	U,SV	SV,P	Debug Reset	53
KRSTCLR	Kernel Reset Status Clear Register	$0EC_H$	U,SV	SV,E,P	Application Reset	57
KRST1	Kernel Reset Register 1	$0F0_H$	U,SV	SV,E,P	Application Reset	56
KRST0	Kernel Reset Register 0	$0F4_H$	U,SV	SV,E,P	Application Reset	55
ACCEN1	Access Enable Register 1	$0F8_H$	U,SV	SV,SE	Application Reset	55
ACCEN0	Access Enable Register 0	$0FC_H$	U,SV	SV,SE	Application Reset	54
CPDRx	Channel Pre Divider Register x	100_H+x* 40_H	SV,U	SV,U,P	Application Reset	25
CFDRx	Channel Fractional Divider Register x	104_H+x* 40_H	SV,U	SV,U,P	Application Reset	25
RCRx	Receiver Control Register x	108_H+x* 40_H	SV,U	SV,U,P	Application Reset	27
RSRx	Receive Status Register x	$10C_H$+x* 40_H	SV,U	BE	Application Reset	32

Table 351 Register Overview - SENT (ascending Offset Address) (cont'd)

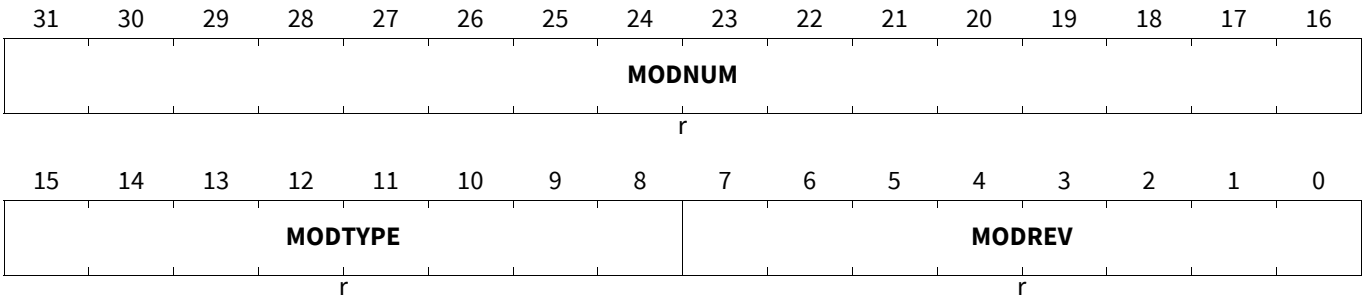
Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
SDSx	Serial Data and Status Register x	110_H+x* 40_H	SV,U	BE	Application Reset	33
IOCRx	Input and Output Control Register x	114_H+x* 40_H	SV,U	SV,U,P	Application Reset	34
SCRx	SPC Control Register x	118_H+x* 40_H	SV,U	SV,U,P	Application Reset	40
VIEWx	Receive Data View Register x	$11C_H$+x* 40_H	SV,U	SV,U,P	Application Reset	38
INTSTATx	Interrupt Status Register x	120_H+x* 40_H	SV,U	BE	Application Reset	42
INTSETx	Interrupt Set Register x	124_H+x* 40_H	SV,U	SV,U,P	Application Reset	46
INTCLRx	Interrupt Clear Register x	128_H+x* 40_H	SV,U	SV,U,P	Application Reset	47
INTENx	Interrupt Enable Register x	$12C_H$+x* 40_H	SV,U	SV,U,P	Application Reset	49
INPx	Interrupt Node Pointer Register x	130_H+x* 40_H	SV,U	SV,U,P	Application Reset	50
WDTx	Watch Dog Timer Register x	134_H+x* 40_H	SV,U	SV,U,P	Application Reset	31
	Reserved (008_H Byte) (x=0-24)	138_H+x* 40_H	BE	BE		
RTSx	Receive Time Stamp Register x	$A80_H$+x* 4	SV,U	BE	Application Reset	37

39.4.1 Module Control

Module Identification Register

The SENT Module Identification Register ID contains read-only information about the module version.

ID
Module Identification Register **(008_H)** **Application Reset Value: 0080 C0XX$_H$**

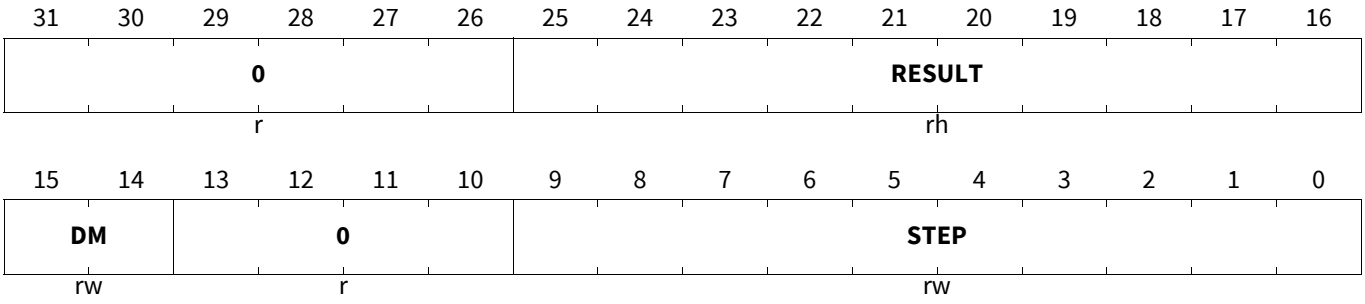


Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01_H (first revision).
MODTYPE	15:8	r	**Module Type** This bit field defines the module as a 32-bit module: $C0_H$
MODNUM	31:16	r	**Module Number Value** This bit field defines the module identification number for the SENT: 0080_H

SENT Fractional Divider Register

The Fractional Divider Register controls the input clock $f_{fracdiv}$ of all SENT channels.

FDR
SENT Fractional Divider Register **($00C_H$)** **Application Reset Value: 0000 0000$_H$**



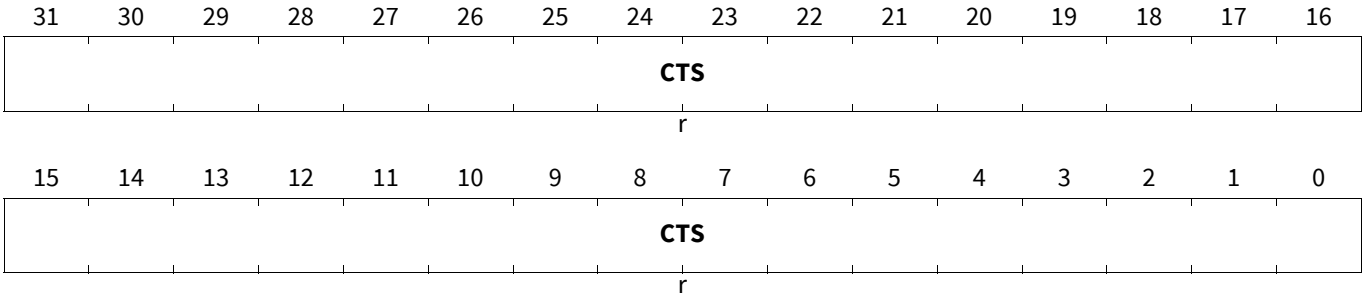
Field	Bits	Type	Description
STEP	9:0	rw	**Step Value** Reload or addition value for RESULT.

Field	Bits	Type	Description
DM	15:14	rw	**Divider Mode** DM selects normal or fractional divider mode. 00_B Fractional divider is switched off; no output clock is generated. The Reset External Divider signal is 1. RESULT is not updated (default after System Reset). 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.
RESULT	25:16	rh	**Result Value** Bit field for the addition result.
0	13:10, 31:26	r	**Reserved** Read as 0; should be written with 0.

Module Time Stamp Register

The SENT Module Time Stamp Register contains read-only information about the current time given in clock cycles generated from TPD since last reset while module was clocked. Writing to TPD clears this register.

TSR
Module Time Stamp Register (018_H) Application Reset Value: $0000\ 0000_H$



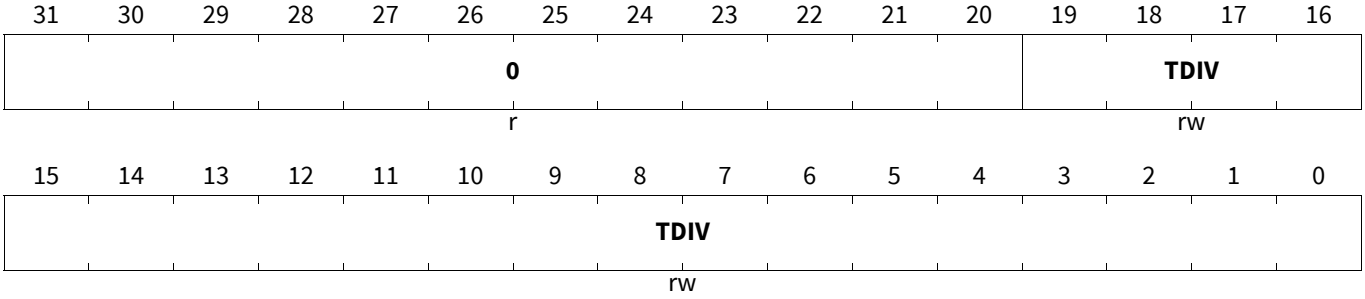
Field	Bits	Type	Description
CTS	31:0	r	**Current Time Stamp for the Module** This bit field shows the current time stamp.

Time Stamp Predivider Register

The SENT Module Time Stamp Predivider Register contains the pre divider that defines the time resolution of TSR. Writing to TPD clears register TSR.

TPD

Time Stamp Predivider Register (01C$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TDIV	19:0	rw	**Divider Factor of Pre Divider for TSR** Divides $f_{fracdiv}$ by (TDIV + 1) and drives TSR.
0	31:20	r	**Reserved** Read as 0; should be written with 0.

39.4.2 Channel Baud Rate Registers

Channel Pre Divider Register x

The Channel Pre Divider Register CPDRx contains the pre divider that is related to the SENT channel baud rate.

See **Equation (39.3)**

CPDRx (x=0-24)

Channel Pre Divider Register x **(100_H+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				PDIV											
r				rw											

Field	Bits	Type	Description
PDIV	11:0	rw	**Divider Factor of Pre Divider for Channel x** Divides $f_{fracdiv}$ by (PDIV + 1) and delivers f_{pdiv_x} to the Channel Fractional Divider. RCR.CEN must be cleared before changing CPDR.PDIV or CFDR.DIV.
0	31:12	r	**Reserved** Read as 0; should be written with 0.

Channel Fractional Divider Register x

The Channel Fractional Divider Register CFDRx contains control bits/bit fields that are related to the SENT channel baud rate.

See **Equation (39.4)** in **Chapter 39.3.4** for a detailed description.

CFDRx (x=0-24)

Channel Fractional Divider Register x **(104_H+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIVM															
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV															
rw															

Field	Bits	Type	Description
DIV	15:0	rw	**Divider Value** Initial and reference divider value for the CFDR. DIV must be programmed > 0. If cleared, DIV becomes 1. If written, DIVM is updated automatically with the same value.RCR.CEN must be cleared before changing CPDR.PDIV or CFDR.DIV. DIV must be set in the range of [2200, 49100].
DIVM	31:16	rh	**Measured Divider Value** DIVM is automatically updated by HW to adjust the receiver frequency to the current sender frequency. This value is kept automatically in the range of 75% DIV < DIVM < 125% DIV Write data is ignored.

39.4.3 Receiver Control and Status Registers

Receiver Control Register x

The Receiver Control Register RCRx contains control bits/bit fields that are related to the SENT receiver operation.

RCRx (x=0-24)

Receiver Control Register x **(108_H+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											FDFL	SUSEN	IDE	ESF	CRZ
r											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRL								CFC	CDIS	SCDIS	SDP	SNI	ACE	IEP	CEN
rw								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CEN	0	rw	**Channel Enable** When CEN is set, the receiver of channel x is enabled. The internal receiver state machine can be initialized by switching the channel off and on. This does not change the current register content. 0_B channel x disabled (default) 1_B channel x enabled
IEP	1	rw	**Ignore End Pulse** When IEP is set, an end pulse is ignored. An end pulse can be generated in SPC mode or as pause pulse. For systems with an end pulse, during synchronize or re-synchronize of reception, if calibration pulses are detected one immediately following the other, the first calibration pulse shall be ignored as it may be a pause pulse with duration matching the calibration pulse range. 0_B End Pulse not ignored (default) 1_B End Pulse ignored
ACE	2	rw	**Alternative CRC Mode Enable** When ACE is set, the CRC is calculated in an alternative way for both: fast (signal) and slow (serial message) data path. If ESF is set, the standard 6 bit CRC is always used for the serial message and ACE is ignored. 0_B Serial CRC calculation as specified in J2716 JAN2010 (default) 1_B Alternative: 4 bit in parallel CRC calculation as used e.g. in hall sensor TLE4998C.
SNI	3	rw	**Status Nibble Included in CRC** When SNI is set, the status Nibble is included in (signal data) CRC. 0_B Status Nibble not included in CRC (default) 1_B Status Nibble included in CRC (as used e.g. in hall sensor TLE4998C).

Field	Bits	Type	Description
SDP	4	rw	**Serial Data Processing Mode** This bit switches automatic serial data processing on. 0_B Automatic Serial Data Processing is disabled. Status and Communication nibble can be read from RSRx for SW processing. (default) 1_B Automatic Serial Data Processing is enabled. Status and Communication nibble can be read from RSRx; Message ID, Serial Data and SCRC can be read from SDSx after serial data interrupt SDI is activated.
SCDIS	5	rw	**CRC for Serial Data Disabled Mode** This bit selects the CRC disabled mode. 0_B CRC is enabled (default) 1_B CRC is disabled CRC nibble can be read from SDSx. The CPU must perform the CRC on the current data by SW.
CDIS	6	rw	**CRC Disabled Mode** This bit selects the CRC disabled mode. 0_B CRC is enabled (default) 1_B CRC is disabled CRC nibble can be read from RSRx. The CPU must perform the CRC on the current data by SW.

Field	Bits	Type	Description
CFC	7	rw	**Consecutive Frame Check** This bit determines the way the most recently received frame buffer is indicated as valid. Note: If FDFL is set, CFC is ignored and the checks described here are not executed. 0_B Check against Past Sync Pulse The current Synchronization / Calibration Pulse is compared to the Synchronization / Calibration Pulse received immediately before. The whole frame is invalid if the Synchronization / Calibration Pulse length differs from the length of the Synchronization / Calibration Pulse before by more than 1.5625%. In this case of error, its length is not used as new reference. In case the check passes and no other error occurs the Frame Buffer is indicated valid (RSI) immediately after CRC calculation result is correct. Resynchronization: On the third successive calibration pulse error, the current calibration pulse value is considered as valid and the message accepted unless the message frame contains other errors. In both cases a receive data interrupt (RDI), or a referring error interrupt is issued. 1_B Check against Future Sync Pulse The current Synchronization / Calibration Pulse is compared with the Synchronization / Calibration Pulse received immediately after the current frame. The whole frame is invalid if the Synchronization / Calibration Pulse length differs from the length of the following Synchronization / Calibration Pulse by more than 1.5625%. In case the check passes and no other error occurs the Frame Buffer is indicated valid (RSI) immediately after the Synchronization / Calibration Pulse following the current frame. Resynchronization: In this case of error, the current length is used as new reference. Note: The whole frame can be indicated valid only after additionally measuring the Synchronization / Calibration Pulse of the successive frame.

Field	Bits	Type	Description
FRL	15:8	rw	**Frame Length** FRL determines the number of data nibbles per frame that the SENT channel x is setup for. Note that FRL does not include the Synchronization / Calibration Pulse, the Status and Communication nibble, the CRC nibble nor the additional zero length nibble that might be introduced by use of SPC If more than 8 nibbles are configured, please note: In addition to the receive success interrupt RSI at the successfully received end of a frame, a receive data interrupt RDI is issued each time 8 nibbles have been transferred to the Receive Data Register RDRx. At the end of a frame, RDI is issued if RSI is issued. If an error occurred, RDI is not set at the end of a frame. If no CRC has been received at the point in time where RDI is issued, the receive data interrupt is no indication whether or not the transfer was successful so far. A CRC Error Interrupt is issued at the end of the frame if Automatic CRC check is enabled and the CRC is wrong. 00_H No data nibble 01_H 1 data nibble 02_H 2 data nibbles 03_H 3 nibbles … 07_H 7 nibbles 08_H 8 nibbles Maximum in normal length mode 09_H 9 nibbles … FF_H 255 nibbles
CRZ	16	rw	**CRC with Zero Nibble for Serial Data** This bit selects the CRC method. If CRZ is cleared, augmentation is selected, (i.e a ZERO NIBBLE is added at the end of CRC calculation (only in calculation)). E.g. as 7th nibble (in case of 6 data nibbles) The enhanced serial message (18 Frames, 6-bit CRC) is not controlled by CRZ but the 6-bit CRC is always augmented according to standard. 0_B Augmentation is selected for both 4-bit message CRC and the 4-bit CRC of the serial messages (legacy, 16 frames) (default) 1_B Augmentation is switched off for both 4-bit message CRC and the 4-bit CRC of the serial messages (legacy, 16 frames).
ESF	17	rw	**Enhanced Serial Frame Mode** This bit selects the serial frame structure. 0_B short (16 frames, 4 bit ID, 8 bit data, 4 bit CRC) 1_B enhanced (18 frames, 4 or 8 bit ID, 12 or 16 bit data, 6 bit CRC)
IDE	18	rw	**Ignore Drift Error Mode** This bit selects if drift errors lead to frame rejection and if an interrupt (INTSTAT.FDI) is generated. Used, if sensors are triggered by SPC. During a long pause period the accumulated drift could be more than 1.5625%. In this special case setting IDE is useful. 0_B Drift Errors enabled (default) 1_B Drift Errors disabled

Field	Bits	Type	Description
SUSEN	19	rw	**Suspend Enable** This bit makes it possible to set the SENT channel into Suspend Mode via OCDS (on chip debug support): Bit SUSEN is reset via OCDS Reset. 0_B An OCDS suspend trigger is ignored by this SENT channel. 1_B An OCDS suspend trigger disables the SENT channel: As soon as the SPC sender logic of the SENT channel becomes idle, the module is stopped while all registers of the channel stay readable. The receiver is stopped unconditionally. A partly received frame is discarded.
FDFL	20	rw	**Frequency Drift Check based on Frame Length** This bit is used for frames with pause pulse only. If set the drift error is not checked by HW. Instead, counter values DIVM and FRLEN are provided to the SW for checking the frequency drift. IEP (Pause Pulse expected) and IDE (no HW check of drift error) must always be set (=1) together with FDFL. Note: If FDFL is set, RCR.CFC is ignored and the checks described there are not executed. 0_B FDFL mode deactivated, RCR.CFC is valid (if IDE is cleared). 1_B FDFL mode is activated, IEP and IDE must be set too for proper function! RCR.CFC is ignored. If no other error occurs the Frame Buffer is indicated valid (RSI) immediately after the Pause Pulse belonging to the current frame.
0	31:21	r	**Reserved** Read as 0; should be written with 0.

Watch Dog Timer Register x

The Watch Dog Timer Register WDTx contains the time out value for channel x. The internal Watch Dog timer is cleared and started automatically each time an RDI (Receive Data Interrupt Request Flag) is set in the INTSTAT of the referring channel and also on any write to WDL that sets WDL>0. Normally the WDTx.WDL is configured only once at module setup. However, if WDTx.WDL is written again with a value larger than 0 during frame reception, the internal Watch Dog timer is NOT cleared and restarted. Instead, the Watch Dog Timer keeps on counting and comparing to the new value. This is not recommended but offers flexibility in exceptional cases. Still, if WDTx.WDL is written with 0 during frame reception, the internal Watch Dog timer is cleared and stopped.

The value entered here defines the time in multiples of Ttick_x (defined in CFDR) from the last RDI on channel x. The internal Watch Dog Timer is compared to WDLx. A match triggers interrupt WDI and stops the internal Watch Dog Counter x. After such a match, the Watch Dog Timer will be cleared and started automatically with clearing bit WDI (by setting INTCLR.WDI). I.e. the Watch Dog Counter is running only if interrupt flag WDI is cleared. If WDL is cleared, the WDTx is stopped/disabled.

WDTx (x=0-24)

Watch Dog Timer Register x **(134_H+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDL															
rw															

Field	Bits	Type	Description
WDL	15:0	rw	**Watch Dog Timer Limit** for channel x .
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Receive Status Register x

The Receive Status Register provides the status information of channel x.

RSRx (x=0-24)

Receive Status Register x **($10C_H$+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRLEN															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				SCN				0		CST		CRC			
r				r				r		r		r			

Field	Bits	Type	Description
CRC	3:0	r	**CRC** of last frame. CRC0 is on bit position 0.
CST	5:4	r	**Channel Status** CST shows the current status of channel x. 00_B STOP Channel is disabled and can be configured 01_B INITIALIZED Channel is configured and enabled and no Synchronization / Calibration Pulse was received since last enable. 10_B RUNNING one or more Synchronization / Calibration Pulses were received and Frequency Range or Frequency Drift not or no longer in range. Fallback status from SYNCHRONIZED. 11_B SYNCHRONIZED Frequency Range and Frequency Drift in range
SCN	11:8	r	**Status and Communication Nibble** of last frame. SCN0 is on bit position 8.

Field	Bits	Type	Description
FRLEN	31:16	r	**Frame Length including Pause Nibble** of last frame. Bit FRLEN0 is on bit position 16. FRLEN is a 16 bit counter with saturation (stops at 0xFFFF / 65.535) and is driven by f_{pdiv}. This counter supports monitoring the following condition for messages with pause pulse and fixed message length: Ratio of calibration pulse to message length varies by > 1/64 or < -1/64 from one message to another. FRLEN is valid after RSI. See chapter “Support for Frequency Drift Analysis in Frames with Pause Pulse”.
0	7:6, 15:12	r	**Reserved** Read as 0; should be written with 0.

Serial Data and Status Register x

The Serial (Receive) Data and Status Register provides the data and status information of channel x.

SDSx (x=0-24)

Serial Data and Status Register x **(110_H+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CON	**0**	**SCRC**						**MID**							
r	r	r						r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD															
r															

Field	Bits	Type	Description
SD	15:0	r	**Serial Data** of last serial data frame. SD0 is on bit position 0. Usually all 16 data bits are used. If RCR.ESF is cleared 8 bits of data are available and bits [15:8] are zero. If RCR.ESF is set and if SDS.CON is cleared 12 bits of data are available and bits [15:12] are zero.
MID	23:16	r	**Message ID** of last serial data frame. ID0 is on bit position 16. If RCR.ESF is cleared, or if SDS.CON is set, bits [23:20] are zero.
SCRC	29:24	r	**SCRC** CRC of last serial data frame. CRC0 is on position 24. If RCR.ESF is cleared, bits [29:28] are always zero.
CON	31	r	**Configuration bit** of last serial frame. 0_B 12-bit data and 8-bit message ID 1_B 16-bit data and 4-bit message ID
0	30	r	**Reserved** Read as 0; should be written with 0.

39.4.4 Input and Output Control

Input and Output Control Register x

The Input and Output Control Register IOCRx determines for the SENT channel x:

for the receiver:

- the alternate input
- the filter depth
- the input signal polarity

for the SPC Unit

- the trigger source
- the output signal polarity

IOCRx (x=0-24)

Input and Output Control Register x **(114_H+x*40_H)** **Application Reset Value: X000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXM	RXM	TRM	CTR	EC								ETS			
rh	rh	rh	rw	rh								rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFEG	CREG	FEG	REG	0	CEC	IIE	OIE	DEPTH				0		ALTI	
rw	rw	rh	rh	r	w	rw	rw	rw				r		rw	

Field	Bits	Type	Description
ALTI	1:0	rw	**Alternate Input Select** Selects the alternate input for channel y: 00_B Alternate Input 0 selected 01_B Alternate Input 1 selected 10_B Alternate Input 2 selected 11_B Alternate Input 3 selected
DEPTH	7:4	rw	**Digital Glitch Filter Depth** DEPTH determines the number of port input samples clocked with f_{pdiv} that are taken into account for the calculation of the floating average. The higher DEPTH is chosen to be, the longer the glitches that are suppressed and the longer the delay of the input signal introduced by this filter. 0_H off, default 1_H 1 T_{pdiv} 2_H 2 … F_H 15
OIE	8	rw	**Output Inverter Enable Channel x** Selects the Pulse Polarity of the output of channel x 0_B Pulse polarity is active low 1_B Pulse polarity is active high

Field	Bits	Type	Description
IIE	9	rw	**Input Inverter Enable Channel x** Selects the Pulse Polarity of the input of channel x 0_B Pulse polarity is active low 1_B Pulse polarity is active high
CEC	10	w	**Clear Edge Counter** If this bit is set, IOCR.EC is cleared. Always reads back as '0'.
REG	12	rh	**Rising Edge Glitch Flag for Channel x** Shows the status of the glitch detection of channel x REG is cleared by setting CREG. 0_B No Glitch detected on rising edge 1_B Glitch detected on rising edge
FEG	13	rh	**Falling Edge Glitch Flag for Channel x** Shows the status of the glitch detection of channel x FEG is cleared by setting CFEG. 0_B No Glitch detected on falling edge 1_B Glitch detected on falling edge
CREG	14	rw	**Clear Rising Edge Glitch Flag for Channel x** Clears the status flag REG CREG always read zero. 0_B REG is not cleared 1_B REG is cleared
CFEG	15	rw	**Clear Falling Edge Glitch Flag for Channel x** Clears the status flag FEG CFEG always read zero. 0_B FEG is not cleared 1_B FEG is cleared
ETS	19:16	rw	**External Trigger Select** Selects the external trigger line if SCRx.TRIG is programmed to 11B. In some products, not all inputs are connected. See **Chapter 39.3.9.1**. 0_H TRIG0 … F_H TRIG15
EC	27:20	rh	**Edge Counter** This bit field contains a counter with saturation (stops at 0xFF). It is incremented with any falling edge that appears on the input pin selected by IOCR.ALTI. Note that this holds true in all states (STOP, INITIALIZED, RUNNING, SYNCHRONIZED). It is intended for debugging, in particular to find a bubbling idiot that sends before enabling the module.
CTR	28	rw	**Clear Trigger Monitor Flag for Channel x** Clears the status flag TRM CTR always read zero. Reset value of CTR is 0. 0_B TRM is not cleared 1_B TRM is cleared

Field	Bits	Type	Description
TRM	29	rh	**Trigger Monitor Flag for Channel x** Shows the status of the trigger detection of channel x TRM is cleared by setting CTR. Reset value of TRM is 0. 0_B No Trigger detected 1_B Trigger detected (one or several)
RXM	30	rh	**Receive Monitor for Channel x** Shows the status of the receive signal of channel x after glitch filtering and inverted as specified by IIE. Reset value of RXM is X. 0_B Current signal is low. 1_B Current signal is high.
TXM	31	rh	**Transmit Monitor for Channel x** Shows the status of the transmit signal of channel x inverted as specified by OIE. Reset value of TXM is X. 0_B Current signal is low. 1_B Current signal is high.
0	3:2, 11	r	**Reserved** Read as 0; should be written with 0.

39.4.5 Receive Data Registers

Receive Data Register x

The Receive Data Registers RDRx for channel x shows the data content of a received data frame. Register VIEWx is used to sort the nibbles.

Register VIEW must be set up correctly to see all data nibbles of the frame! By default the application software set VIEW to 7654 3210_H.

The internal receive buffer is always cleared (0x0000 0000_H) at each frame start. Thus unused nibbles are always read as zero.

RDRx (x=0-24)

Receive Data Register x **(080_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RD7				RD6				RD5				RD4			
rh				rh				rh				rh			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD3				RD2				RD1				RD0			
rh				rh				rh				rh			

Field	Bits	Type	Description
RDy (y=0-7)	4*y+3:4*y	rh	**Receive Data Nibble y** RDy shows the nibble from the received frame that is sorted to this position. It can be selected by any of VIEWx.RDNPy (y = 0-7). By default all nibbles are sorted to RD0 as the reset value of VIEW is 0x0000 0000H. I.e. at the end of frame reception RD0 contains the last data nibble of the frame.

Receive Time Stamp Register x

The SENT Channel x Receive Time Stamp Register contains read-only information about the time the last frame for channel x was received. Time is captured with the second falling edge, i.e. at the Status/Communication data pulse.

RTSx (x=0-24)

Receive Time Stamp Register x **($A80_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LTS															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTS															
r															

Field	Bits	Type	Description
LTS	31:0	r	**Last Receive Time Stamp for Channel x** This bit field shows the time stamp of the last frame on channel x.

Receive Data View Register x

The Receive Data View Registers VIEWx stores the nibble pointers. They determine the sequence in which the received data nibbles are presented to the host. This reduces the SW effort to sort the nibbles. The data nibble that is received first in the received frame is moved to the location given in register VIEW.RDNP0, the second to VIEW.RDNP1 and so on until VIEW.RDNP7. If more than one VIEW.RDNPx point to a certain location in RDR, the last one will overwrite the previous ones. Example: two 12 bit values are transmitted. One with most significant nibble first and one with least significant nibble first. The frame looks like this: 456321_H. Note that the 1 is received as first data nibble and the 4 comes in as last data nibble. The actual signal values are $0x123_H$ and $0x456_H$. By using VIEWx this can be sorted out into two 16bit values by HW. In the example VIEWx would be set to: 73 654 012_H. 73 is a dummy value and is not regarded if not more than 6 nibbles are received in a frame. The Register RDRx looks like this: 0x0456 0123_H to the host.

In the example RDR nibbles 3 and 7 contain $0x0000_B$ as the Receive Buffer is always cleared (0x0000 0000_H) before new data is received.

If a frame contains more than eight nibbles and the sorting can not be specified statically, VIEW can be set to e.g. 7654 3210_H.

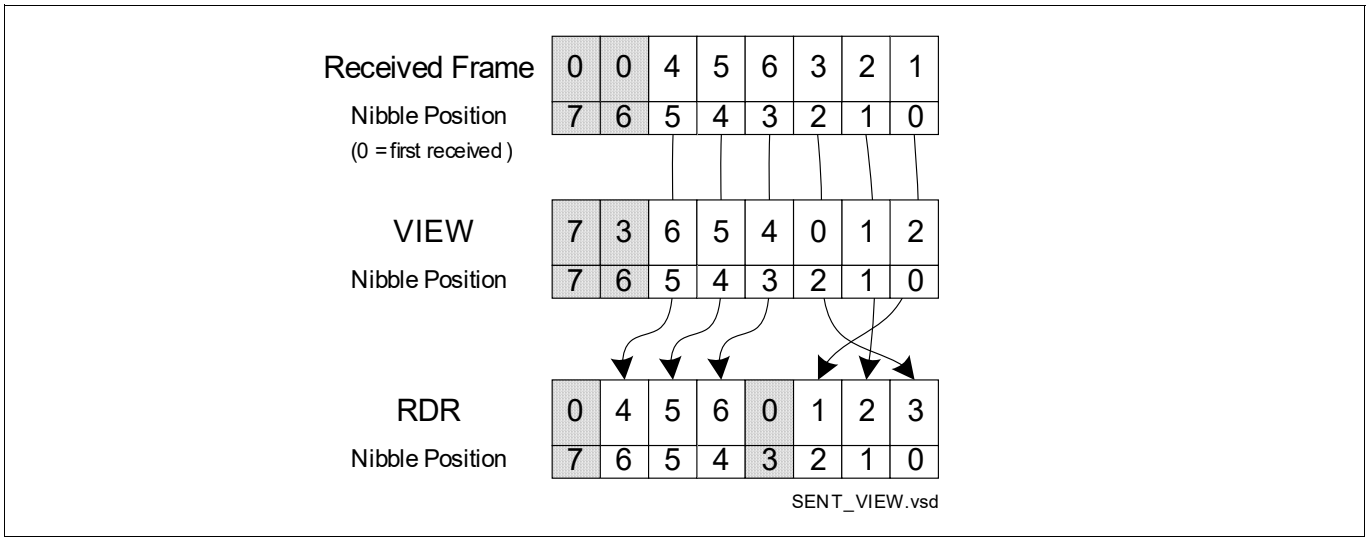


Figure 578 Functionality of VIEW Register

VIEWx (x=0-24)

Receive Data View Register x **($11C_H$+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**RDNP7**		**0**		**RDNP6**		**0**		**RDNP5**		**0**		**RDNP4**	
r		rw		r		rw		r		rw		r		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**RDNP3**		**0**		**RDNP2**		**0**		**RDNP1**		**0**		**RDNP0**	
r		rw		r		rw		r		rw		r		rw	

Field	Bits	Type	Description
RDNPy (y=0-7)	4*y+2:4*y	rw	**Receive Data Target Nibble Pointer y** RDNPy points to the Nibble in Receive Data Register RDRx where the nibble y from the received frame is sorted to. Nibble 0 is the first data nibble in the frame. It gets moved to the position defined in RDNP0. And on. RDNPy must be written before first frame reception. All RDNPy must have different values. (Higher RDNPy overwrite lower RDNPy.) 000_B Nibble 0 selected … 111_B Nibble 7 selected
0	3, 7, 11, 15, 19, 23, 27, 31	r	**Reserved** Read as 0; should be written with 0.

39.4.6 SPC Control

SPC Control Register x

The SPC Control Register SCR contains data to be transmitted during the sync pulse of the data frames. It contains as well the trigger control bits required for sending nibbles from the SENT module to the sensor / external SENT device.

The SPC Control Register is used to control the trigger mode and time base of the SPC channel transmission.

Data and the control bits are collected in this single register to ease transfer of multiple pulses in cases where dynamic switching of the trigger condition is required.

The SPC Control Register is used to control single pulse transfers only.

Thus it is possible to change the control settings of an individual channel from pulse to pulse as it is required in SPC mode “Bidirectional transmit”. Here it might be considered useful to change trigger mode between Mode 1 (immediately) and Mode 2 (falling edge of next Synchronization / Calibration Pulse with programmable delay).

In addition repeating transfers with the same control settings are supported. For a pulse transfer to be initiated a synchronization signal is sufficient and no further SW intervention is required. Here the data can be changed (“ID-Selection” Mode) or simply left constant (“Sync” Mode). Only a HW trigger needs to be set up.

In Mode 0, SPC is deactivated for this channel. A write access to SCRx does not initiate an SPC pulse transmission. All state transmitter machines are initialized.

In Mode 1, an SPC pulse is sent, each time SPC Control Register SCRx is written to. If a transfer is ongoing, the channel waits automatically until the internal transmission register is ready. Transmit Buffer Underflow bit TBIx is set after data has been completely transferred (PLEN exceeded) and no new data was written to SCRx. After the data was transferred to the internal transmission register, interrupt TDIx signals that a new value can be written. INTSTATx.TDIx must be cleared by SW. Independently from this interrupt pending bit, a new interrupt pulse is generated on each transfer of an SPC pulse. This mode is important for back to back transfers of several nibbles as in bidirectional SPC mode.

In Mode 2, an SPC pulse is sent, each time the first falling edge of any Synchronization / Calibration Pulse is received. In this mode, the programmable delay DEL is most useful. In SPC mode “Bidirectional Transmit” this mode is useful to synchronize the transmission. TDIx and TBIx work as in Mode 1.

In Mode 3, an SPC pulse is sent (with current DEL and PLEN) after each external trigger event (defined by IOCRx.ETS). This is most useful in SPC “sync” mode. TDIx and TBIx work as in Mode 1.

SCRx (x=0-24)

SPC Control Register x **(118_H+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRQ	BASE	DEL						TRIG		PLEN					
r	rw	rw						rw		rw					

Field	Bits	Type	Description
PLEN	5:0	rw	**Pulse Length** Defines the length of the pulse in tick times. The time base is the measured tick time of the latest received frame if selected so by BASE. In case this measured tick time was invalid or not already available after enable of the channel, the nominal time base of the module is used. 00_H Pulse length is 0 ticks … $3F_H$ Pulse length is 63 ticks
TRIG	7:6	rw	**Trigger Source and Mode Selection** Selects the Trigger Source and Mode. The internal sender state machine can be initialized by switching the channel off (TRIG is cleared) and on. This does not change the current register content. 00_B No Pulse is generated, OFF When cleared, an ongoing transfer is stopped immediately and the transmit output is driven recessive. 01_B Pulse starts immediately (no auto repetition) 10_B Pulse starts each time the first falling edge of any Synchronization / Calibration Pulse is received (auto repetition on next Sync. / Cal. Pulses) 11_B Pulse starts after each external trigger event. (auto repetition on next trigger) IOCRx.ETS selects the source of this event.
DEL	13:8	rw	**Delay Length** Selects how long the SPC pulse is delayed after the trigger condition. The time base is the measured tick time of the latest received frame if selected so by BASE. In case this measured tick time was invalid or not already available after enable of the channel, the nominal time base of the module is used. 00_H Pulse is not delayed 01_H Pulse is delayed by 1 tick 02_H Pulse is delayed by 2 ticks … $3F_H$ Pulse is delayed by 63 ticks
BASE	14	rw	**Time Base** Selects the Pulse Time Base 0_B Pulse is based on measured frequency of last Synchronization/Calibration Pulse 1_B Pulse is based on nominal frequency
TRQ	15	r	**Transfer Request in Progress** While an SPC Pulse is being sent this bit is set. Write access is ignored.
0	31:16	r	**Reserved** Read as 0; should be written with 0.

39.4.7 Interrupt Control Registers

Interrupt Overview Register

INTOV
Interrupt Overview Register (014$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							IPC24	IPC23	IPC22	IPC21	IPC20	IPC19	IPC18	IPC17	IPC16
r							rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPC15	IPC14	IPC13	IPC12	IPC11	IPC10	IPC9	IPC8	IPC7	IPC6	IPC5	IPC4	IPC3	IPC2	IPC1	IPC0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
IPCy (y=0-24)	y	rh	**Interrupt Pending on Channel y** If any interrupt requested flag is set for channel y in register INTSTATy AND the referring interrupt is enabled in INTENx then IPCy is set. It is automatically reset if all flags in INTSTATy are cleared for which the referring interrupt is enabled in INTENx. *Note: Not all IPC0-24 are available on all products, the number of Interrupt Pending on Channel is equivalent to the SENT channels available, e.g 4 SENT channels has 4 Interrupt Pending IPC0-3 and vice versa.*
0	31:25	r	**Reserved** Read as 0.

Interrupt Status Register x

The Interrupt Status Register INTSTATx contains status bits that show the status of any interrupt of SENT channel x.

The bits are set independently from the referring Interrupt Enable in Register INTENx. Thus they can be used as status bits as well e.g. by a SW based on polling.

INTSTATx (x=0-24)
Interrupt Status Register x (120$_H$+x*40$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		WDI	SCRI	SDI	WSI	CRCI	NVI	NNI	FDI	FRI	TBI	TDI	RBI	RDI	RSI
r		rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RSI	0	rh	**Receive Success Interrupt Request Flag** This bit is set at the successfully received end of a frame. Depending on bit RCRx.CDIS this indicates a successful check of the CRC. This bit can be cleared by bit INTCLRx.RSI. This bit can be set by bit INTSETx.RSI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
RDI	1	rh	**Receive Data Interrupt Request Flag** RDI is activated when a received frame is moved to a Receive Data Register RDR. Both RDI and RSI will be issued together in normal use cases where the frame size is not bigger than 8 nibbles and CRC is correct or not checked (if RCRx.CDIS is cleared). For frames with more than 8 nibbles RDI is issued after 8 nibbles. After the last nibbles of the frame are received, a last RDI and an RSI are issued together. Note that this is true independently from RCR.FDFL. This bit can be cleared by bit INTCLRx.RDI. This bit can be set by bit INTSETx.RDI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
RBI	2	rh	**Receive Buffer Overflow Interrupt Request Flag** This bit is set after a frame has been received while the old one was not read from RDRx. I.e. the kernel wants to set any of the two interrupts RSI and RDI and finds any of these two interrupts already set.The old data is overwritten by the new data. This bit is NOT cleared by reading RDRx. This bit can be cleared by bit INTCLRx.RBI. This bit can be set by bit INTSETx.RBI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
TDI	3	rh	**Transfer Data Interrupt Request Flag** This bit is set after the trigger condition was detected. Data to be transferred has been moved internally. Thus a new value can be written to SCRx. This can be used for back to back transfers. This bit is automatically cleared by writing SCRx. This bit can be cleared by bit INTCLRx.TDI. This bit can be set by bit INTSETx.TDI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

Field	Bits	Type	Description
TBI	4	rh	**Transmit Buffer Underflow Interrupt Request Flag** This bit is set after data has been completely transferred (PLEN exceeded) and no new data was written to SCRx. This bit is NOT cleared by writing SCRx. This bit can be cleared by bit INTCLRx.TBI. This bit can be set by bit INTSETx.TBI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
FRI	5	rh	**Frequency Range Interrupt Request Flag** This bit is set after a Synchronization / Calibration pulse was received that deviates more than +/- 25% from the nominal value. The referring data is ignored. This bit can be cleared by bit INTCLRx.FRI. This bit can be set by bit INTSETx.FRI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
FDI	6	rh	**Frequency Drift Interrupt Request Flag** This bit is set after a subsequent Synchronization / Calibration pulse was received that deviates more than 1.5625% (1/64) from its predecessor. (See RCR.CFC) This bit can be cleared by bit INTCLRx.FDI. This bit can be set by bit INTSETx.FDI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
NNI	7	rh	**Number of Nibbles Wrong Request Flag** This bit is set after more nibbles have been received than expected or a Synchronization / Calibration Pulse is received too early thus too few nibbles have been received. This bit can be cleared by bit INTCLRx.NNI. This bit can be set by bit INTSETx.NNI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
NVI	8	rh	**Nibbles Value out of Range Request Flag** This bit is set after a too long or too short nibble pulse has been received. I.e. value < 0 or value > 15. This bit can be cleared by bit INTCLRx.NVI. This bit can be set by bit INTSETx.NVI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

Field	Bits	Type	Description
CRCI	9	rh	**CRC Error Request Flag** This bit is set if the CRC fails. This bit can be cleared by bit INTCLRx.CRCI. This bit can be set by bit INTSETx.CRCI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
WSI	10	rh	**Wrong Status and Communication Nibble Error Request Flag** In Short Serial Frame Mode (RCR.ESF is cleared), this bit is set if the Status and Communication nibble shows a start bit in a frame other than frame number n x 16. In Enhanced Serial Frame Mode this bit is without function. This bit can be cleared by bit INTCLRx.WSI. This bit can be set by bit INTSETx.WSI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
SDI	11	rh	**Serial Data Receive Interrupt Request Flag** This bit is set after all serial data bits have been received via the Status and Communication nibble. Depending on bit RCRx.SCDIS this indicates a successful check of the CRC. This bit can be cleared by bit INTCLRx.SDI. This bit can be set by bit INTSETx.SDI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
SCRI	12	rh	**Serial Data CRC Error Request Flag** This bit is set if the CRC of the serial message fails. In Enhanced Serial Message Format, this includes a check of the Serial Communication Nibble for correct 0 values of bit 3 in frames 7, 13 and 18. This bit can be cleared by bit INTCLRx.SCRI. This bit can be set by bit INTSETx.SCRI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
WDI	13	rh	**Watch Dog Error Request Flag** This bit is set if the Watch Dog Timer of the channel x expires. This bit can be cleared by bit INTCLRx.WDI. This bit can be set by bit INTSETx.WDI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
0	31:14	r	**Reserved** Read as 0.

Interrupt Set Register x

The Interrupt Set Register INTSETx contains control bits that trigger an interrupt pulse for any interrupt of SENT channel x.

INTSETx (x=0-24)
Interrupt Set Register x **(124_H+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		WDI	SCRI	SDI	WSI	CRCI	NVI	NNI	FDI	FRI	TBI	TDI	RBI	RDI	RSI
r		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RSI	0	w	**Set Interrupt Request Flag RSI** Setting this bit set bit INTSTATx.RSI. Clearing this bit has no effect. Reading this bit returns always zero.
RDI	1	w	**Set Interrupt Request Flag RDI** Setting this bit set bit INTSTATx.RDI. Clearing this bit has no effect. Reading this bit returns always zero.
RBI	2	w	**Set Interrupt Request Flag RBI** Setting this bit set bit INTSTATx.RBI. Clearing this bit has no effect. Reading this bit returns always zero.
TDI	3	w	**Set Interrupt Request Flag TDI** Setting this bit set bit INTSTATx.TDI. Clearing this bit has no effect. Reading this bit returns always zero.
TBI	4	w	**Set Interrupt Request Flag TBI** Setting this bit set bit INTSTATx.TBI. Clearing this bit has no effect. Reading this bit returns always zero.
FRI	5	w	**Set Interrupt Request Flag FRI** Setting this bit set bit INTSTATx.FRI. Clearing this bit has no effect. Reading this bit returns always zero.
FDI	6	w	**Set Interrupt Request Flag FDI** Setting this bit set bit INTSTATx.FDI. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
NNI	7	w	**Set Interrupt Request Flag NNI** Setting this bit set bit INTSTATx.NNI. Clearing this bit has no effect. Reading this bit returns always zero.
NVI	8	w	**Set Interrupt Request Flag NVI** Setting this bit set bit INTSTATx.NVI. Clearing this bit has no effect. Reading this bit returns always zero.
CRCI	9	w	**Set Interrupt Request Flag CRCI** Setting this bit set bit INTSTATx.CRCI. Clearing this bit has no effect. Reading this bit returns always zero.
WSI	10	w	**Set Interrupt Request Flag WSI** Setting this bit set bit INTSTATx.WSI. Clearing this bit has no effect. Reading this bit returns always zero.
SDI	11	w	**Set Interrupt Request Flag SDI** Setting this bit set bit INTSTATx.SDI. Clearing this bit has no effect. Reading this bit returns always zero.
SCRI	12	w	**Set Interrupt Request Flag SCRI** Setting this bit set bit INTSTATx.SCRI. Clearing this bit has no effect. Reading this bit returns always zero.
WDI	13	w	**Set Interrupt Request Flag WDI** Setting this bit set bit INTSTATx.WDI. Clearing this bit has no effect. Reading this bit returns always zero.
0	31:14	r	**Reserved** Read as 0; should be written with 0.

Interrupt Clear Register x

The Interrupt Clear Register INTCLRx contains control bits that clear the status of any interrupt of SENT channel x.

INTCLRx (x=0-24)

Interrupt Clear Register x **(128_H+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		WDI	SCRI	SDI	WSI	CRCI	NVI	NNI	FDI	FRI	TBI	TDI	RBI	RDI	RSI
r		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RSI	0	w	**Clear Interrupt Request Flag RSI** Setting this bit clears bit INTSTATx.RSI. Clearing this bit has no effect. Reading this bit returns always zero.
RDI	1	w	**Clear Interrupt Request Flag RDI** Setting this bit clears bit INTSTATx.RDI. Clearing this bit has no effect. Reading this bit returns always zero.
RBI	2	w	**Clear Interrupt Request Flag RBI** Setting this bit clears bit INTSTATx.RBI. Clearing this bit has no effect. Reading this bit returns always zero.
TDI	3	w	**Clear Interrupt Request Flag TDI** Setting this bit clears bit INTSTATx.TDI. Clearing this bit has no effect. Reading this bit returns always zero.
TBI	4	w	**Clear Interrupt Request Flag TBI** Setting this bit clears bit INTSTATx.TBI. Clearing this bit has no effect. Reading this bit returns always zero.
FRI	5	w	**Clear Interrupt Request Flag FRI** Setting this bit clears bit INTSTATx.FRI. Clearing this bit has no effect. Reading this bit returns always zero.
FDI	6	w	**Clear Interrupt Request Flag FDI** Setting this bit clears bit INTSTATx.FDI. Clearing this bit has no effect. Reading this bit returns always zero.
NNI	7	w	**Clear Interrupt Request Flag NNI** Setting this bit clears bit INTSTATx.NNI. Clearing this bit has no effect. Reading this bit returns always zero.
NVI	8	w	**Clear Interrupt Request Flag NVI** Setting this bit clears bit INTSTATx.NVI. Clearing this bit has no effect. Reading this bit returns always zero.
CRCI	9	w	**Clear Interrupt Request Flag CRCI** Setting this bit clears bit INTSTATx.CRCI. Clearing this bit has no effect. Reading this bit returns always zero.
WSI	10	w	**Clear Interrupt Request Flag WSI** Setting this bit clears bit INTSTATx.WSI. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
SDI	11	w	**Clear Interrupt Request Flag SDI** Setting this bit clears bit INTSTATx.SDI. Clearing this bit has no effect. Reading this bit returns always zero.
SCRI	12	w	**Clear Interrupt Request Flag SCRI** Setting this bit clears bit INTSTATx.SCRI. Clearing this bit has no effect. Reading this bit returns always zero.
WDI	13	w	**Clear Interrupt Request Flag WDI** Setting this bit clears bit INTSTATx.WDI. Clearing this bit has no effect. Reading this bit returns always zero.
0	31:14	r	**Reserved** Read as 0; should be written with 0.

Interrupt Enable Register x

The Interrupt Enable Register INTENx contains control bits that enable the interrupt source of any interrupt of SENT channel x.

The Interrupt Status bits in register INTSTATx are set independently from the Interrupt Enable in Register INTENx.

INTENx (x=0-24)
Interrupt Enable Register x **($12C_H$+x*40_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		WDI	SCRI	SDI	WSI	CRCI	NVI	NNI	FDI	FRI	TBI	TDI	RBI	RDI	RSI
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RSI	0	rw	**Enable Interrupt Request RSI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RDI	1	rw	**Enable Interrupt Request RDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RBI	2	rw	**Enable Interrupt Request RBI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TDI	3	rw	**Enable Interrupt Request TDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source

Field	Bits	Type	Description
TBI	4	rw	**Enable Interrupt Request TBI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
FRI	5	rw	**Enable Interrupt Request FRI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
FDI	6	rw	**Enable Interrupt Request FDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
NNI	7	rw	**Enable Interrupt Request NNI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
NVI	8	rw	**Enable Interrupt Request NVI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
CRCI	9	rw	**Enable Interrupt Request CRCI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
WSI	10	rw	**Enable Interrupt Request WSI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
SDI	11	rw	**Enable Interrupt Request SDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
SCRI	12	rw	**Enable Interrupt Request SCRI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
WDI	13	rw	**Enable Interrupt Request WDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
0	31:14	r	**Reserved** Read as 0; should be written with 0.

Interrupt Node Pointer Register x

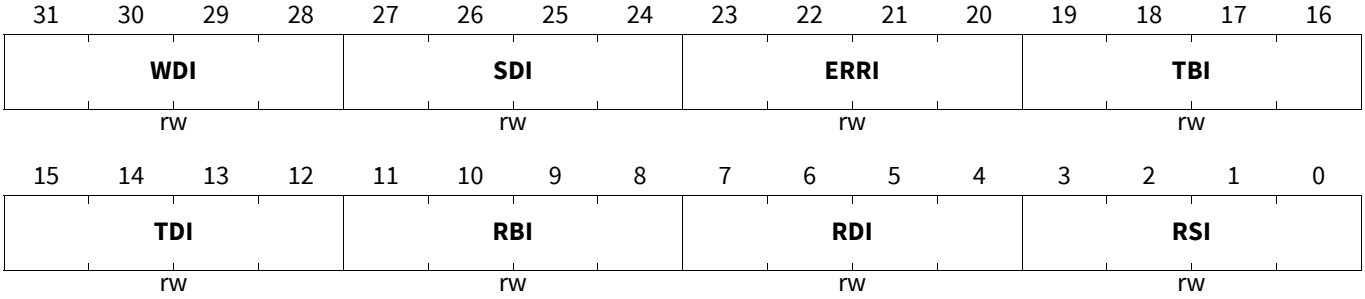
The Interrupt Node Pointer Register INPx contains the node pointers of SENT channel x.

Node Pointer ERRI is one single node pointer for the following error interrupts:

- FRI
- FDI
- NNI
- NVI
- CRCI
- WSI
- SCRI

INPx (x=0-24)

Interrupt Node Pointer Register x **(130_H+x*40_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RSI	3:0	rw	**Interrupt Node Pointer for Interrupt RSI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.RSI (if enabled by bit INTENx.RSI). 0_H Trigger Output TRIGO0 is selected … F_H Trigger Output TRIGO15 is selected
RDI	7:4	rw	**Interrupt Node Pointer for Interrupt RDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.RDI (if enabled by bit INTENx.RDI). For bit field definition, see RSI.
RBI	11:8	rw	**Interrupt Node Pointer for Interrupt RBI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.RBI (if enabled by bit INTENx.RBI). For bit field definition, see RSI.
TDI	15:12	rw	**Interrupt Node Pointer for Interrupt TDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.TDI (if enabled by bit INTENx.TDI). For bit field definition, see RSI.
TBI	19:16	rw	**Interrupt Node Pointer for Interrupt TBI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.TBI (if enabled by bit INTENx.TBI). For bit field definition, see RSI.
ERRI	23:20	rw	**Interrupt Node Pointer for Interrupt FRI, FDI, NNI, NVI, CRCI, WSI, SCRI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.FRI (if enabled by bit INTENx.FRI) or INTSTATx.FDI (if enabled by bit INTENx.FDI) or INTSTATx.NNI (if enabled by bit INTENx.NNI) or INTSTATx.NVI (if enabled by bit INTENx.NVI) or INTSTATx.CRCI (if enabled by bit INTENx.CRCI) or INTSTATx.WSI (if enabled by bit INTENx.WSI) or INTSTATx.SCRI (if enabled by bit INTENx.SCRI) For bit field definition, see RSI.

Field	Bits	Type	Description
SDI	27:24	rw	**Interrupt Node Pointer for Interrupt SDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.SDI (if enabled by bit INTENx.SDI). For bit field definition, see RSI.
WDI	31:28	rw	**Interrupt Node Pointer for Interrupt WDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.WDI (if enabled by bit INTENx.WDI). For bit field definition, see RSI.

39.4.8 Bus Control Interface Registers

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock, sleep mode and disable mode for the module.

Notes

1. *After a hardware reset operation, the f_{SENT} and $f_{fracdiv}$ clocks are switched off and the SENT module is disabled (DISS set).*

CLC

Clock Control Register (000_H) Application Reset Value: 0000 0003$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

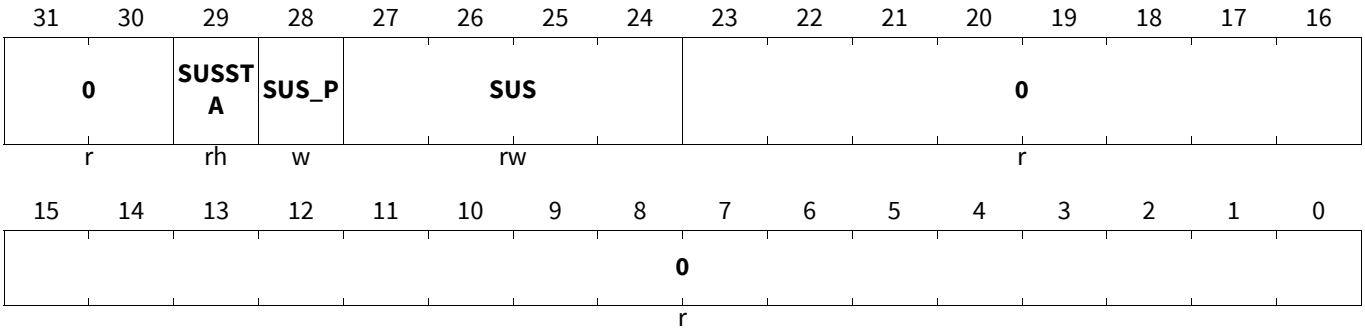
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode.
0	2, 31:4	r	**Reserved** Read as 0; should be written with 0.

OCDS Control and Status

The OCDS Control and Status (OCS) register is cleared by Debug Reset. The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32bit wide only and requires Supervisor Mode.

OCS

OCDS Control and Status (0E8$_H$) Debug Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. (SPC low pulse breaks immediately, Port Pin keeps the last value if suspend state is entered) 2_H Soft suspend (SPC low pulse will be finished before suspend state is entered) **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended (for channels where SUSEN is set)
0	23:0, 31:30	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access[1] for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, ... , EN31 -> TAG ID 011111B.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

ACCEN0

Access Enable Register 0 **(0FC$_H$)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the TC3xx devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, ... , EN31 -> TAG ID 111111B.

ACCEN1

Access Enable Register 1 **(0F8$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0

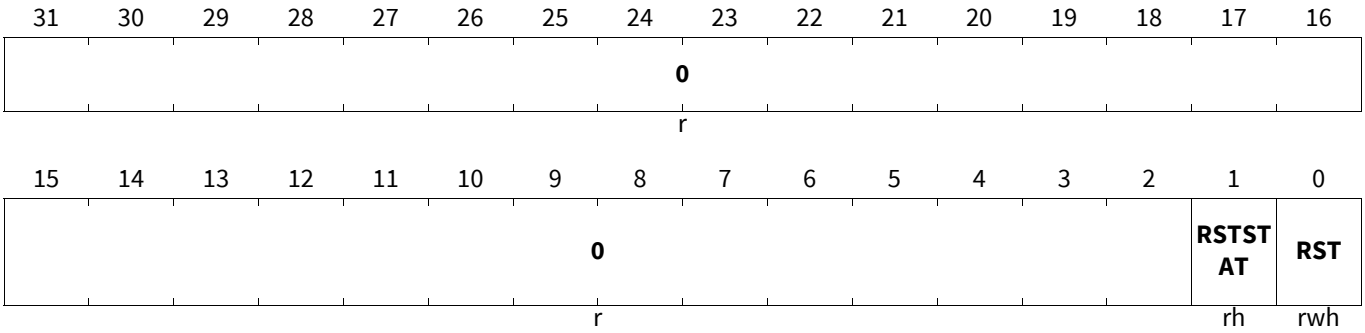
The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLE.CLR register bit.

During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0

Kernel Reset Register 0 **($0F4_H$)** **Application Reset Value: $0000\ 0000_H$**



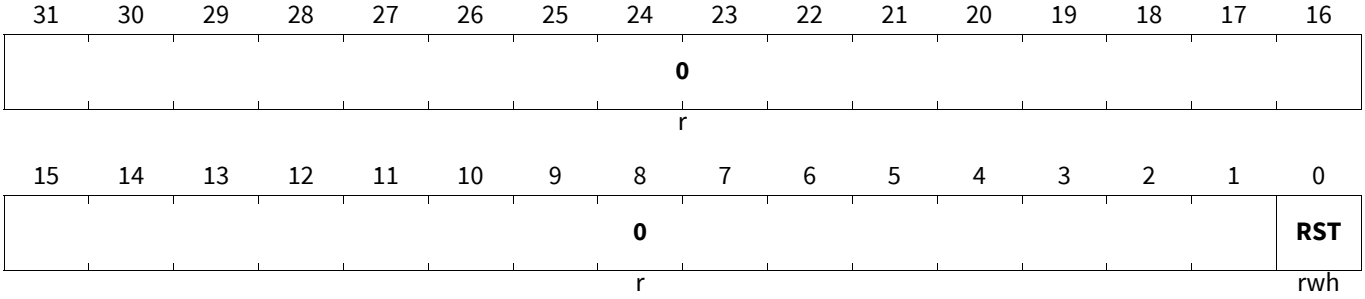
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1

Kernel Reset Register 1 **($0F0_H$)** **Application Reset Value: $0000\ 0000_H$**



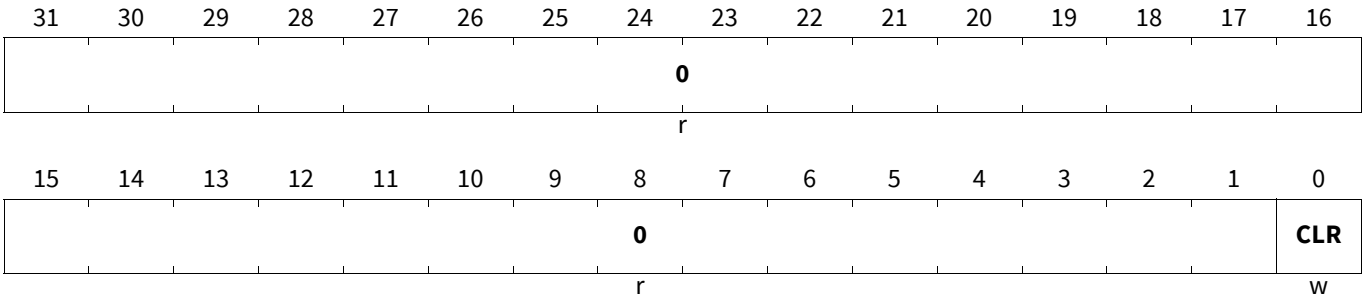
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **($0EC_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

39.5 IO Interfaces

The table below lists all the interfaces of the SENT to other modules.

Table 352 List of SENT Interface Signals

Interface Signals	I/O	Description
TRIG(15:0)	in	**GTM timer output vector**
TRIGO(9:0)	out	**SENT TRIGm Service Request**
SPC(24:0)	out	**Transmit output** The number indicates the channel.
SENT0A	in	**Receive input channel 0**
SENT0B		
SENT0C		
SENT0D		
RXD0(4)		
SENT1A	in	**Receive input channel 1**
SENT1B		
SENT1C		
SENT1D		
RXD1(4)		
SENT2A	in	**Receive input channel 2**
SENT2B		
SENT2C		
SENT2D		
RXD2(4)		
SENT3A	in	**Receive input channel 3**
SENT3B		
SENT3C		
SENT3D		
RXD3(4)		
SENT4A	in	**Receive input channel 4**
SENT4B		
SENT4C		
SENT4D		
RXD4(4)		
SENT5A	in	**Receive input channel 5**
SENT5B		
SENT5C		
SENT5D		
RXD5(4)		

Table 352 List of SENT Interface Signals (cont'd)

Interface Signals	I/O	Description
SENT6A	in	**Receive input channel 6**
SENT6B		
SENT6C		
SENT6D		
RXD6(4)		
SENT7A	in	**Receive input channel 7**
SENT7B		
SENT7C		
SENT7D		
RXD7(4)		
SENT8A	in	**Receive input channel 8**
SENT8B		
SENT8C		
SENT8D		
RXD8(4)		
SENT9A	in	**Receive input channel 9**
SENT9B		
SENT9C		
SENT9D		
RXD9(4)		
SENT10A	in	**Receive input channel 10**
SENT10B		
SENT10C		
SENT10D		
RXD10(4)		
SENT11A	in	**Receive input channel 11**
SENT11B		
SENT11C		
SENT11D		
RXD11(4)		
SENT12A	in	**Receive input channel 12**
SENT12B		
SENT12C		
SENT12D		
RXD12(4)		

Table 352 List of SENT Interface Signals (cont'd)

Interface Signals	I/O	Description
SENT13A	in	**Receive input channel 13**
SENT13B		
SENT13C		
SENT13D		
RXD13(4)		
SENT14A	in	**Receive input channel 14**
SENT14B		
SENT14C		
SENT14D		
RXD14(4)		
SENT15A	in	**Receive input channel 15**
SENT15B		
SENT15C		
SENT15D		
RXD15(4)		
SENT16A	in	**Receive input channel 16**
SENT16B		
SENT16C		
SENT16D		
RXD16(4)		
SENT17A	in	**Receive input channel 17**
SENT17B		
SENT17C		
SENT17D		
RXD17(4)		
SENT18A	in	**Receive input channel 18**
SENT18B		
SENT18C		
SENT18D		
RXD18(4)		
SENT19A	in	**Receive input channel 19**
SENT19B		
SENT19C		
SENT19D		
RXD19(4)		

Table 352 List of SENT Interface Signals (cont'd)

Interface Signals	I/O	Description
SENT20A	in	**Receive input channel 20**
SENT20B		
SENT20C		
SENT20D		
RXD20(4)		
SENT21A	in	**Receive input channel 21**
SENT21B		
SENT21C		
SENT21D		
RXD21(4)		
SENT22A	in	**Receive input channel 22**
SENT22B		
SENT22C		
SENT22D		
RXD22(4)		
SENT23A	in	**Receive input channel 23**
SENT23B		
SENT23C		
SENT23D		
RXD23(4)		
SENT24A	in	**Receive input channel 24**
SENT24B		
SENT24C		
SENT24D		
RXD24(4)		

39.6 Revision History

Table 353 Revision History

Reference	Change to Previous Version	Comment
V2.1.9		
Page 61	Cross References in Revision History corrected.	
Page 61	Revision History updated.	
V2.1.10		
Page 2	Message storage description updated, no functional change.	
–	"SENT specification J2716 JAN2010" replaced by "SAE J2716 042016" throughout document (14 times).	
Page 1	Information regarding backward compatiblity for SAE J2716 042016 added.	

40 CAN Interface (MCMCAN)

The MCMCAN implements Bosch M_CAN as CAN nodes. The M_CAN performs communication according to ISO 11898-1 and according to ISO 11898-4 (Time-triggered communication on CAN). The M_CAN provides all features of time-triggered communication specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation.In addition the M_CAN supports classical CAN and CAN FD communication according to ISO 11898-1. The CAN FD option can be used together with event-triggered and time-triggered CAN communication.

Figure 579 shows the AURIX™ TC3xx Platform specific implementation details and interconnections of the MCMCAN module. The I/O lines of the MCMCAN module (two I/O lines of each CAN node) are connected to the Ports listed in **Chapter 40.3.1.3**. The MCMCAN module is also supplied by clock control, interrupt control, and address decoding logic.

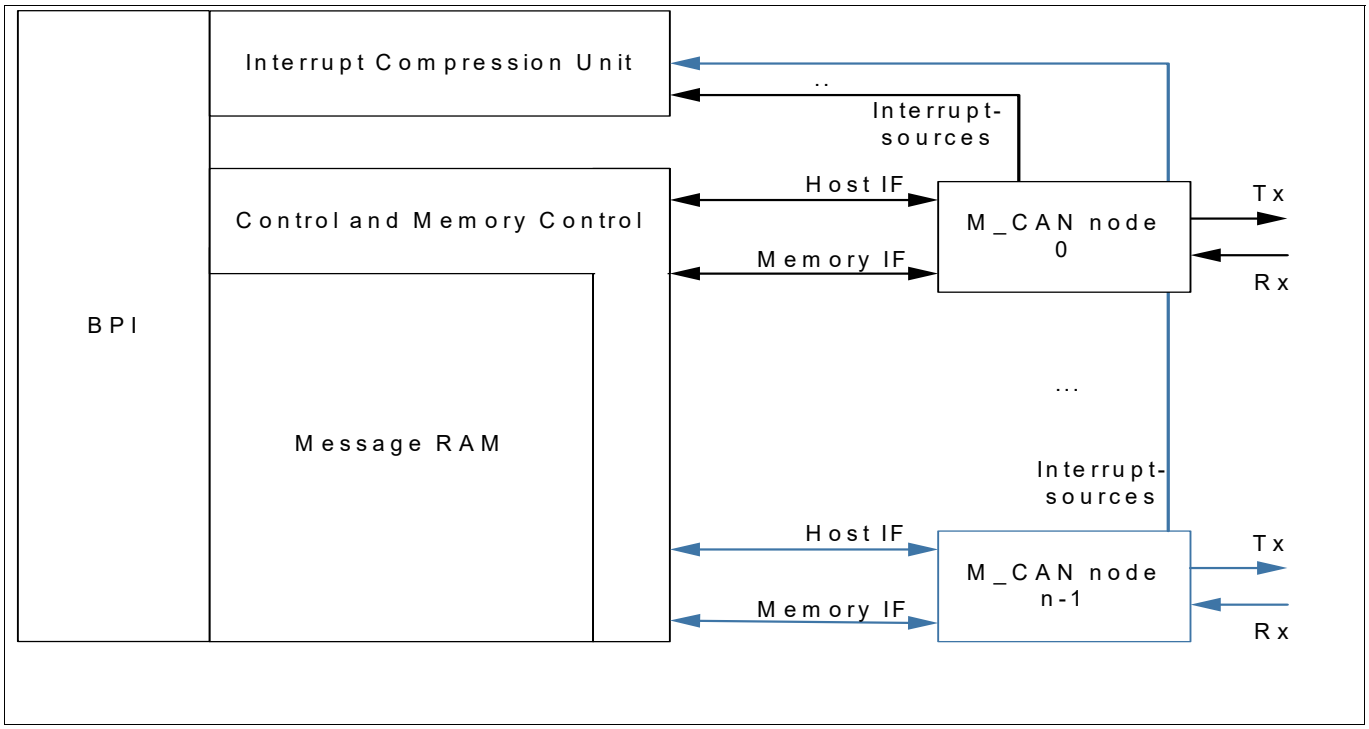


Figure 579 MCMCAN Module

40.1 Feature List

The following are the features of the MCMCAN module

- ISO 11898-1, -4
- CAN FD with up to 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware
- Event synchronized time-triggered communication supported
- CAN Error Logging
- AUTOSAR optimized
- SAE J1939 optimized
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signalling on reception of High Priority Messages
- Up to 4 CAN nodes
- Direct Message RAM access for Host CPU
- Multiple M_CANs share the same Message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- 8/16/32-bit Generic Slave Interface for connection customer-specific Host CPUs

Features offered by each M_CAN node:

- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Up to 64 dedicated Receive Buffers
- Up to 32 dedicated Transmit Buffers
- Configurable Transmit FIFO
- Configurable Transmit Queue
- Configurable Transmit Event FIFO

40.1.1 Delta to AURIX

MCMCAN is the new CAN interface, in which the user interface to the module is different than the previous MultiCAN+ CAN interface. The following are the notable changes:

- Message Objects are replaced by configurable Message RAM
- Supports Debug on CAN

40.2 Overview

An overview of an MCMCAN module is shown in **Figure 579**. The MCMCAN supports the following variants of CAN communication protocols.

- Classical CAN and CAN FD according to ISO 11898-1
- Time Triggered CAN according to ISO 11898-4

The MCMCAN consists of Bosch M_CAN as CAN nodes and a wrapper around the M_CAN called as the user interface. M_CAN provides the following functionality:

- CAN protocol Contoller
- Receive and transmit time stamp generation

- Transmit Handler
- Receive Handler

The user interface of MCMCAN provides the following functionality:

- A configurable Message RAM to store the message to be transmitted or received. The message RAM is shared by all the CAN nodes within a MCMCAN module
- Grouping and signalling of interrupts through Interrupt Compression Unit
- Clock selection and generation through Clock Control Block
- Access protection through BPI
- Timer based transmission of CAN frame and timeouts for reception of CAN frames.

Note: Refer appendix of a product variant for the number of MCMCAN modules, number of M_CAN nodes within a module and Message RAM allocation for individual nodes.

40.3 Functional Description

This section describes the functionality of MCMCAN. The functionality is described as three different sub-sections:

- User Interface
- M_CAN functionality
- TTCAN operation

40.3.1 MCMCAN User Interface

The MCMCAN User Interface describes about the clock generation, grouping of interrupts, external connections to the module and IO configurations.

40.3.1.1 MCMCAN Clockpaths

A general overview of clocks within the MCMCAN module.

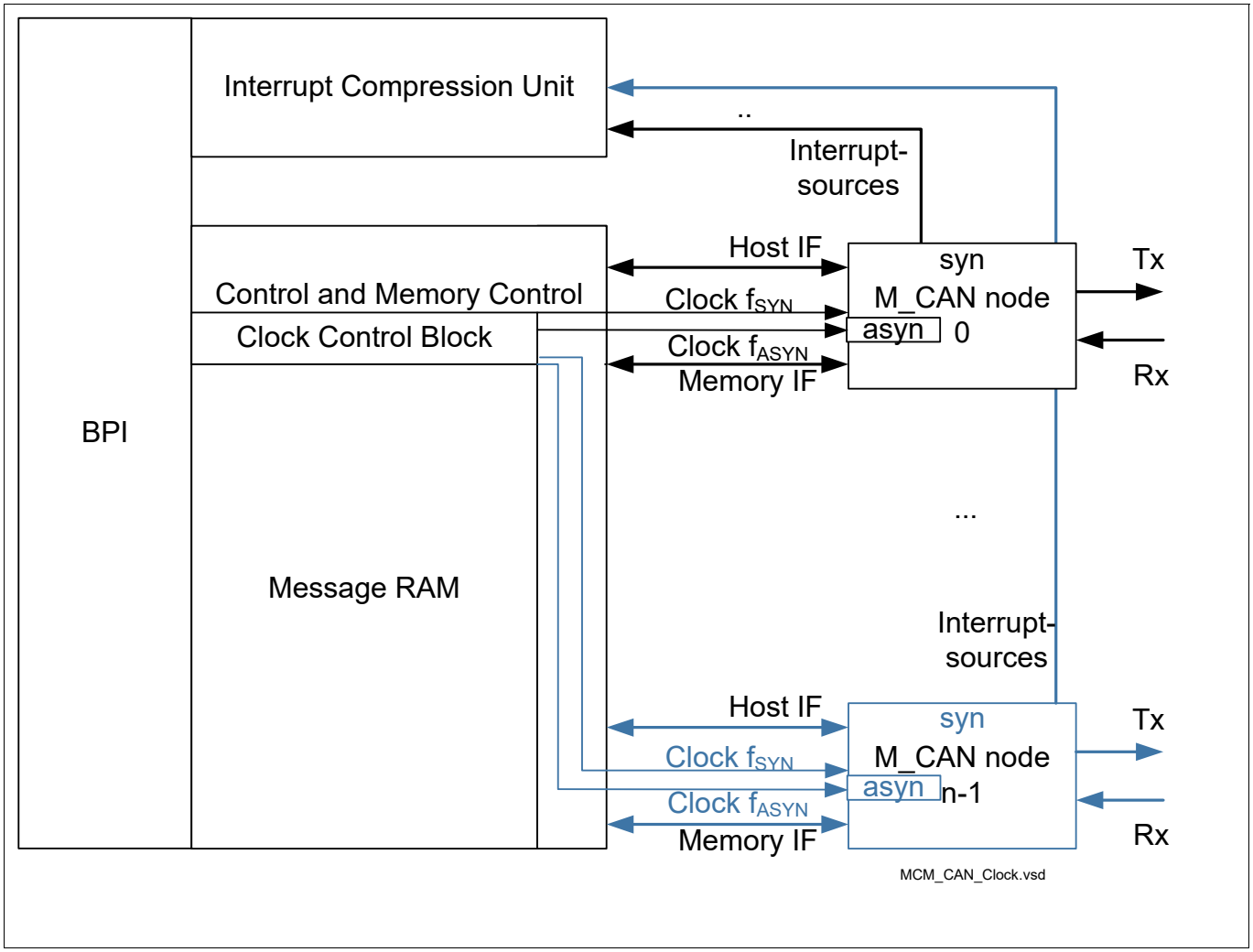


Figure 580 MCMCAN Clock Interconnects with CCU

The MCMCAN module clock inputs are connected to the Clock Control Unit (CCU). The CLC setting supplies the global module registers with its clocks. To supply the M_CAN nodes with the corresponding clocks, the **MCR**.CLKSELi registers have to be set. See **Figure 581** and **Table 354**. The asynchronous clock as well as the synchronous clock of each single M_CAN node can be switched on/off via MCR.CLKSELi register bitfields as shown in **Figure 582** and **Table 354** below.

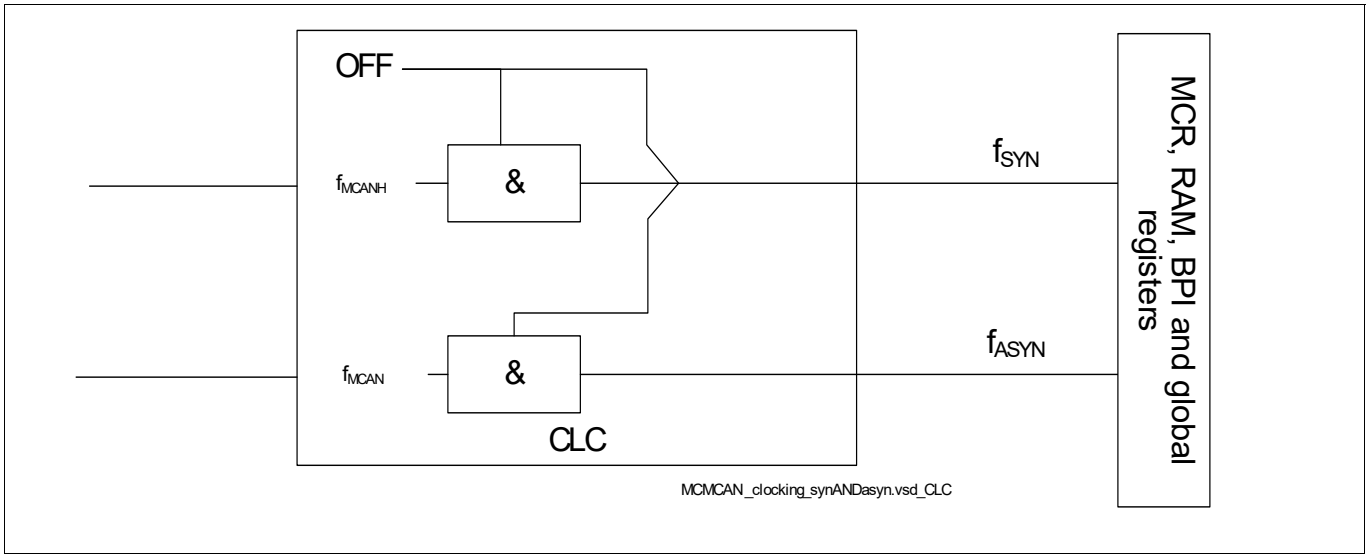


Figure 581 MCMCAN Clock Switch via CLC

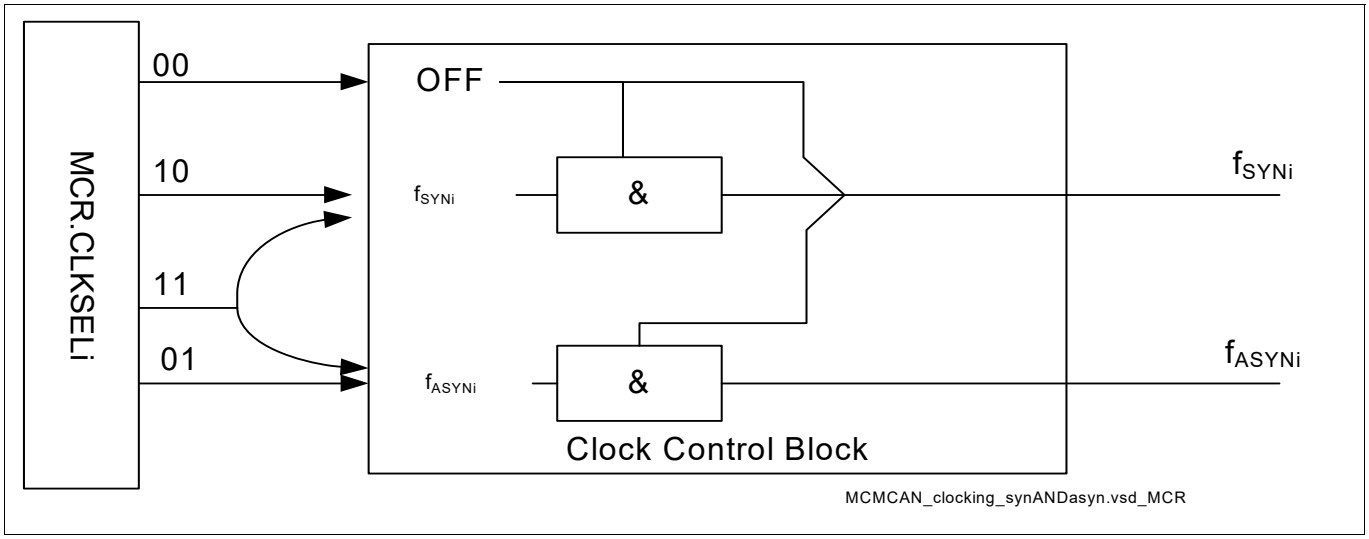


Figure 582 MCMCAN Clock Switch for single nodes

$f_{SYN}$is supplied from f_{MCANH} and f_{ASYN} is supplied from f_{MCAN} from CCU. f_{SYN} is used as the clock source for Register and RAM interface,f_{ASYN} is used to generate the nominal and fast CAN FD baudrates. It is recommended to use f_{ASYN} as 80, 40, 20 MHz from the Peripheral clock or also from f_{OSC}, in order to achieve commonly used nominal and fast CAN FD baudrates. The condition that f_{SYN} >= f_{ASYN} is essential for proper functioning of MCMCAN.

Note: *f_{SPB} is used as the clock source for SSH of MCMCAN RAM . Since a synchronization between f_{SPB} and f_{MCANH} happens within SSH, during normal operation of MCMCAN, f_{SPB} should be configured to frequency equal to f_{MCANH}. During Pretended Networking where f_{SPB} is on reduced frequency than f_{MCANH}, the MCMCAN SSH registers should be accessed only after leaving the Pretended Networking Mode. Refer to MTU chapter for additional details.*

Table 354 MCMCANClock Interconnects

CAN Clock Inputs	Connected to	Description
f_{MCAN}	CCU	f_{MCAN} of the MCMCAN module is one of the clock inputs of the Clock Control Block, providing the MCMCAN with the clock for the asynchronous clock path f_{ASYN}.
f_{MCANH}	CCU	f_{MCANH} of the MCMCAN module is the clock input of the Clock Control Register Block, providing the main kernel clock f_{SYN}.
f_{SYN}	CLC	f_{MCANH} becomes f_{SYN} within the module
f_{ASYN}	CLC	f_{MCAN} becomes f_{ASYN} within the module
f_{SYNi}	MCR.CLKSELi	f_{SYN} becomes f_{SYNi} clocking the synchronous part of M_CAN node i (Nodes can be switched on/off individually)
f_{ASYNi}	MCR.CLKSELi	f_{ASYN} becomes f_{ASYNi} clocking the asynchronous part of M_CAN node i (Nodes can be switched on/off individually)

40.3.1.1.1 Module Clock Generation

This chapter describes the clock generation and clock destinations.

Clock Selection

The asynchronous clock part of the M_CAN and the rest of the MCMCAN module are separate frequency domains and can be driven by separate independent frequencies. The clocks for the module are chosen within the clock control unit. The asynchronous clock can be chosen in the CCU among the Peripheral PLL or with direct drive from the oscillator.

The purpose of supplying the asynchronous clock part with a direct oscillator clock is to avoid the clock jitter added by the PLL, necessary when the chip is driven by a low cost ceramic resonator instead of by a high precision quartz crystal.

As shown in **Figure 581**, the clock signals for the MCMCAN module are generated and controlled by a clock control unit. This clock control unit is responsible for the enable/disable control, the clock frequency adjustment.

Clock control register

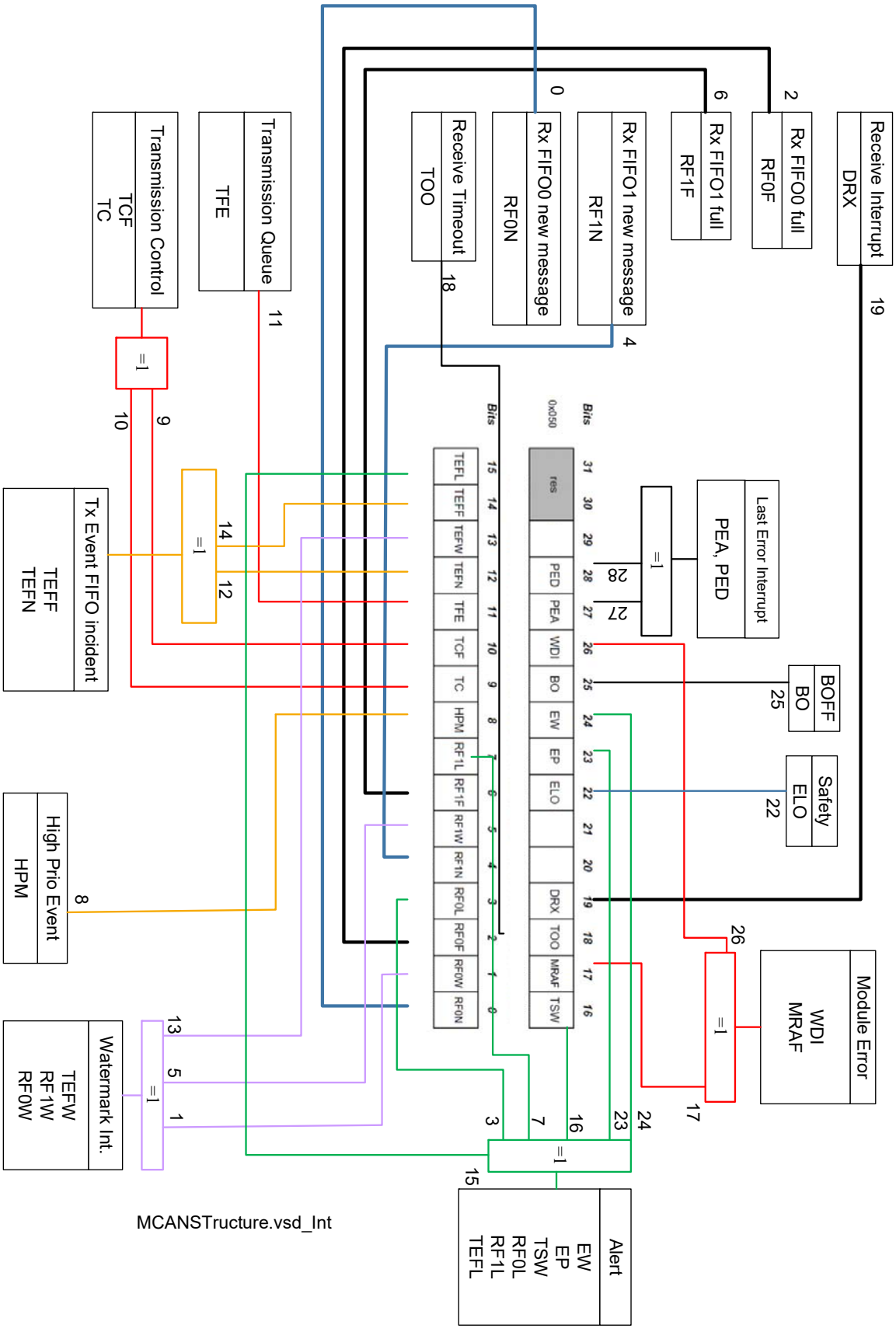
The global registers do include the **CLC** register: Module clock enabled. The module control clock f_{SYN} is used inside the MCMCAN module for control purposes such as clocking of control logic and register operations. The frequency of f_{SYN} is sourced by f_{MCANH} from CCU module. This clock is independent to f_{SPB} and allows M_CAN to continue operation when f_{SPB} is reduced in frequency, thus enabling pretended networking. The clock control register CLC makes it possible to enable/disable f_{SYN} and f_{ASYN} under certain conditions.

40.3.1.2 Interrupt groups

Interrupt grouping is fixed, as shown in figure **Figure 583** and **Figure 584**. For the complete module, 16 interrupt nodes are existing. The interrupt groups can be freely assigned to the nodes by using **GRINT1i (i=0-3)** and **GRINT2i (i=0-3)**.

40.3.1.2.1 Mapping of interrupts

The interrupt assignment from the interrupt register, only forwarding the enabled interrupt sources is as following:



MCANSTructure.vsd_Int

Figure 583 Mapping of interrupts into groups (without TTCAN)

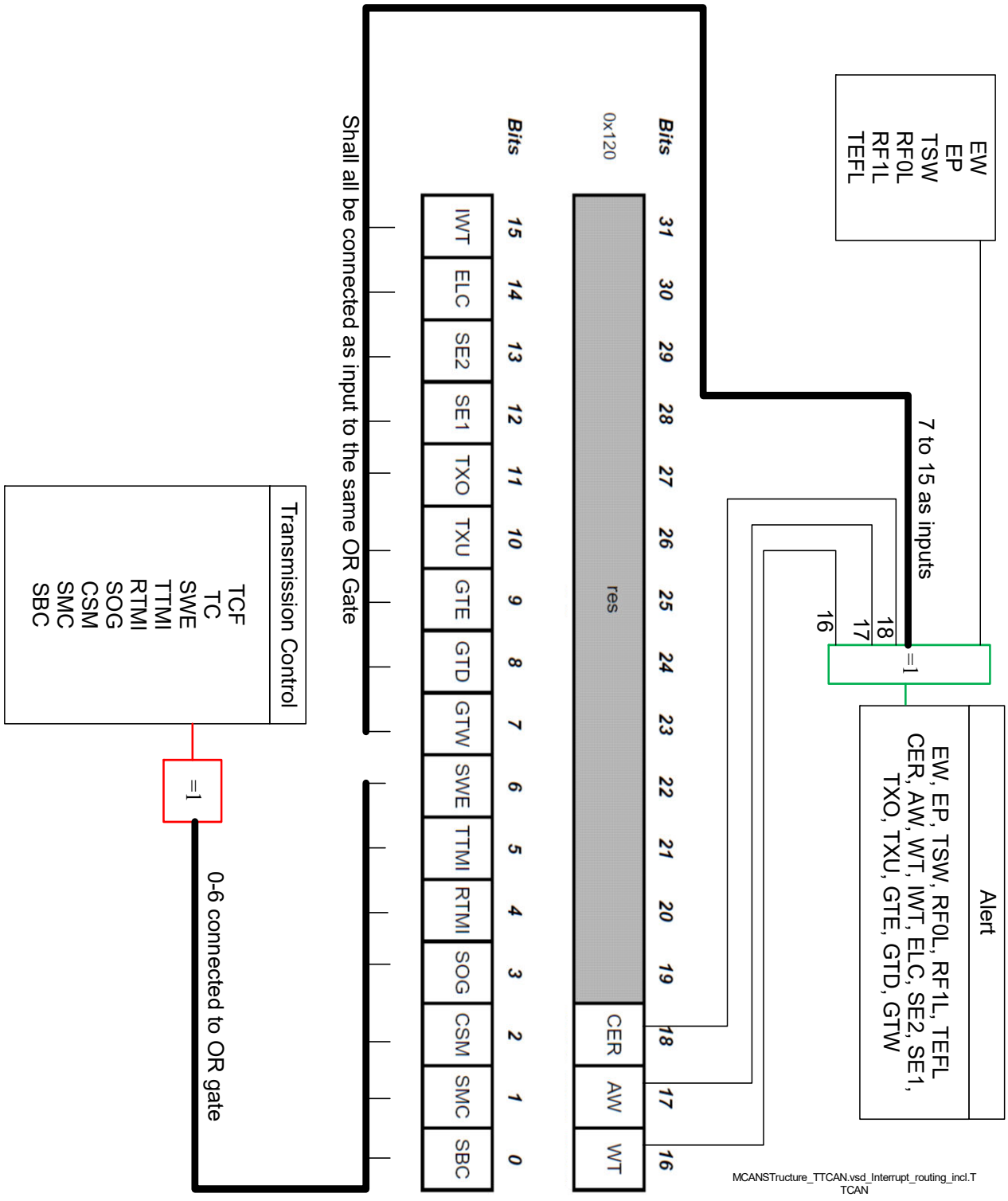


Figure 584 Mapping of TTCAN interrupts into groups

40.3.1.2.2 Signalling interrupts of groups

The groups defined in the previous paragraph are also shown in an **Interrupt Signalling Register i**. 0 means, that no interrupt is pending on the corresponding interrupt, 1 means pending.

40.3.1.2.3 Connections to Interrupt Router Inputs

The interrupt output line INT_O0-15 of MCMCAN is connected to the Interrupt Router module, see **Table 355**.

Table 355 Interrupt Router Inputs

Interrupt Router Input	Connected to CAN Interrupt Output
SRC_CANzINT0	INT_O0 MCMCAN
SRC_CANzINT1	INT_O1 MCMCAN
SRC_CANzINT2	INT_O2 MCMCAN
SRC_CANzINT3	INT_O3 MCMCAN
SRC_CANzINT4	INT_O4 MCMCAN
SRC_CANzINT5	INT_O5 MCMCAN
SRC_CANzINT6	INT_O6 MCMCAN
SRC_CANzINT7	INT_O7 MCMCAN
SRC_CANzINT8	INT_O8 MCMCAN
SRC_CANzINT9	INT_O9 MCMCAN
SRC_CANzINT10	INT_O10 MCMCAN
SRC_CANzINT11	INT_O11 MCMCAN
SRC_CANzINT12	INT_O12 MCMCAN
SRC_CANzINT13	INT_O13 MCMCAN
SRC_CANzINT14	INT_O14 MCMCAN
SRC_CANzINT15	INT_O15 MCMCAN

Interrupt Control

The general interrupt structure is shown in **Figure 585**. The interrupt event can trigger the interrupt generation. The interrupt pulse is generated based on the interrupt flag in the interrupt (status) register (**IRi (i=0-3)** and **TTIR0**) and the interrupt enable bit in Interrupt Enable register (**IEi (i=0-3)**and **TTIE0**). It is purely based on AND logic between the interrupt flags and interrupt enable bit fields. The interrupt flag can be reset by software by writing a '1' to the CANn_IRi bit.

If enabled by the related interrupt enable bit in the corresponding interrupt enable register (**IEi (i=0-3)**, **NTRTRi (i=0-3)**.TEIE and **TTIE0**), an interrupt pulse can be generated at one of the 16 interrupt output lines INT_On of the MCMCAN module using **GRINT1i (i=0-3)** and **GRINT2i (i=0-3)**. If more than one interrupt source is connected to the same interrupt node (in **GRINT1i (i=0-3)** and **GRINT2i (i=0-3)**), the requests are combined to one common line.

The interrupt groups are only ORing the interrupts of the corresponding CAN nodes. The interrupt request has to be reset in the node.

Note: Enabling an interrupt in Interrupt Enable register when the corresponding flag is already set in Interrupt Register will also generate an interrupt pulse. Hence it is recommended to clear the interrupt flags before enabling the corresponding interrupts.

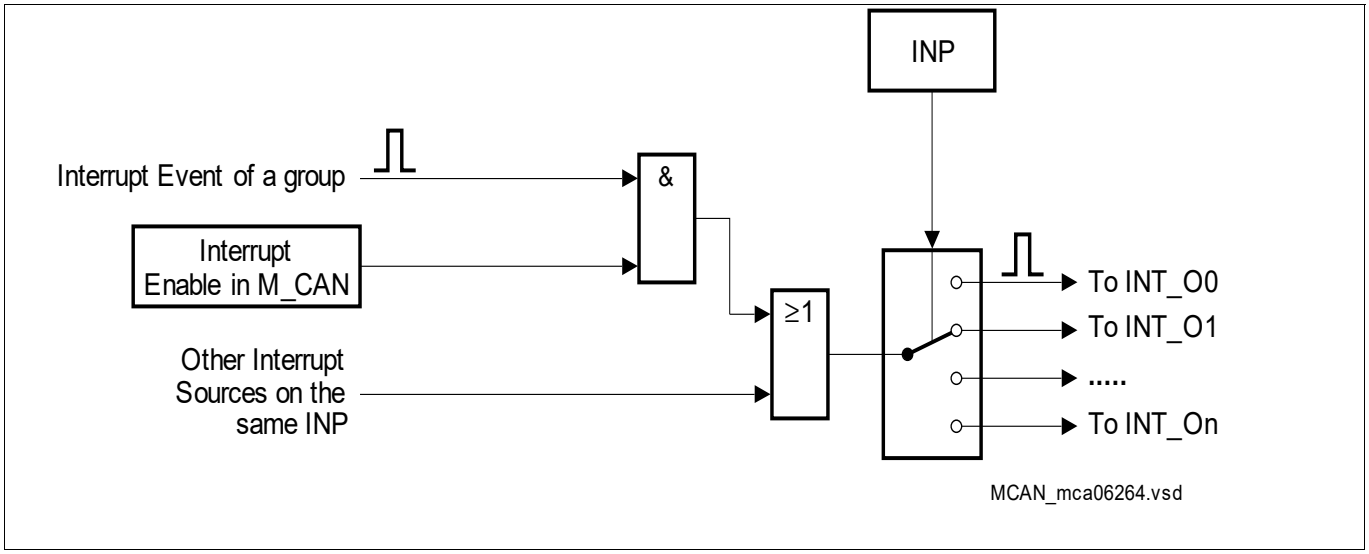


Figure 585 General Interrupt Structure

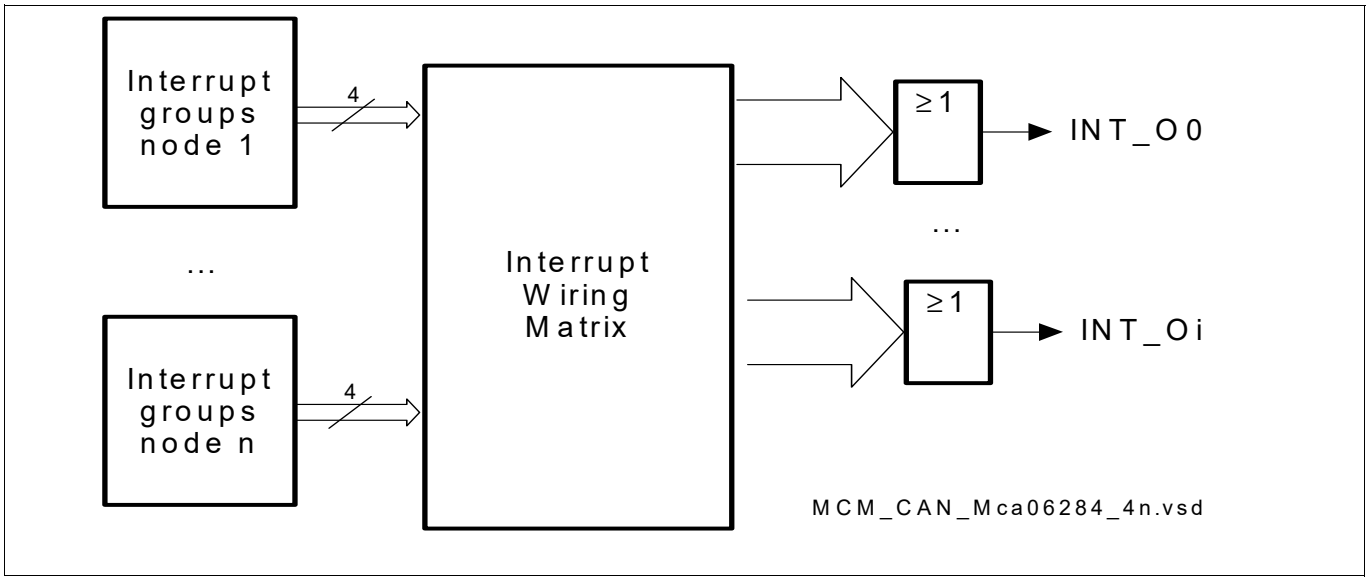


Figure 586 Interrupt compression Unit

40.3.1.3 Port, I/O Line Control and Interconnects

The interconnections between the MCMCAN module and the port I/O lines are controlled in the port logic. Additionally to the port input selection, the following port control operations must be executed:

- Input/output function selection (IOCR registers)
- Pad driver characteristics selection for the outputs (PDR registers)
- This chapter describes all connections, which are in relation

40.3.1.3.1 Input/Output Function Selection in Ports

The port input/output control registers contain the bit fields that select the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the MCMCAN module are controlled by the port input/output control registers, which are described in the port chapter.

Table 356 shows the corresponding pins, sorted by node. Even though the table is pair wise, it is possible to select a different pairing for RXD and TXD. In addition the RXSEL value to be programmed for the Receive Pins is part of this table. For more information on RXSEL please see **Node Receive Input Selection**.

Note: It is recommended to use "Fast Pads with Strong driver, medium/sharp edge" configuration when CAN-FD is used."

Table 356 MCMCAN I/O Control Selection and Setup

Node	RXD	Receive Port	NPCRx.RXSEL	Transmit Port
CAN00	CAN00_RXDA	P2.1	000_B	P2.0
	CAN00_RXDB	P20.7	001_B	P20.8
	CAN00_RXDC	P12.0	010_B	P12.1
	CAN00_RXDD	P33.12	011_B	P33.13
	CAN00_RXDE	P33.7	100_B	P33.8
	CAN00_RXDF	P1.8	101_B	P1.13
	CAN00_RXDG	P34.2	110_B	P34.1
	CAN00_RXDH	P2.14	111_B	P2.13
CAN01	CAN01_RXDA	P15.3	000_B	P15.2
	CAN01_RXDB	P14.1	001_B	P14.0
	CAN01_RXDC	P1.4	010_B	P1.3
	CAN01_RXDD	P33.10	011_B	P33.9
	CAN01_RXDE	P2.10	100_B	P2.9
	CAN01_RXDF	not used	101_B	
	CAN01_RXDG	not used	110_B	
	CAN01_RXDH	not used	111_B	
CAN02	CAN02_RXDA	P15.1	000_B	P15.0
	CAN02_RXDB	P2.3	001_B	P2.2
	CAN02_RXDC	P32.6	010_B	P32.5
	CAN02_RXDD	P14.8	011_B	P14.10
	CAN02_RXDE	P10.2	100_B	P10.3
	CAN02_RXDF	not used	101_B	
	CAN02_RXDG	not used	110_B	
	CAN02_RXDH	not used	111_B	
CAN03	CAN03_RXDA	P0.3	000_B	P0.2
	CAN03_RXDB	P32.2	001_B	P32.3
	CAN03_RXDC	P20.0	010_B	P20.3
	CAN03_RXDD	P11.10	011_B	P11.12
	CAN03_RXDE	P20.9	100_B	P20.10
	CAN03_RXDF	P1.0	101_B	P1.2
	CAN03_RXDG	not used	110_B	
	CAN03_RXDH	not used	111_B	

Table 356 MCMCAN I/O Control Selection and Setup (cont'd)

Node	RXD	Receive Port	NPCRx.RXSEL	Transmit Port
CAN10	CAN10_RXDA	P0.1	000_B	P0.0
	CAN10_RXDB	P14.7	001_B	P14.9
	CAN10_RXDC	P23.0	010_B	P23.1
	CAN10_RXDD	P13.1	011_B	P13.0
	CAN10_RXDE	not used	100_B	
	CAN10_RXDF	not used	101_B	
	CAN10_RXDG	not used	110_B	
	CAN10_RXDH	not used	111_B	
CAN11	CAN11_RXDA	P2.4	000_B	P2.5
	CAN11_RXDB	P0.5	001_B	P0.4
	CAN11_RXDC	P23.7	010_B	P23.6
	CAN11_RXDD	P11.7	011_B	P11.0
	CAN11_RXDE	not used	100_B	
	CAN11_RXDF	not used	101_B	
	CAN11_RXDG	not used	110_B	
	CAN11_RXDH	not used	111_B	
CAN12	CAN12_RXDA	P20.6	000_B	P20.7
	CAN12_RXDB	P10.8	001_B	P10.7
	CAN12_RXDC	P23.3	010_B	P23.2
	CAN12_RXDD	P11.8	011_B	P11.1
	CAN12_RXDE	not used	100_B	
	CAN12_RXDF	not used	101_B	
	CAN12_RXDG	not used	110_B	
	CAN12_RXDH	not used	111_B	
CAN13	CAN13_RXDA	P14.7	000_B	P14.6
	CAN13_RXDB	P33.5	001_B	P33.4
	CAN13_RXDC	P22.5	010_B	P22.4
	CAN13_RXDD	P11.13	011_B	P11.4
	CAN13_RXDE	not used	100_B	
	CAN13_RXDF	not used	101_B	
	CAN13_RXDG	not used	110_B	
	CAN13_RXDH	not used	111_B	
CAN20	CAN20_RXDA	P10.5	000_B	P10.6
	CAN20_RXDB	P10.8	001_B	P10.7
	CAN20_RXDC	P34.2	010_B	P34.1
	CAN20_RXDD	P2.14	011_B	P2.13
	CAN20_RXDE	P1.8	100_B	P1.13
	CAN20_RXDF	P11.14	101_B	P11.5

Table 356 MCMCAN I/O Control Selection and Setup (cont'd)

Node	RXD	Receive Port	NPCRx.RXSEL	Transmit Port
	CAN20_RXDG	not used	110_B	
	CAN20_RXDH	not used	111_B	
CAN21	CAN21_RXDA	P0.3	000_B	P0.2
	CAN21_RXDB	P13.12	001_B	P13.9
	CAN21_RXDC	P20.0	010_B	P20.3
	CAN21_RXDD	P32.2	011_B	P32.3
	CAN21_RXDE	P1.0	100_B	P1.2
	CAN21_RXDF	P22.7	101_B	P22.6
	CAN21_RXDG	not used	110_B	
	CAN21_RXDH	not used	111_B	
CAN22	CAN22_RXDA	P33.13	000_B	P33.12
	CAN22_RXDB	P32.7	001_B	P32.6
	CAN22_RXDC	P23.6	010_B	P23.5
	CAN22_RXDD	P14.14	011_B	P14.13
	CAN22_RXDE	P22.9	100_B	P22.8
	CAN22_RXDF	not used	101_B	
	CAN22_RXDG	not used	110_B	
	CAN22_RXDH	not used	111_B	
CAN23	CAN23_RXDA	P14.10	000_B	P14.9
	CAN23_RXDB	P23.3	001_B	P23.2
	CAN23_RXDC	P14.15	010_B	P14.14
	CAN23_RXDD	P13.5	011_B	P13.4
	CAN23_RXDE	P22.11	100_B	P22.10
	CAN23_RXDF	not used	101_B	
	CAN23_RXDG	not used	110_B	
	CAN23_RXDH	not used	111_B	

Node Receive Input Selection

Additionally to the I/O control selection, as defined in the port chapter, the selection of a CAN node's receive input line requires that bit field RXSEL in its node port control register **NPCRi (i=0-3)** must be set according to **Table 356**. Values for **NPCRi (i=0-3)**.RXSEL other than those of the table mentioned above will result in a recessive receive input for node x. As a hint for the RXSEL values, a receive pin ending with A results in 0x0, B in 0x1 until H resulting in the value of 0x7.

40.3.1.3.2 Trigger inputs

The following paragraph describes all interconnections, which are relevant for triggering events.

40.3.1.3.3 External CAN Time Trigger Inputs

The external CAN time trigger inputs ECTT[8:1] of MCMCANcan be used as a transmit trigger for a reference message. In the AURIX™ TC3xx Platform, these input lines are connected as shown in **Table 357**.

Table 357 External CAN Time Trigger Inputs

Multiplexer Input	Connected to	From / to Module
ECTT1	P2.4 / ECTT1	Port 2
ECTT2	P2.5 / ECTT2	Port 2
ECTT3	Output IOUT2	External Request Unit (SCU)
ECTT4	Output IOUT3	External Request Unit (SCU)
ECTT5	eray.eray_tint0_o	Timer Service Request 0 (E-Ray)
ECTT6	eray.eray_tint1_o	Timer Service Request 1 (E-Ray)
ECTT7	gtm_0.sx_gtm2mcan_mcan 0.0	GTM TTCAN Trigger 0
ECTT8	gtm_0.sx_gtm2mcan_mcan 0.1	GTM TTCAN Trigger 1

40.3.1.3.4 CAN Transmit Trigger Inputs

The CAN transmit trigger inputs of MCMCAN are used to trigger for transmission of a message. In the AURIX™ TC3xx Platform, these input lines are connected as shown in **Table 358**. **Table 358** is CAN0 only, for CAN1 and following the mcan0 has to be exchanged to mcan1 and following.

TC39xA only: The cross connections from GTM are identical for CAN1 and CAN2.

Table 358 CAN Transmit Trigger Inputs

Receive Input	Connected to	From / to Module
CAN Node 0		
STM_N0_TRIG	STM0.SR0_INT	System Timer Module (STM)
GTM_N0_TRIG	gtm_0.sx_gtm2mcan_mcan0.0	General Timer Module (GTM)
…		
…	…	…
CAN Node 3		

Table 358 CAN Transmit Trigger Inputs (cont'd)

Receive Input	Connected to	From / to Module
STM_N3_TRIG	STM0.SR0_INT	System Timer Module (STM)
GTM_N3_TRIG	gtm_0.sx_gtm2mcan_mcan0.3	General Timer Module (GTM)

40.3.1.3.5 TT Capture Time Trigger Input

The CAN local time register capture input of MCMCAN is used to trigger the capture of the TT Cycle Time & Count (TTCTC) to the TT Capture Time (TTCPT). In the AURIX™ TC3xx Platform, the input lines are connected as shown in **Table 359** and can be selected using **TTCR0**.TTCTSS.

Table 359 TT Capture Time Register Trigger Input

Receive Input	Connected to	From / to Module
TTCPT_TRIG1	STM0.SR0_INT	System Timer Module (STM)
TTCPT_TRIG2	STM1.SR0_INT	System Timer Module (STM)
TTCPT_TRIG3	STM2.SR0_INT	System Timer Module (STM)
TTCPT_TRIG4	Output IOUT4	External Request Unit (SCU)

Transmit Trigger for a Reference Message

The transmission of a reference message can be triggered by a time mark or by an external event. If it is triggered by a time mark, the "Next is Gap" bit (TTOCN0.NIG for receiving **TTOST0**.WFE) of the previous reference message has been 0. If it is triggered by an external event, the "Next is Gap" bit of the previous reference message has been 1.

The software can set bit **TTOCN0**.NIG in order to initiate the synchronization of a reference message to an event. This event can be an edge at an external input or a software action (write bit **TTOCN0**.FGP = 1, software trigger event).

The transmit trigger generation logic for the reference message is shown in **Figure 587**. The edge detection for ECTTx contains a synchronization stage.

The trigger event can be selected by bit field **TTCR0**.ETESEL (external trigger event selection). Only trigger events are taken into account for the transfer of a reference message that have been detected after the start of the current basic cycle.

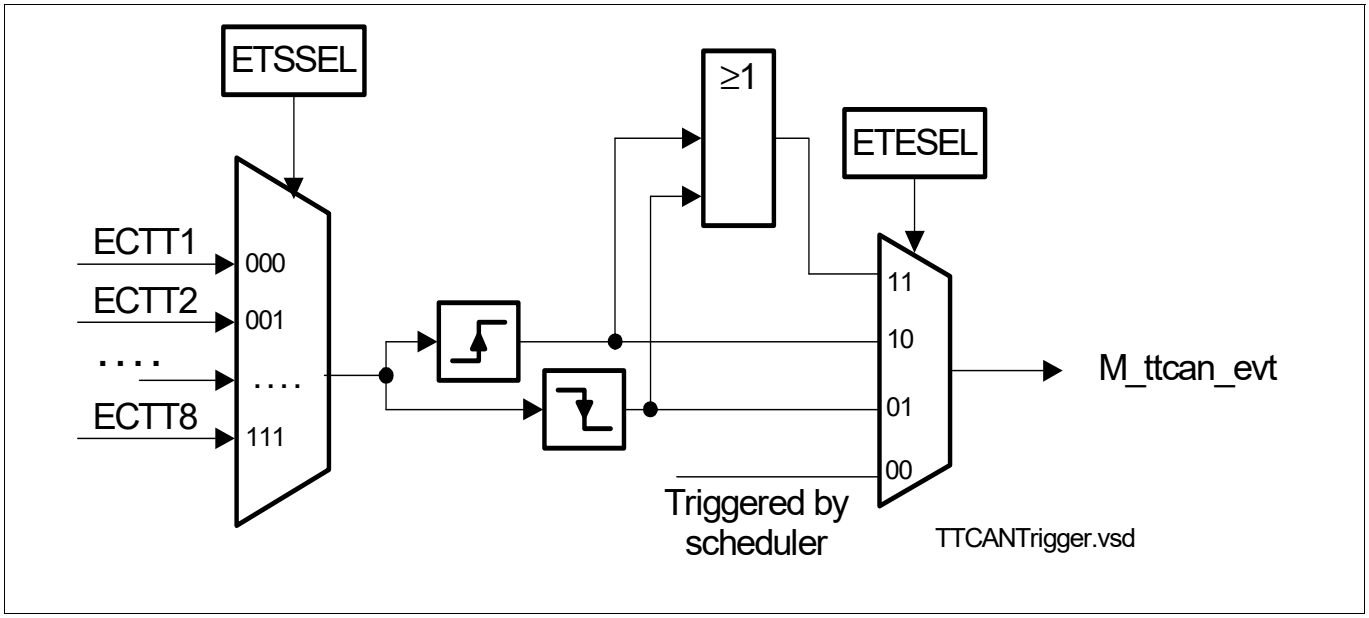


Figure 587 External Trigger Generation and Transmit Trigger for the Reference Message

40.3.1.3.6 Signals to other modules

GTM: Interrupt 12 to 15 are connected to GTM inputs.

Connections to General Timer Module (GTM) Inputs

The interrupt output line INT_O12-15 of MCMCAN is connected to the General Timer Module, see **Table 360**.

Table 360 General Timer Module Inputs

GTM Input	Connected to CAN Interrupt Output
Timer input (gtm.tim_0_muxin_1_i(13))	INT_O12 (mcan0.int_req_o(12))
Timer input (gtm.tim_1_muxin_1_i(13))	
Timer input (gtm.tim_2_muxin_1_i(13))	
Timer input (gtm.tim_0_muxin_2_i(13))	INT_O13 (mcan0.int_req_o(13))
Timer input (gtm.tim_1_muxin_2_i(13))	
Timer input (gtm.tim_2_muxin_2_i(13))	
Timer input (gtm.tim_0_muxin_3_i(13))	INT_O14 (mcan0.int_req_o(14))
Timer input (gtm.tim_1_muxin_3_i(13))	
Timer input (gtm.tim_2_muxin_3_i(13))	
Timer input (gtm.tim_0_muxin_4_i(13))	INT_O15 (mcan0.int_req_o(15))
Timer input (gtm.tim_1_muxin_4_i(13))	
Timer input (gtm.tim_2_muxin_4_i(13))	

Table 361 TIM0 Mapping

CH1SEL	Pad / Input	Name	Packages/[Products]
1101_B	can0_int[12]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH2SEL	**Pad / Input**	**Name**	Packages/[Products]

Table 361 TIM0 Mapping (cont'd)

1101_B	can0_int[13]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH3SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[14]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH4SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[15]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH5SEL	**Pad / Input**	**Name**	Packages/[Products]

Table 362 TIM1 Mapping

CH1SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[12]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH2SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[13]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH3SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[14]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]
CH4SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[15]	CAN	QFP144,176,BGA292,516/[TC39x,38x,37x,36x,33x]

Table 363 TIM2 Mapping

CH1SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[12]	CAN	QFP144,176,BGA292,516/ [TC39x,38x,37x,36x]
CH2SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[13]	CAN	QFP144,176,BGA292,516/ [TC39x,38x,37x,36x]
CH3SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[14]	CAN	QFP144,176,BGA292,516/ [TC39x,38x,37x,36x]
CH4SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[15]	CAN	QFP144,176,BGA292,516/ [TC39x,38x,37x,36x]

Table 364 TIM3 Mapping

CH1SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[12]	CAN	QFP176,BGA292,BGA516/[TC39x/38x/37x]
CH2SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[13]	CAN	QFP176,BGA292,BGA516/[TC39x/38x/37x]
CH3SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[14]	CAN	QFP176,BGA292,BGA516/[TC39x/38x/37x]
CH4SEL	**Pad / Input**	**Name**	Packages/[Products]
1101_B	can0_int[15]	CAN	

40.3.1.4 Connecting the module to the outside world

The MCMCAN module can be connected per node either to a port configuring the **NPCRi (i=0-3)**.RXSEL for the input, or can be connected to an module internal CAN bus.

40.3.1.4.1 Module internal Loop-Back Mode

The MCMCAN module provides a Module internal Loop-Back Mode to enable an in-system test of the MCMCAN module as well as the development of CAN driver software without access to an external CAN bus.

The loop-back feature consists of an internal CAN bus (inside the MCMCAN module) and a bus select switch for each CAN node. With the switch, each CAN node can be connected either to the internal CAN bus (Internal Loop-Back Mode activated) or the external CAN bus, respectively to transmit and receive pins (normal operation). The CAN bus that is not currently selected is driven recessive; this means the transmit pin is held at 1, and the receive pin is ignored by the CAN nodes that are in Loop-Back Mode.

The Internal Loop-Back Mode is selected for CAN node x by setting the Node x Port Control Register bit **NPCRi (i=0-3)**.LBM. All CAN nodes that are in Loop-Back Mode may communicate together via the internal CAN bus without affecting the normal operation of the other CAN nodes that are not in Loop-Back Mode.

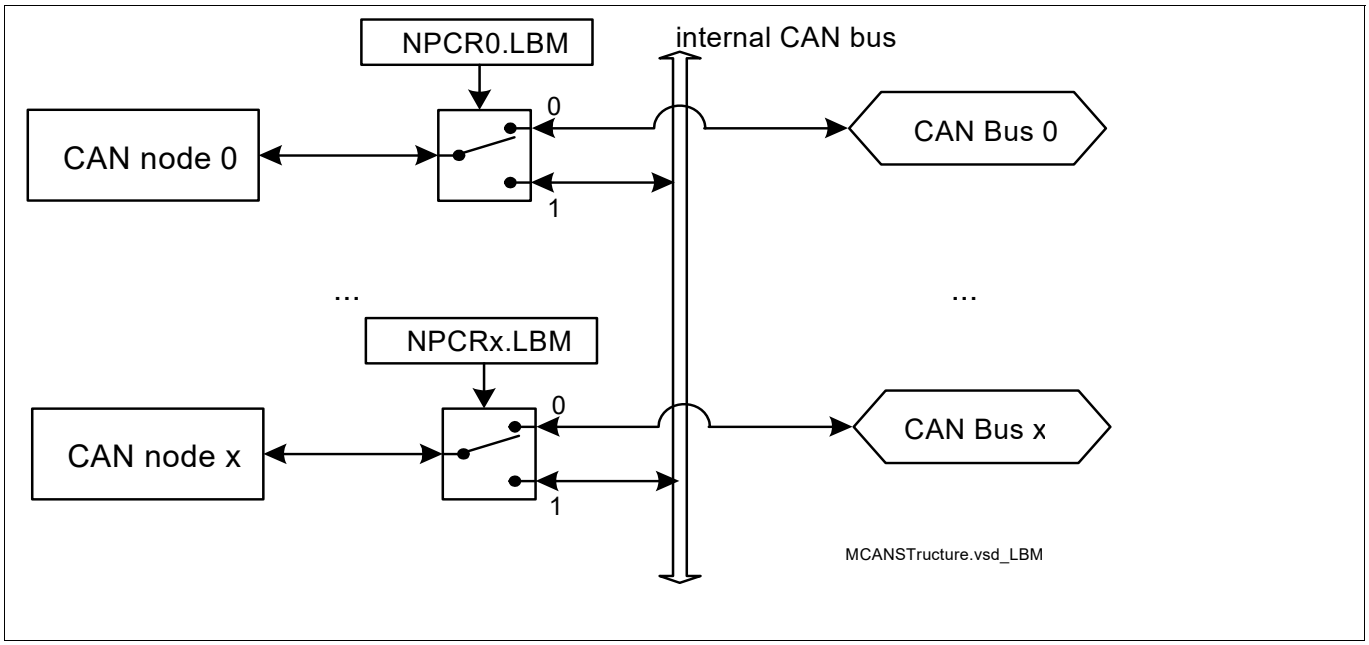


Figure 588 Module Internal Loop-Back Mode

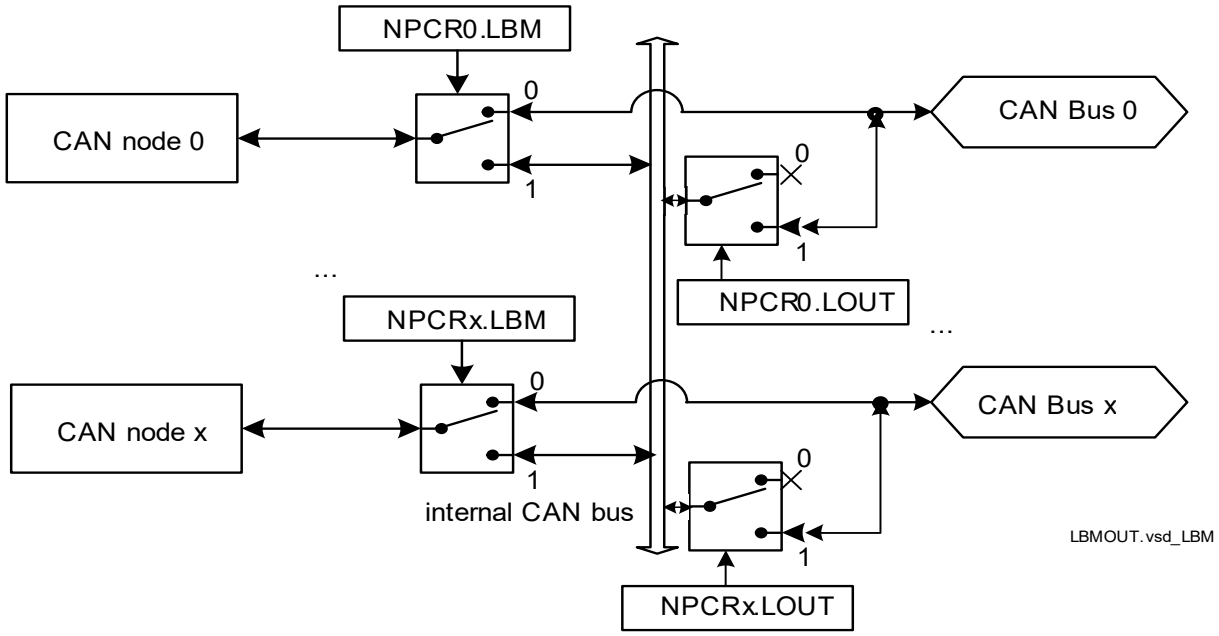


Figure 589 Module Loop-Back Mode Out

40.3.1.4.2 Module Loop Back Mode Out (B-step feature)

Setting **NPCRi (i=0-3)**.LOUT bit will send the signals from the internal loop back bus to the corresponding pins of the CAN node and vice versa. Therefore a receive and transmit between internal loop back bus and a bus system outside is possible. When **NPCRi (i=0-3)**.LOUT is set to one, the internal loop back bus and the external bus will be communicating with each other. The table below explains the behaviour of the CAN module for different combinations **NPCRi (i=0-3)**.LOUT and **NPCRi (i=0-3)**.LBM.

Table 365 Combinations of Internal Loop Back Mode and Loop Back Mode Out

NPCR.LOUT	NPCR.LBM	Description
0	0	Normal CAN operation. Internal and loop back out mode deactivated
0	1	Internal Loop Back mode
1	0	Loop Back Mode Out
1	1	Reserved

40.3.1.5 Fixing the address protection for the CAN nodes

Inside the BPI part, the address protection is defined. Here the nodes become protectable, as well as the control area. With the nodes a certain RAM area is assigned, therefore a start address and an end address per node is necessary. An overlap of the protected areas, is not found by the hardware. **STARTADRi (i=0-3)** and **ENDADRi (i=0-3)** registers, give the possibility to define a range within CAN RAM, belonging to a node, which can only be written by certain masters. The mechanism is identical to ACCEN registers. The RAM will only be write and not read protected.

40.3.1.6 Node Timing Functions

A CAN node offers the following timing functions:

- A receive time-out mode that can detect the reception within message buffers. The timeout expires, when no message has been received.
- Without CPU involvement, a selectable message buffers can be transmitted periodically, triggered by a timer, a System Timer (STM) or General Timer Module (GTM).

The clocking options for the node timer is controlled by Node Timer Clock Control Register, **Node i Timer Clock Control Register**, the node timing functions for Receive Timeout Mode is controlled by Node x Timer Receive Timeout Register, **Node i Timer Receive Timeout Register** and for Transmit Trigger Mode is controlled by Node x Timer A/B/C Transmit Trigger Registers, **Node i Timer A Transmit Trigger Register**, **Node i Timer B Transmit Trigger Register**, **Node i Timer C Transmit Trigger Register**. Links are only for CAN0, but the feature is available for all CAN modules. In general, the timers will not run, without setting the START bit and not without setting the RELOAD value, to a different value than 0. Whatever comes last, will start the timer.

Modes with Timer Usage

A CAN node timer is driven by the synchronous clock or by the corresponding clock sources, divided by a prescaler selected via bit field TPSC in the corresponding timer control register. The timers are enabled by writing to the RELOAD bits in the relevant node timer registers; then it decrements from its initial value. The further behavior depends on the selected timer mode:

- **Receive Timeout Mode:**The receive timeout function is valid for the receive buffers. A receive time-out check is enabled, which may be a received frame or remote data frame. If any of the message buffers receives before the timer is 0, the timer will be reloaded. When the timer reaches 0, it will stop. Bit TE in **NTRTRi (i=0-3)** register will be set. With bit **NTRTRi (i=0-3)**.TEIE = 1, an interrupt will be generated.
- **Transmit Trigger Mode:**When the timer reaches 0, the TXRQ of the corresponding message buffer is selected.

Transmit Trigger by System Timer or General Timer Module

For Node i Timer the trigger for transmission of a message can also be set by a System Timer (STM) trigger event or General Timer Module (GTM) trigger event. See **Figure 590**.
Bit **NTCCRi (i=0-3)**.TRIGSRC in the timer clock control register enables this feature and the timer is started once values are written to the RELOAD bits of the relevant **NTATTRi (i=0-3)**, **NTBTTRi (i=0-3)**, or **NTCTTRi (i=0-3)** registers. In transmit trigger mode, when a trigger event occurs (STM or GTM), the node timer will be decremented per trigger event timing prescaled by (TPSC+1) till it reaches zero where the transmit request is set for the corresponding transmit message buffer.

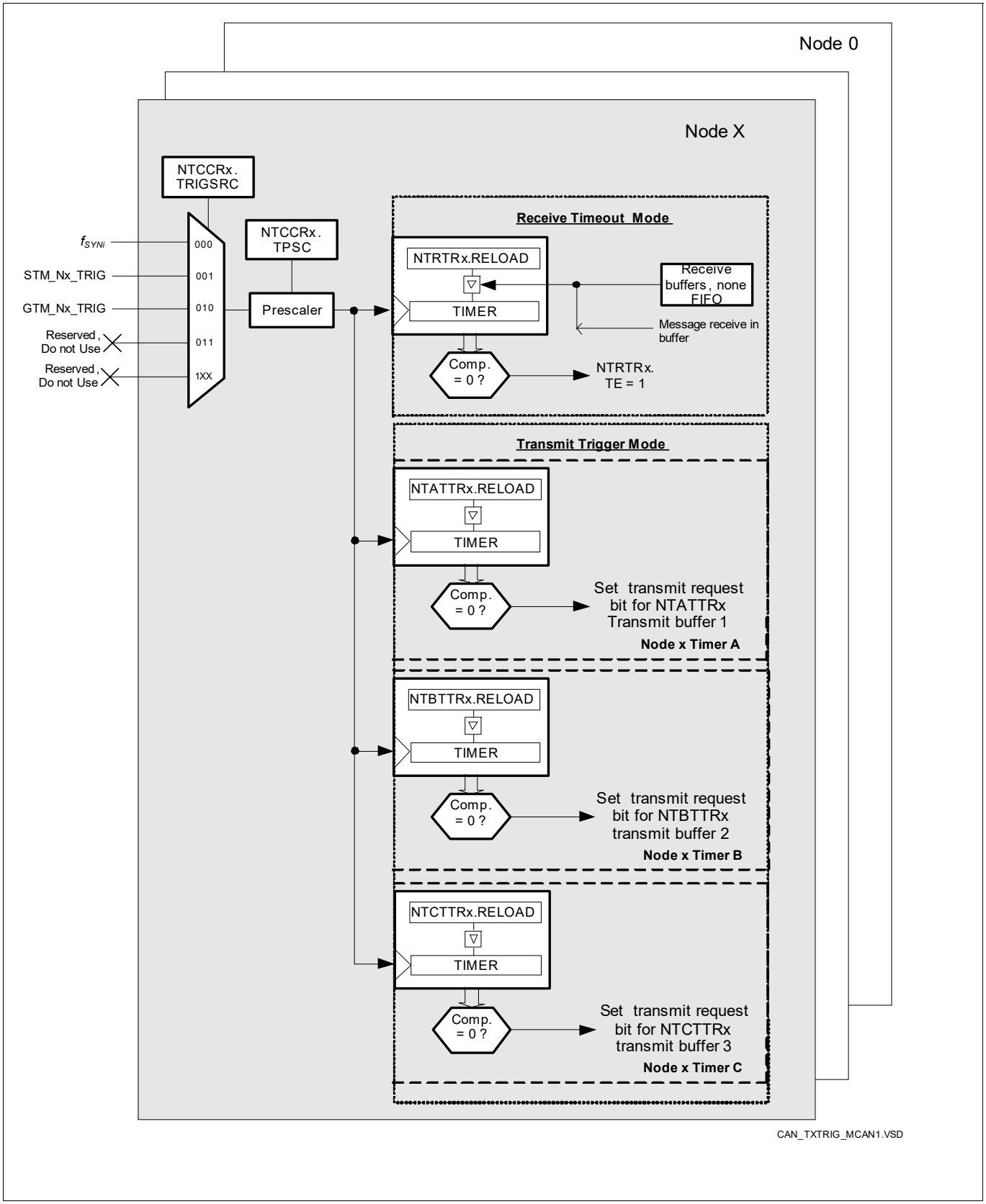


Figure 590 CAN Node Timing Modes

40.3.1.6.1 Automatic transferring of messages

NTATTRi (i=0-3), **NTBTTRi (i=0-3)**, **NTCTTRi (i=0-3)** registers enable the M_CAN IP to trigger messages on timer events. The timer per module have one clock source. The counting events can take place on an STM interrupt, a GTM interrupt or by f_{SYNi}. These timers can be used to enable Pretended Networking, or to have hardware support for a gateway functionality. This functionality needs a receive interrupt or a watermark interrupt within a FIFO, triggering a DMA transfer. The transmit objects will be automatically triggered to transfer the messages below, via timer underflow. As the transmit triggers are fixed to message RAM buffers 1-3, this feature only works if these buffers are available.

40.3.1.7 Destructive Debug Entry

On destructive debug entry ie., when DMU_SP_PROCONHSMCFG.DESTDBG = 11b and DMU_HF_PROCONDBG.EDM = 11b, the CAN transmit data output (TXD) is tied to “Recessive state” (logic ‘1b’). This blocks any further CAN transmission when AURIX™ TC3xx Platform has entered the Destructive Debug state.

40.3.2 M_CAN Functional Description

40.3.2.1 Operating Modes

40.3.2.1.1 Software Initialization

Software initialization is started by setting bit CCCRi.INIT, either by software or by a hardware reset, or by going Bus_Off. While **CCCRi (i=0-3)**.INIT is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output TXD is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting **CCCRi (i=0-3)**.INIT does not change any configuration register. Resetting **CCCRi (i=0-3)**.INIT finishes the software initialization. Afterwards the Bit Stream Processor BSP synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the M_CAN configuration registers is only enabled when both bits **CCCRi (i=0-3)**.INIT and **CCCRi (i=0-3)**.CCE are set (protected write).

CCCRi (i=0-3).CCE can only be set/reset while **CCCRi (i=0-3)**.INIT = '1'. **CCCRi (i=0-3)**.CCE is automatically reset, when CCCRi.INIT is cleared.

The following registers are reset when **CCCRi (i=0-3)**.CCE is set

- **HPMSi (i=0-3)** - High Priority Message Status
- **RXF0Si (i=0-3)** - Rx FIFO 0 Status
- **RXF1Si (i=0-3)** - Rx FIFO 1 Status
- **TXFQSi (i=0-3)** - Tx FIFO/Queue Status
- **TXBRPi (i=0-3)** - Tx Buffer Request Pending
- **TXBTOi (i=0-3)** - Tx Buffer Transmission Occurred
- **TXBCFi (i=0-3)** - Tx Buffer Cancellation Finished
- **TXEFSi (i=0-3)** - Tx Event FIFO Status
- **TTOST0** - TT Operation Status
- **TTLGT0** - TT Local & Global Time, only Global Time TTLGT0.GT is reset
- **TTCTC0** - TT Cycle Time & Count
- **TTCSM0** - TT Cycle Sync Mark

The Timeout Counter value **TOCVi (i=0-3)**.TOC is preset to the value configured by **TOCCi (i=0-3)**.TOP when **CCCRi (i=0-3)**.CCE is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while **CCCRi (i=0-3)**.CCE = '1'.

The following registers are only writeable while **CCCRi (i=0-3)**.CCE = '0'

- **TXBARi (i=0-3)** - Tx Buffer Add Request
- **TXBCRi (i=0-3)** - Tx Buffer Cancellation Request

CCCRi (i=0-3).TEST and **CCCRi (i=0-3)**.MON can only be set by the Host while **CCCRi (i=0-3)**.INIT = '1' and **CCCRi (i=0-3)**.CCE = '1'. Both bits may be reset at any time. **CCCRi (i=0-3)**.DAR can only be set/reset while **CCCRi (i=0-3)**.INIT = '1' and **CCCRi (i=0-3)**.CCE = '1'.

40.3.2.1.2 Normal Operation

The M_CAN's default operating mode after hardware reset is event-driven CAN communication without time triggers (TTOCF0.OM = "00"). It is required that both **CCCRi (i=0-3)**.INIT and **CCCRi (i=0-3)**.CCE are set before the TT Operation Mode can be changed.

Once the M_CAN is initialized and **CCCRi (i=0-3)**.INIT is reset to zero, the M_CAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx Queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

40.3.2.1.3 CAN FD Operation

There are two variants in the CAN FD frame transmission, first the CAN FD frame without bit rate switching. The second variant is the CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame. The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit. FDF = recessive signifies a CAN FD frame, FDF = dominant signifies a Classical CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol. In case the M_CAN receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit **PSRi (i=0-3)**.PXE. When Protocol Exception Handling is enabled (**CCCRi (i=0-3)**.PXHD = '0'), this causes the operation state to change from Receiver (**PSRi (i=0-3)**.ACT = "10") to Integrating (**PSRi (i=0-3)**.ACT = "00") at the next sample point. In case Protocol Exception Handling is disabled (**CCCRi (i=0-3)**.PXHD = '1'), the M_CAN will treat a recessive res bit as an form error and will respond with an error frame.CAN FD operation is enabled by programming **CCCRi (i=0-3)**.FDOE. In case **CCCRi (i=0-3)**.FDOE = '1', transmission and reception of CAN FD frames is enabled. Transmission and reception of Classical CAN frames is always possible. Whether a CAN FD frame or a Classical CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With **CCCRi (i=0-3)**.FDOE = '0', received frames are interpreted as Classical CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. CCCRi.FDOE and CCCRi.BRSE can only be changed while CCCRi.INIT and CCCRi.CCE are both set.With CCCRi.FDOE = '0', the setting of bits FDF and BRS is ignored and frames are transmitted in Classical CAN format. With **CCCRi (i=0-3)**.FDOE = '1' and **CCCRi (i=0-3)**.BRSE = '0', only bit FDF of a Tx Buffer element is evaluated. With **CCCRi (i=0-3)**.FDOE = '1' and **CCCRi (i=0-3)**.BRSE = '1', transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classical CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN Partial Networking have to be transmitted in Classical CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classical CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, which in standard CAN all code a data field of 8 bytes, are coded according to **Table 366** below.

Table 366 Coding of DLC in CAN FD

DLC	9	10	11	12	13	14	15
Number of Data Bytes	12	16	20	24	32	48	64

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the standard CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register NBTPi. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing & Prescaler Register **DBTPi (i=0-3)**. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN clock frequency (asynchronous clock). Example: with a CAN clock frequency of 20 MHz and the shortest configurable bit time of 4 tq, the bit rate in the data phase is 5 Mbit/s. (As 4 tq is quite tight, please check if this works in your environment.)

Note: MCMCAN supports upto 5 Mbit/s considering the physical medium CAN-FD timing requirements from ISO 11898-2:2016. At f_{MCAN} = 80MHz, CAN-FD data phase bit rate can be extended to 8 Mbit/s, while the bit asymmetry effect from CAN-FD transceiver, physical layer network topology etc., has to be considered by the user.

In both data frame formats, CAN FD long and CAN FD fast, the value of the bit ESI (Error Status Indicator) is determined by the transmitter's error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

40.3.2.1.4 Transmitter Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin TXD the protocol controller receives the transmitted data from its local CAN transceiver via pin RXD. The received data is delayed by the CAN transmitter delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transmitter delay, the delay compensation is introduced. Without transmitter delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transmitter delay.

Description

The CAN FD protocol unit has implemented a delay compensation mechanism to compensate the CAN transceiver's loop delay, the transmitter delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver and port delay.

For the transmitter delay compensation the following boundary conditions have to be considered:

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point. If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit **DBTPi (i=0-3)**.TDC.The received bit is compared against the transmitted bit at the Secondary Sample Point. The Secondary Sample Point position is defined as the sum of the measured delay from the M_CAN's transmit output TX through the transceiver to the receive input RX plus the transmitter delay compensation offset as configured by **TDCRi (i=0-3)**.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq.PSRi.TDCV shows the actual transmitter delay compensation value. **PSRi (i=0-3)**.TDCV is cleared when CCCRi.INIT is set and is updated at each transmission of an FD frame while **DBTPi (i=0-3)**.TDC is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the M_CAN:

- The sum of the measured delay from TX to RX and the configured transmitter delay compensation offset **TDCRi (i=0-3)**.TDCO has to be less than 6 bit times in the data phase.The sum of the measured delay from RX to RX and the configured transmitter delay compensation offset TDCRi.TDCO has to be less or equal 127 mtq. In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used for transmitter delay compensation.The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the Secondary Sample Points.

Transmitter Delay Compensation Measurement

If transmitter delay compensation is enabled by programming **DBTPi (i=0-3)**.TDC = '1', the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input RX of the transmitter. The resolution of this measurement is one mtq.

Figure 591 below describes how the transmitter loop delay is measured.

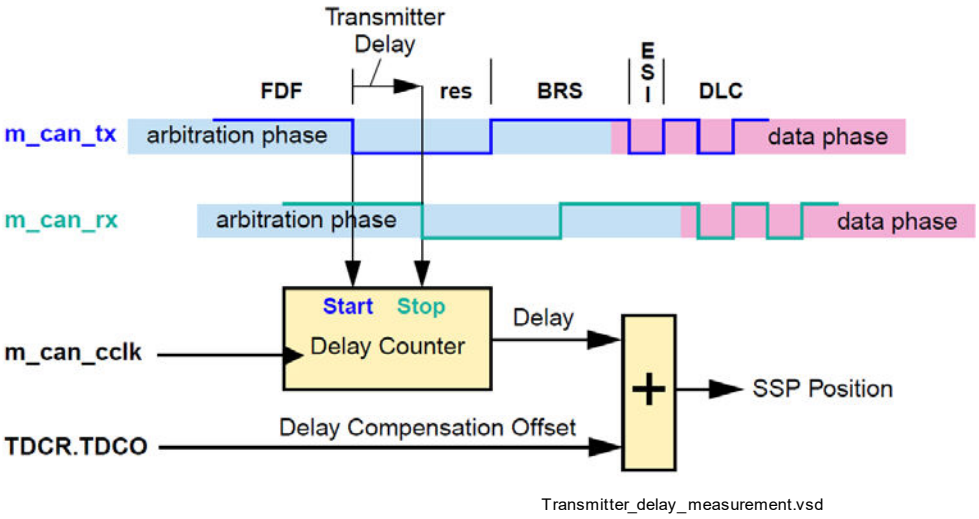


Figure 591 Transmitter delay measurement

To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received “res” bit, resulting in a to early Secondary Sample Point position, the use of a transmitter delay compensation filter window can be enabled by programming **TDCRi (i=0-3)**.TDCF.This defines a minimum value for the Secondary Sample Point position. Dominant edges on RX, that would result in an earlier Secondary Sample Point position are ignored for transmitter delay measurement. The measurement is stopped when the Secondary Sample Point position is at least **TDCRi (i=0-3)**.TDCF AND RX is low.

40.3.2.1.5 Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle to resynchronize itself to the CAN communication. The error counters (**ECRi (i=0-3)**.REC and **ECRi (i=0-3)**.TEC) are frozen while Error Logging (**ECRi (i=0-3)**.CEL) is active. The Host can set the M_CAN into Restricted Operation mode by setting bit CCCRi.ASM. The bit can only be set by the Host when both CCCRi.CCE and CCCRi.INIT are set to ‘1’. The bit can be reset by the Host at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the Host CPU has to reset **CCCRi (i=0-3)**.ASM.

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

40.3.2.1.6 Bus Monitoring Mode

The M_CAN is set in Bus Monitoring Mode by programming **CCCRi (i=0-3)**.MON to one or when error level S3 (**TTOST0**.EL = “11”) is entered. In Bus Monitoring Mode (see ISO11898-1, 10.12 Bus monitoring), the M_CAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the M_CAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the M_CAN monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register TXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. **Figure 592** shows the connection of signals TXD and RXD to the M_CAN in Bus Monitoring Mode.

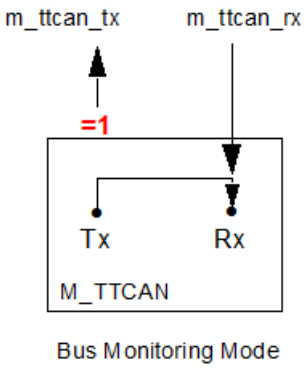


Figure 592 Pin Control in Bus Monitoring Mode

40.3.2.1.7 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898-1, 6.3.3 Recovery Management), the M_CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1, chapter 9.2, the automatic retransmission may be disabled via **CCCRi (i=0-3)**.DAR.

Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer's Tx Request Pending bit **TXBRPi (i=0-3)**.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit **TXBTOi (i=0-3)**.TOx set
  - Corresponding Tx Buffer Cancellation Finished bit **TXBCFi (i=0-3)**.CFx not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit **TXBTOi (i=0-3)**.TOx set
  - Corresponding Tx Buffer Cancellation Finished bit **TXBCFi (i=0-3)**.CFx set
- Arbitration lost or frame transmission disturbed:
  - Corresponding Tx Buffer Transmission Occurred bit **TXBTOi (i=0-3)**.TOx not set
  - Corresponding Tx Buffer Cancellation Finished bit **TXBCFi (i=0-3)**.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = "10" (transmission in spite of cancellation).

40.3.2.1.8 Power Down (Sleep Mode)

The M_CAN can be set into power down mode controlled by input signal clock stop request or via CC Control Register **CCCRi (i=0-3)**.CSR. As long as the clock stop request signal is active, bit **CCCRi (i=0-3)**.CSR is read as one.

When all pending transmission requests have completed, the M_CAN waits until bus idle state is detected. Then the M_CAN sets then **CCCRi (i=0-3)**.INIT to one to prevent any further CAN transfers. Now the M_CAN acknowledges that it is ready for power down by setting **CCCRi (i=0-3)**.CSA to one. In this state, before the clocks are switched off, further register accesses can be made. A write access to **CCCRi (i=0-3)**.INIT will have no effect. Now the module clocks may be switched off.

To leave power down mode, the application has to turn on the module clocks before resetting CC Control Register flag **CCCRi (i=0-3)**.CSR. The M_CAN will acknowledge this by resetting **CCCRi (i=0-3)**.CSA. Afterwards, the application can restart CAN communication by resetting bit **CCCRi (i=0-3)**.INIT.

40.3.2.1.9 Test Modes

To enable write access to register TEST (see **Test Register i**), bit **CCCRi (i=0-3)**.TEST has to be set to one. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin TXD by programming **TESTi (i=0-3)**.TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the M_CAN's bit timing and it can drive constant dominant or recessive values. The actual value at pin RXD can be read from **TESTi (i=0-3)**.RX. Both functions can be used to check the CAN bus' physical layer.

Due to the synchronization mechanism between CAN clock and Host clock domain, there may be a delay of several Host clock periods between writing to **TESTi (i=0-3)**.TX until the new configuration is visible at output pin TXD. This applies also when reading input pin RXD via **TESTi (i=0-3)**.RX.

Note: Test modes should be used for production tests or self test only. The software control for pin TXD interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External Loop Back Mode

The M_CAN can be set in External Loop Back Mode by programming **TESTi (i=0-3)**.LBCK to one. In Loop Back Mode, the M_CAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. **Figure 593** shows the connection of TXD and RXD to the M_CAN in External Loop Back Mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the M_CAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the M_CAN performs an internal feedback from its Tx output to its Rx input. The actual value of the RXD input pin is disregarded by the M_CAN. The transmitted messages can be monitored at the TXD pin.

Internal Loop Back Mode

Internal Loop Back Mode is entered by programming bits **TESTi (i=0-3)**.LBCK and **CCCRi (i=0-3)**.MON to one. This mode can be used for a "Hot Selftest", meaning the M_CAN can be tested without affecting a running CAN system connected to the pins TXD and RXD. In this mode pin RXD is disconnected from the M_CAN and pin TXD is held recessive. **Figure 593** shows the connection of TXD and RXD to the M_CAN in case of Internal Loop Back Mode.

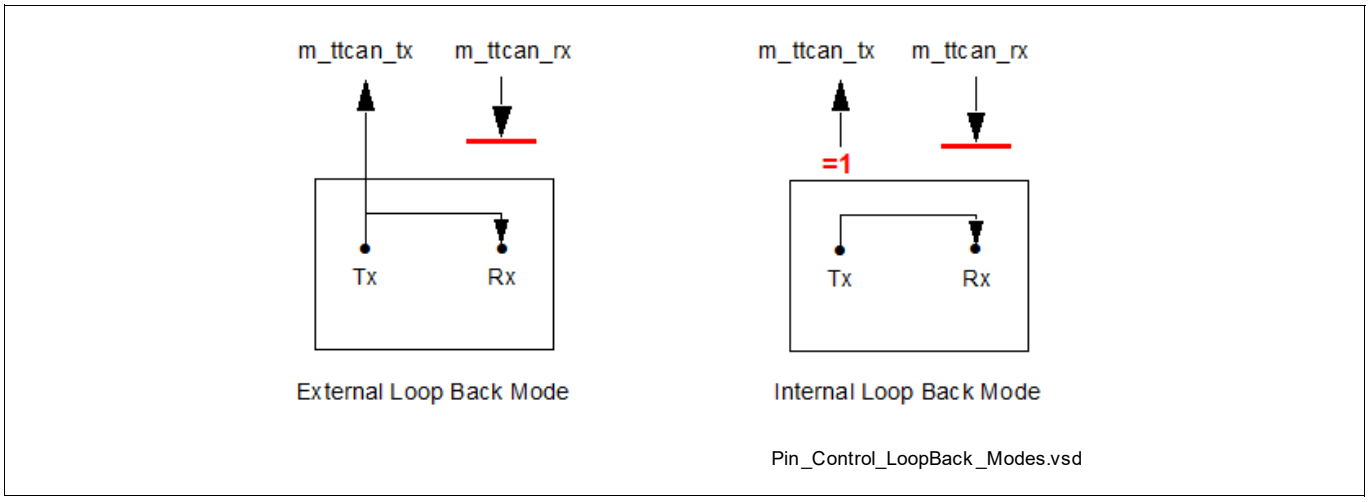


Figure 593 Pin Control in Loop Back Modes

40.3.2.1.10 Application Watchdog

The application watchdog is served by reading register **TTOST0**. When the application watchdog is not served in time, bit **TTOST0**.AWE is set, all TTCAN communication is stopped, and the M_CAN is set into Bus Monitoring Mode.

The TT Application Watchdog can be disabled by programming the Application Watchdog Limit **TTOCF0**.AWL to 0x00. The TT Application Watchdog should not be disabled in a TTCAN application program.

40.3.2.2 Timestamp Generation

For timestamp generation the M_CAN supplies a 16-bit wrap-around counter. A prescaler **TSCCi (i=0-3)**.TCP can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via **TSCVi (i=0-3)**.TSC. A write access to register TSCV resets the counter to zero. When the timestamp counter wraps around interrupt flag **IRi (i=0-3)**.TSW is set.

On start of frame reception / transmission the counter value is captured and stored into the timestamp section of an Rx Buffer / Rx FIFO (RXTS[15:0]) or Tx Event FIFO (TXTS[15:0]) element.

By programming bit **TSCCi (i=0-3)**.TSS an external 16-bit timestamp can be used. The external timer can be controlled using the **NTCCRi (i=0-3)** register. The external timer can be started by setting **NTCCRi (i=0-3)**.STSTART. It can be reset to zero by setting **NTCCRi (i=0-3)**.STRESET. A write of '1b' to **NTCCRi (i=0-3)**.STSTART is effective only when **NTCCRi (i=0-3)**.STRESET is '0b' ie., the external timer cannot be started when **NTCCRi (i=0-3)**.STRESET is '1b'.

40.3.2.3 Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the M_CAN supplies a 16-bit Timeout Counter. It operates as down-counter and uses the same prescaler controlled by **TSCCi (i=0-3)**.TCP as the Timestamp Counter. The Timeout Counter is configured via register **TOCCi (i=0-3)**. The actual counter value can be read from **TOCVi (i=0-3)**.TOC.

The Timeout Counter can only be started while **CCCRi (i=0-3)**.INIT = '0'. It is stopped when **CCCRi (i=0-3)**.INIT = '1', e.g. when the M_CAN enters Bus_Off state.

The operation mode is selected by **TOCCi (i=0-3)**.TOS. When operating in Continuous Mode, the counter starts when CCCRi.INIT is reset. A write to **TOCVi (i=0-3)** presets the counter to the value configured by **TOCCi (i=0-3)**.TOP and continues down-counting.

When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by **TOCCi (i=0-3)**.TOP. Down-counting is started when the first FIFO element is stored. Writing to **TOCVi (i=0-3)** has no effect.

When the counter reaches zero, interrupt flag **IRi (i=0-3)**.TOO is set. In Continuous Mode, the counter is immediately restarted at **TOCCi (i=0-3)**.TOP.

Note: The clock signal for the Timeout Counter is derived from the CAN Core's sample point signal. Therefore the point in time where the Timeout Counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the baud rate switch feature in CAN FD is used, the timeout counter is clocked differently in arbitration and data field.

40.3.2.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's Put and Get Indices.

40.3.2.4.1 Acceptance Filtering

The M_CAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from - to)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration **GFCi (i=0-3)**
- Standard ID Filter Configuration **SIDFCi (i=0-3)**
- Extended ID Filter Configuration **XIDFCi (i=0-3)**
- Extended ID AND Mask **XIDAMi (i=0-3)**

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag **IRi (i=0-3)**.HPM
- Set High Priority Message interrupt flag **IRi (i=0-3)**.HPM and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32-bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer
New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type see **PSRi (i=0-3)**.LEC respectively **PSRi (i=0-3)**.FLEC.

Rx FIFO
Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type see **PSRi (i=0-3)**.LEC respectively **PSRi (i=0-3)**.FLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in **Rx FIFO Overwrite Mode** have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

Range Filter

The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

EFT = “00”: The Message ID of received frames is ANDed with the Extended ID AND Mask (XIDAM) before the range filter is applied

EFT = “11”: The Extended ID AND Mask (XIDAM) is not used for range filtering

Filter for specific IDs

A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

Classic Bit Mask Filter

Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

In case all mask bits are one, a match occurs only when the received Message ID and the Message ID filter are identical. If all mask bits are zero, all Message IDs match.

Standard Message ID Filtering

Figure 594 below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in **Section 40.4.6.5**.

Controlled by the Global Filter Configuration **GFCi (i=0-3)** and the Standard ID Filter Configuration SIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

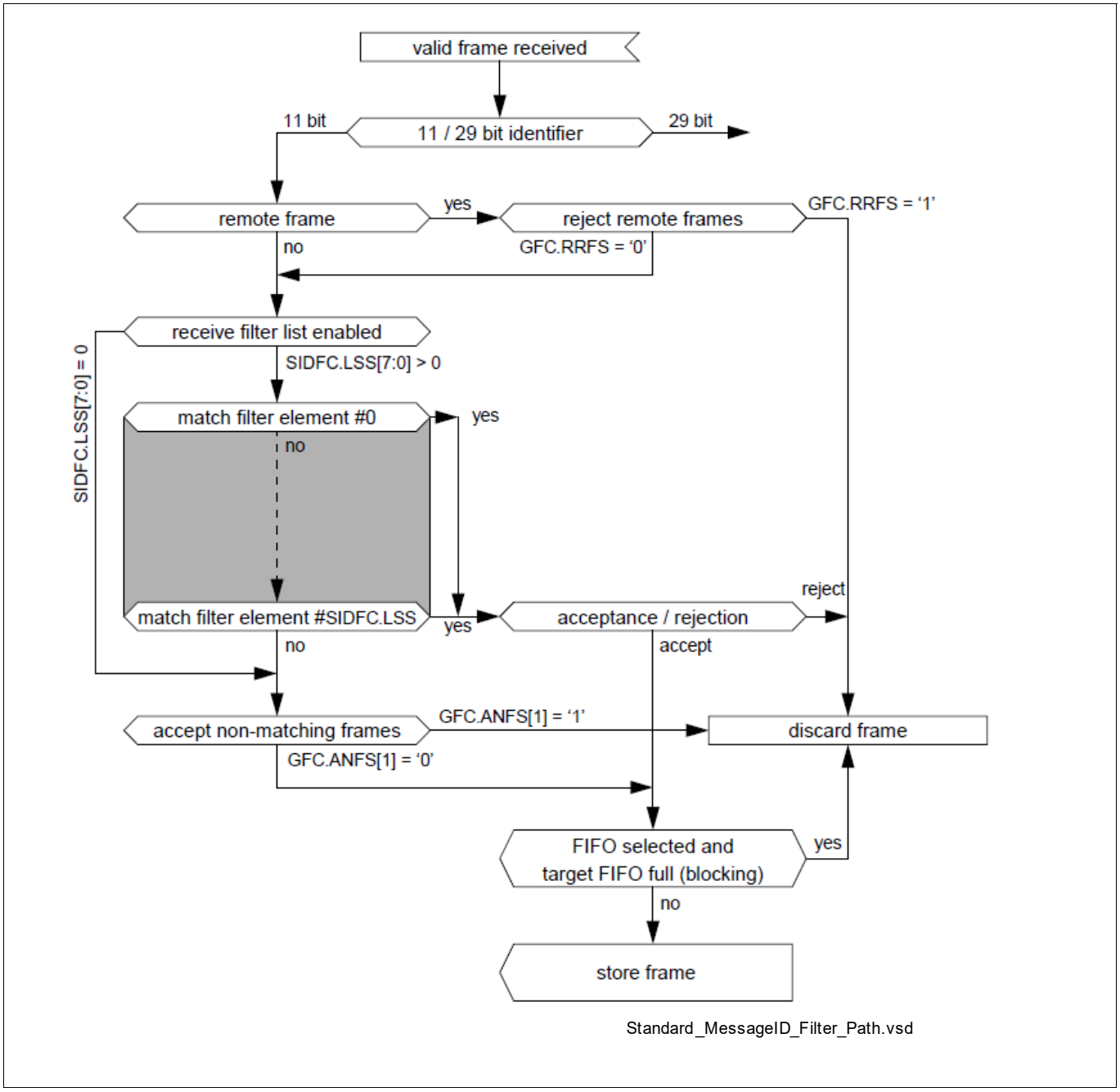


Figure 594 Standard Message ID Filter Path

Extended Message ID Filtering

Figure 595 below shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in **Section 40.4.6.6**.

Controlled by the Global Filter Configuration GFC and the Extended ID Filter Configuration XIDFC Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask XIDAM is ANDed with the received identifier before the filter list is executed.

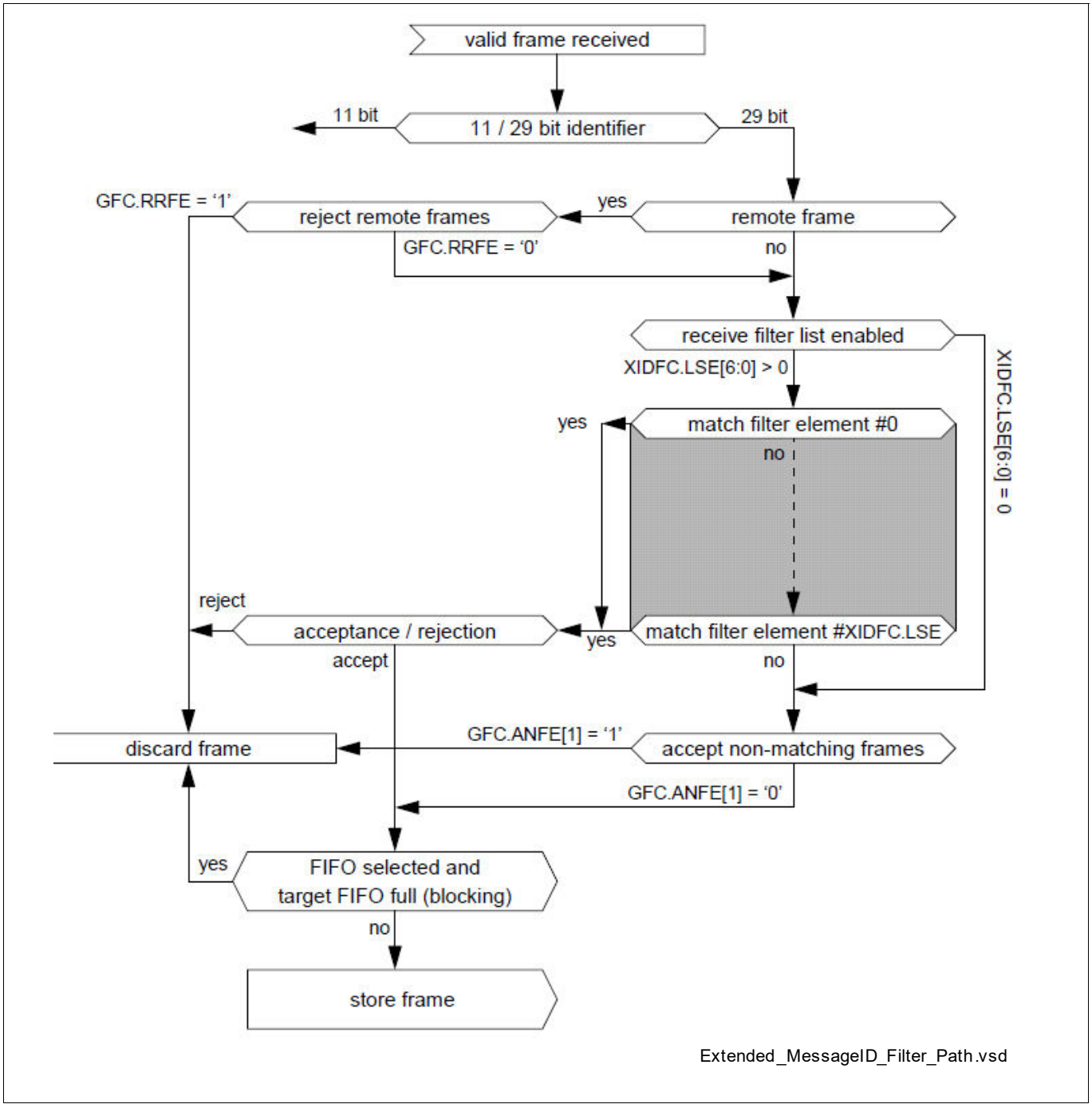


Figure 595 Extended Message ID Filter Path

40.3.2.4.2 Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers **RXF0Ci (i=0-3)** and **RXF1Ci (i=0-3)**.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 see **Section 40.3.2.4.1**. The Rx FIFO element is described in **Section 40.4.6.2**.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by RXFnC.FnWM, interrupt flag IRi.RFnW is set. When the Rx FIFO Put Index reaches the Rx

FIFO Get Index an Rx FIFO Full condition is signalled by RXFnS.FnF. In addition interrupt flag **IRi (i=0-3)**.RFnF is set.

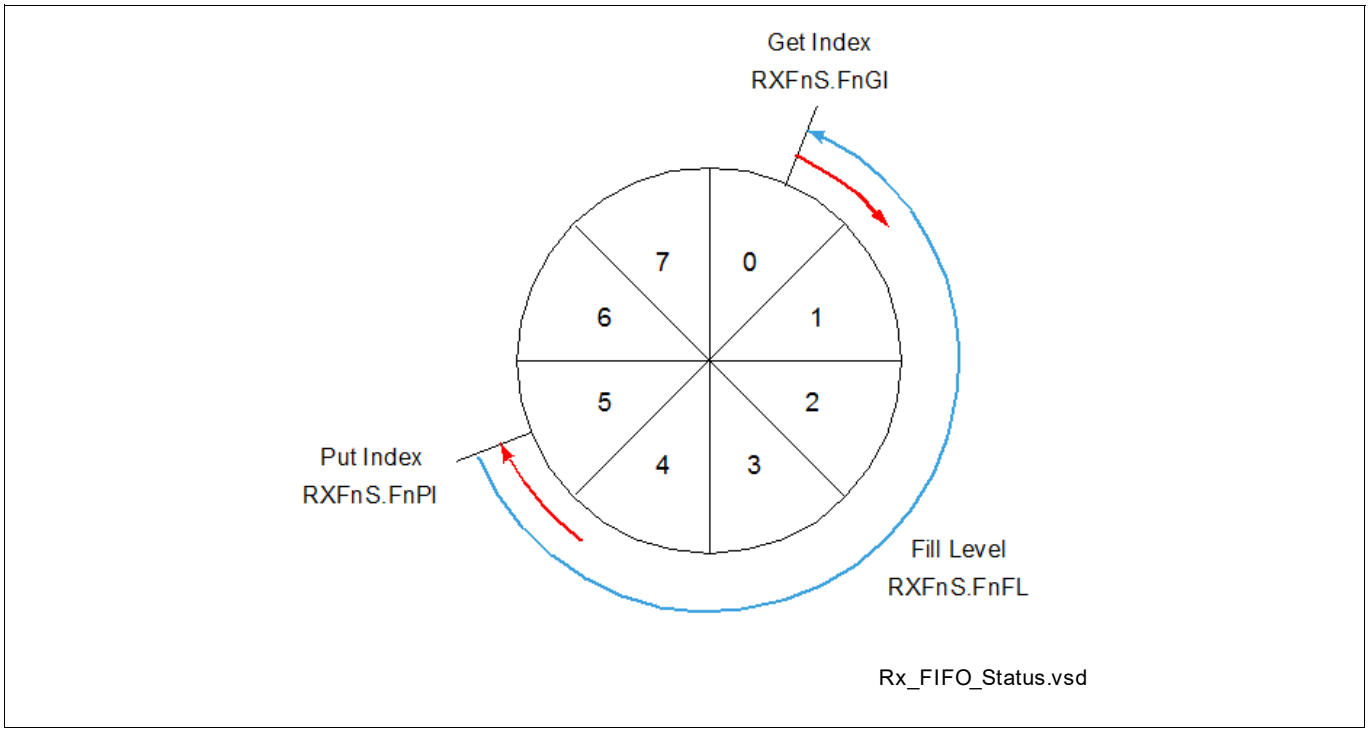


Figure 596 Rx FIFO Status

When reading from an Rx FIFO, Rx FIFO Get Index RXFnS.FnGI • FIFO Element Size has to be added to the corresponding Rx FIFO start address RXFnC.FnSA.

Table 367 Rx Buffer / FIFO Element Size

RXESCi.RBDS[2:0] RXESCi.FnDS[2:0]	Data Field [bytes]	FIFO Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

Rx FIFO Blocking Mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = ‘0’. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by RXFnS.FnF = ‘1’. In addition interrupt flag IRi.RFnF is set.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RXFnS.RFnL = ‘1’. In addition interrupt flag IRi.RFnL is set.

Rx FIFO Overwrite Mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = ‘1’.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signalled by RXFnS.FnF = ‘1’, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. **Figure 597** shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

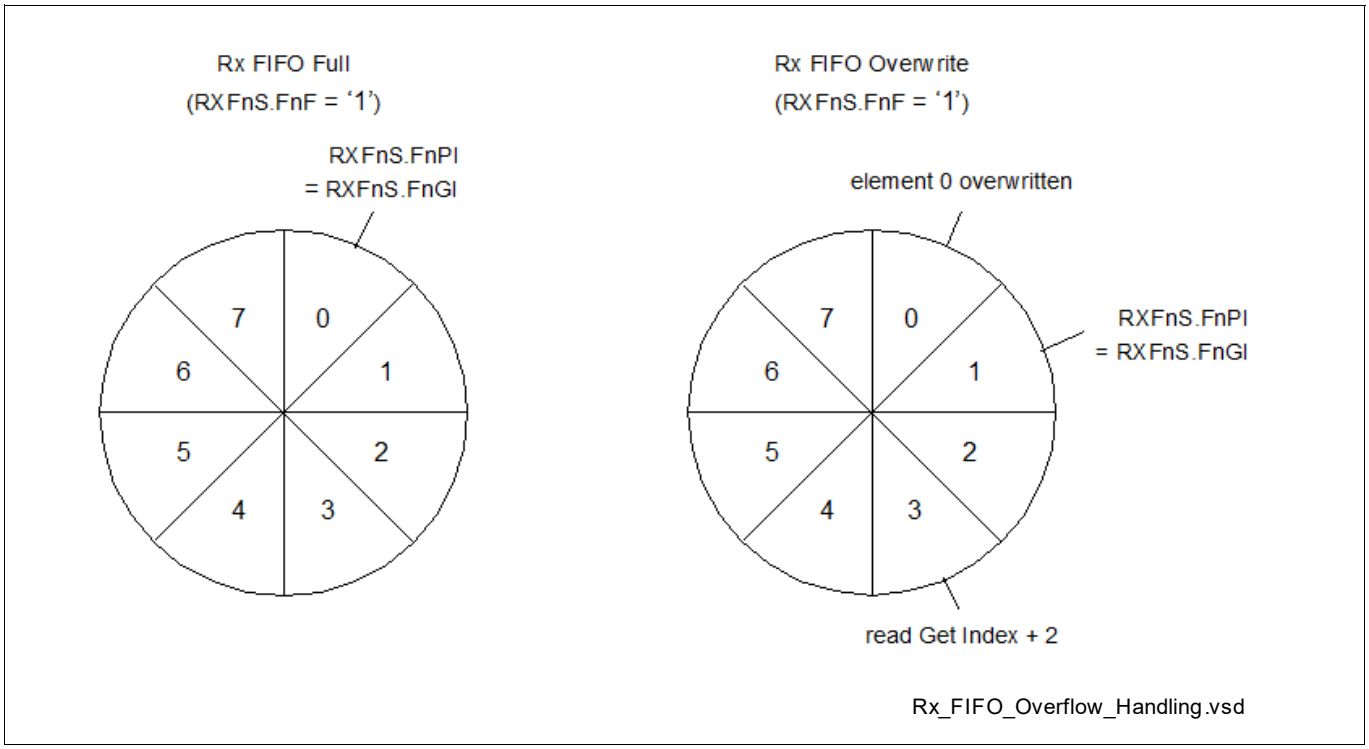


Figure 597 Rx FIFO Overflow Handling

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = ‘0’).

40.3.2.4.3 Dedicated Rx Buffers

The M_CAN supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via **RXBCi (i=0-3)**.RBSA.

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = “111” and SFID2 / EFID2[10:9] = “00” has to be configured (see **Section 40.4.6.5** and **Section 40.4.6.6**).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag IRi.DRX (Message stored in dedicated Rx Buffer) in the interrupt register is set.

Table 368 Example Filter Configuration for Rx Buffers

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register **NDAT1i (i=0-3)**,**NDAT2i (i=0-3)** is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the Host by writing a '1' to the respective bit position.

While an Rx Buffer's New Data flag is set, a Message ID Filter Element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. Following Message ID Filter Elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

Rx Buffer Handling

- Reset interrupt flag IRi.DRX
- Read New Data registers
- Read messages from Message RAM
- Reset New Data flags of processed messages

40.3.2.5 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The CAN mode for transmission (Classic CAN or CAN FD) can be configured separately for each Tx Buffer element. The Tx Buffer element is described in **Section 40.4.6.3**. **Table** below describes the possible configurations for frame transmission.

Table 369 Possible Configuration for Frame Transmission

CCCR		Tx Buffer Element		Frame Transmission
BRSE	FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classical CAN
0	1	0	ignored	Classical CAN
0	1	1	ignored	FD without bit rate switching
1	1	0	ignored	Classical CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation

The Tx Handler starts a Tx scan to check for the highest priority pending Tx request (Tx Buffer with lowest Message ID) when the Tx Buffer Request Pending register **TXBRPi (i=0-3)** is updated, or when a transmission has been started.

40.3.2.5.1 Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU's CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit **CCCRi (i=0-3)**.TXP. If the bit is set, the M_CAN will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (**CCCRi (i=0-3)**.TXP = '0').

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

40.3.2.5.2 Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the Host CPU. Each dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an "Add Request" via **TXBARi (i=0-3)**.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx Queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A dedicated Tx Buffer allocates Element Size 32-bit words in the Message RAM (see **Chapter 40.3.2.5.3**). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0…31) • Element Size to the Tx Buffer Start Address **TXBCi (i=0-3)**.TBSA.

Table 370 Tx Buffer / FIFO / Queue Element Size

TXESCi.TBDS[2:0]	Data Field [bytes]	Element Size [RAM words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

40.3.2.5.3 Tx FIFO

Tx FIFO operation is configured by programming **TXBCi (i=0-3)**.TFQM to '0'. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the Get Index **TXFQSi (i=0-3)**.TFGI. After each transmission the Get Index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The M_CAN calculates the Tx FIFO Free Level **TXFQSi (i=0-3)**.TFFL as difference between Get and Put Index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the Put Index **TXFQSi (i=0-3)**.TFQPI. An "Add Request" increments the Put Index to the next free Tx FIFO element. When the Put Index reaches the Get Index, Tx FIFO Full (**TXFQSi (i=0-3)**.TFQF = '1') is signalled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a '1' to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO's Put Index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via **TXBARi (i=0-3)**. The Put Index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the Get Index is cancelled, the Get Index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO Free Level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the Get Index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates Element Size 32-bit words in the Message RAM (see **Table 370**). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding Tx FIFO/Queue Put Index **TXFQSi (i=0-3)**.TFQPI (0…31) • Element Size to the Tx Buffer Start Address **TXBCi (i=0-3)**.TBSA.

40.3.2.5.4 Tx Queue

Tx Queue operation is configured by programming **TXBCi (i=0-3)**.TFQM to '1'. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index **TXFQSi (i=0-3)**.TFQPI. An "Add Request" cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (**TXFQSi (i=0-3)**.TFQF = "1"), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register TXBRP instead of the Put Index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates Element Size 32-bit words in the Message RAM (see **Table 370**). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue Put Index **TXFQSi (i=0-3)**.TFQPI (0…31) • Element Size to the Tx Buffer Start Address **TXBCi (i=0-3)**.TBSA.

40.3.2.5.5 Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx FIFO. The number of Dedicated Tx Buffers is configured by TXBCi.NDTB. The number of Tx Buffers assigned to the Tx FIFO is configured by **TXBCi (i=0-3)**.TFQSi. In case **TXBCi (i=0-3)**.TFQS is programmed to 0x0, only dedicated Tx Buffers are used.

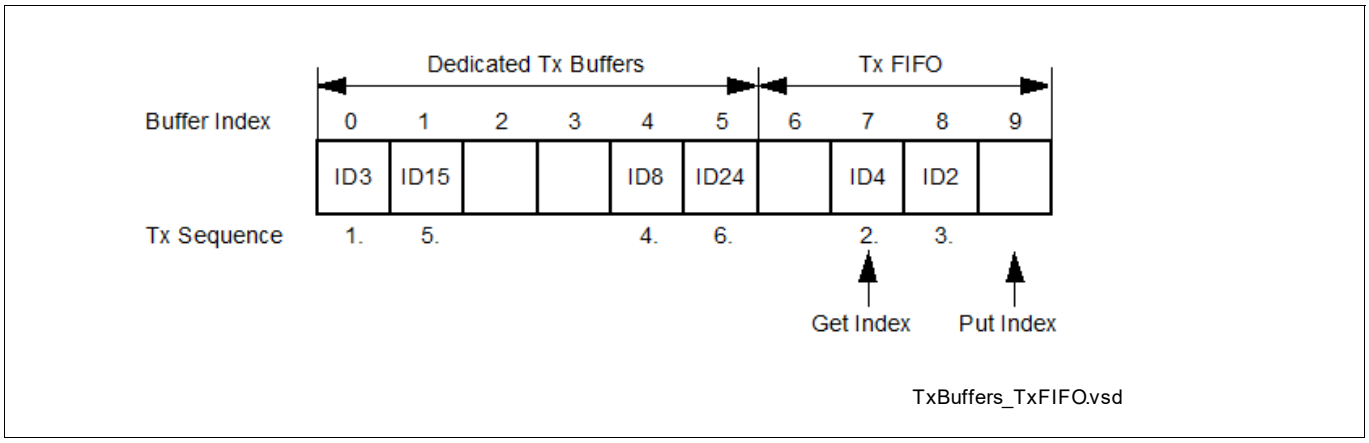


Figure 598 Example of mixed Configuration Dedicated Tx Buffers / Tx FIFO

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by **TXFQSi (i=0-3)**.TFGI)
- Buffer with lowest Message ID gets highest priority and is transmitted next

40.3.2.5.6 Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of Dedicated Tx Buffers and a Tx Queue. The number of Dedicated Tx Buffers is configured by **TXBCi (i=0-3)**.NDTB. The number of Tx Queue Buffers is configured by **TXBCi (i=0-3)**.TFQSi. In case **TXBCi (i=0-3)**.TFQS is programmed to zero, only Dedicated Tx Buffers are used.

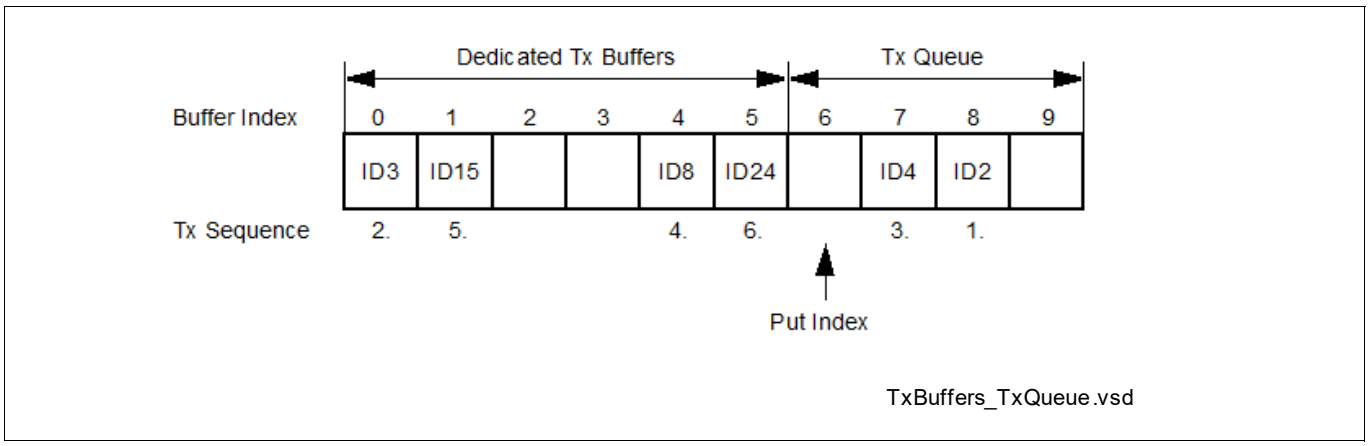


Figure 599 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

40.3.2.5.7 Transmit Cancellation

The M_CAN supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a Dedicated Tx Buffer or a Tx Queue Buffer the Host has to write a '1' to the corresponding bit position (=number of Tx Buffer) of register **TXBCRi (i=0-3)**. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signalled by setting the corresponding bit of register TXBCF to '1'.

In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding TXBTO and TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding TXBCF bit is set.

Note: In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

40.3.2.5.8 Tx Event Handling

To support Tx event handling the M_CAN has implemented a Tx Event FIFO. After the M_CAN has transmitted a message on the CAN bus, Message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the Message Marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in **Section 40.4.6.4**.

The purpose of the Tx Event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx Buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx Event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx Buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx Buffer before overwriting that Tx Buffer.

When a Tx Event FIFO full condition is signalled by **IRi (i=0-3)**.TEFF, no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO Get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag **IRi (i=0-3)**.TEFL is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by **TXEFCi (i=0-3)**.EFWM, interrupt flag IRi.TEFW is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO Get Index **TXEFSi (i=0-3)**.EFGI has to be added to the Tx Event FIFO start address **TXEFCi (i=0-3)**.EFSA.

40.3.2.6 FIFO Acknowledge Handling

The Get Indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (see **Rx FIFO 0 Acknowledge i**, **Rx FIFO 1 Acknowledge i**, and **Tx Event FIFO Acknowledge i**). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

When only a single element has been read from the FIFO (the one being pointed to by the Get Index), this Get Index value is written to the FIFO Acknowledge Index.

When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO's Get Index.

Due to the fact that the CPU has free access to the M_CAN's Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (Get Index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the Get Index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The M_CAN does not check for erroneous values.

40.3.2.7 OCDS Suspend Behaviour Support

In case of Hard Suspend, the clock is switched off to the MCMCAN module, and any ongoing transmission or reception event will be suspended immediately.

In case of Soft Suspend, the clock is switched off to the MCMCAN module after completion of ongoing transmission and reception events. The CCCR.INIT bit will be set during the Soft Suspend. Hence, when leaving the Soft Suspend, the debugger has to ensure clearing of the CCCR.INIT bit, inorder to continue normal CAN operation.

40.3.3 TTCAN Operation

40.3.3.1 Reference Message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the Time Master (sender of the reference message).

For Level 1 the data length must be at least one; for Level 0,2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to 8 potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

40.3.3.1.1 Level 1

Level 1 operation is configured via TTOCF0.OM = “01” and TTOCF0.GEN. External clock synchronization is not available in Level 1.

The information related to the reference message is stored in the first data byte as shown in **Table 371** below. Cycle_Count is optional.

Table 371 First byte of Level 1 reference message

Bits	7	6	5	4	3	2	1	0
First Byte	Next_is_Gap	0	Cycle_Count[5:0]					

40.3.3.1.2 Level 2

Level 2 operation is configured via TTOCF0.OM = “10” and TTOCF0.GEN.

The information related to the reference message is stored in the first four data bytes as shown in **Table 372** below. Cycle_Count and the lower four bits of NTU_Res are optional. The M_CAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

Table 372 First four bytes of Level 2 reference message

Bits	7	6	5	4	3	2	1	0
First Byte	Next_is_Gap	0	Cycle_Count[5:0]					
Second Byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third Byte	Master_Ref_Mark[7:0]							
Fourth Byte	Master_Ref_Mark[15:8]							

40.3.3.1.3 Level 0

Level 0 operation is configured via TTOCF0.OM = “11”. External event-synchronized time-triggered operation is not available in Level 0.

The information related to the reference message is stored in the first four data bytes as shown in **Table 372** below. In Level 0 Next_is_Gap is always zero. Cycle_Count and the lower four bits of NTU_Res are optional. The M_CAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as zero.

Table 373 First four bytes of Level 0 reference message

Bits	7	6	5	4	3	2	1	0
First Byte	Next_is_Gap	0	Cycle_Count[5:0]					
Second Byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third Byte	Master_Ref_Mark[7:0]							
Fourth Byte	Master_Ref_Mark[15:8]							

40.3.3.2 TTCAN Configuration

40.3.3.2.1 TTCAN Timing

The Network Time Unit NTU is the unit in which all times are measured. The NTU is a constant of the whole network and is defined a priori by the network system designer. In TTCAN Level 1 the NTU is the nominal CAN bit time. In TTCAN Level 0 and Level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN Level 0 and Level 2 the length of the NTU is defined by the Time Unit Ratio TUR. The TUR is in principle a non-integer number and given by the formula TUR = TURNA0.NAV/TURCF0.DC. The length of the NTU is given by the formula NTU = CAN Clock Period • TUR.

The TUR Numerator Configuration NC is an 18-bit number, TURCF0.NCL[15:0] can be programmed in the range 0x0000-0xFFFF. TURCF0.NCH[17:16] is hard wired to 0b01. When the number 0xnnnn is written to TURCF0.NCL[15:0], TURNA0.NAV starts with the value 0x10000 + 0x0nnnn = 0x1nnnn. The TUR Denominator Configuration TURCF0.DC is a 14-bit number. TURCF0.DC may be programmed in the range 0x0001 - 0x3FFF, 0x0000 is an illegal value.

In Level 1, NC must be ≥ 4 • TURCF0. DC. In Level 0,2 NC must be ≥ 8 • TURCF0.DC to allow the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets TURCF0.DC to 0x1000 and TURCF0.NCL to 0x10000, resulting in an NTU consisting of 16 CAN clock periods. Local time and application watchdog are not started before either the CCCRi.INIT is reset, or TURCF0.ELT is set. TURCF0.ELT may not be set before the NTU is configured. Setting TURCF0.ELT to ‘1’ also locks the write access to register TURCF0.

At startup TURNA0.NAV is updated from NC (= TURCF0.NCL + 0x10000) when TURCF0.ELT is set. In TTCAN Level 1 there is no drift compensation. TURNA0.NAV does not change during operation, it always equals NC.

In TTCAN Level 0 and Level 2 there are two possibilities for TURNA0.NAV to change. When operating as time slave or backup time master, and when TTOCF0.ECC is set, TURNA0.NAV is updated automatically to the value calculated from the monitored global time speed, as long as the M_CAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization it returns to NC. When operating as the actual time master, and when TTOCF0.EECS is set, the Host may update TURCF0.NCL. When the Host sets TTOCN0.ECS, TURNA0.NAV will be updated from the new value of NC at the next reference message. The status flag TTOST0.WECS as is set when TTOCN0.ECS is set and is cleared when TURNA0.NAV is updated. TURCF0.NCL is write locked while TTOST0.WECS is set.

In TTCAN Level 0 and Level 2 the clock calibration process adapts TURNA0.NAV in the range of the Synchronization Deviation Limit SDL of NC ± $2^{(TTOCF0.LDSDL+5)}$. TURCF0.NCL should be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of TURNA0.NAV.

The synchronization deviation SD is the difference between NC and TURNA0.NAV (SD = |NC - TURNA0.NAV|). It is limited by the Synchronization Deviation Limit SDL, which is configured by its dual logarithm TTOCF0.LDSDL (SDL = $2^{(TTOCF0.LDSDL+5)}$) and should not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new Basic Cycle. When the calculated TURNA0.NAV deviates by more than SDL from NC, or if the Disc_Bit in the reference message is set, the drift compensation is suspended and TTIR0.GTE is set and TTOSC.QCS is reset, or in case of the Disc_Bit = ‘1’, TTIR0.GTD is set.

TUR configuration examples are shown in **Table 374** below.

Table 374 TUR Configuration Examples

TUR	8	10	24	50	510	125000	32.5	100/12	529/17
NC	0x1FFF8	0x1FFFE	0x1FFF8	0x1FFEA	0x1FFFE	0x1E848	0x1FFE0	0x19000	0x10880
TURCF0.DC	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

TTOCN0.ECS schedules NC for activation by the next reference message. TTOCN0.SGT schedules TTGTP0.TP for activation by the next reference message. Setting of TTOCN0.ECS and TTOCN0.SGT requires TTOCF0.EECS to be set (external clock synchronization enabled) while the M_CAN is actual time master.

The M_CAN module provides an application watchdog to verify the function of the application program. The Host has to serve this watchdog regularly, else all CAN bus activity is stopped. The Application Watchdog Limit TTOCF0.AWL specifies the number of NTUs between two times the watchdog has to be served. The maximum number of NTUs is 256. The Application Watchdog is served by reading register TTOST0. TTOST0.AWE indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, interrupt flag TTIR0.AW is set. For software development, the application watchdog may be disabled by programming TTOCF0.AWL to 0x00 (see also **Chapter 40.3.2.1.10**).

40.3.3.2.2 Message Scheduling

TTOCF0.TM controls whether the M_CAN operates as a potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message's identifier TTRMC.RID define the master priority, 0 giving the highest and 7 giving the lowest priority. There may not be two nodes in the network using the same master priority. TTRMC.RID is used for recognition of reference messages. TTRMC.RMPS is not relevant for time slaves.

The Initial Reference Trigger Offset TTOCF0.IRTO is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for TTOCF0.IRTO is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, should correspond to their master priority, even with maximum clock drift.

TTOCF0.OM decides whether the node operates in TTCAN Level 0, Level 1, or Level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on Level 1 in a Level 2 network, but not vice versa. The configuration of the TTCAN operation mode via TTOCF0.OM is the last step in the setup. With TTOCF0.OM = "00" (event-driven CAN communication), the M_CAN operates according to ISO 11898-1, without time triggers. With TTOCF0.OM = "01" (Level 1), the M_CAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With TTOCF0.OM = "10" (Level 2), the M_CAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With TTOCF0.OM = "11" (Level 0), the M_CAN operates as event-driven CAN but maintains a calibrated global time base as in Level 2.

TTOCF0.EECS enables the external clock synchronisation, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in Level 0,2 only) the global clock phase to an external reference.

TTMLM.ENTT in the TT Matrix Limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger's repeat factors have to be taken into account. An inaccurate configuration of TTMLM.ENTT will result in either a Tx Count Underflow (TTIR0.TXU = '1' and TTOST0.EL = "01", severity 1) or in a Tx Count Overflow (TTIR0.TXO = '1' and TTOST0.EL = "10", severity 2).

Note: In case the first reference message seen by a node does not have Cycle_Count zero, this node may finish its first matrix cycle with its Tx count resulting in a Tx Count Underflow condition. As long as a node is in state Synchronizing its Tx_Triggers will not lead to transmissions.

TTMLM.CCM specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of 8 basic cycles TTMLM.CCM would be 7. TTMLM.CCM is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

TTMLM.TXEW specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If the sample point of the first bit of a transmit message is not inside the Tx enable window because of e.g. a slight overlap from the previous time window's message, the transmission cannot be started in that time window at all. TTMLM.TXEW has to be chosen with respect to the network's synchronisation quality and with respect to the relation between the length of the time windows and the length of the messages.

40.3.3.2.3 Trigger Memory

The trigger memory is part of the external Message RAM to which the M_CAN is connected via its Generic Master Interface (see **Figure 605**). It stores up to 64 trigger elements. A trigger memory element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, Filter Type FTYPE, Message Number MNR, Message Status Count MSC, Time Mark Event Internal TMIN andTime Mark Event External TMEX.

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory have to be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the TTCAN node switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a transmit trigger, the Tx Handler starts to read the message from the Message RAM as soon as the TTCAN node switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx Handler has to be able to prepare the transmission before the transmit trigger's time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mark is reached. If the reference message is n NTU long, a trigger with a time mark< n will never become active and will be treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message's start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger's time mark. The M_CAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialisation) be seen at a value< n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger's time mark+1 NTU+1 CAN bit time

The trigger list will be different for all nodes in the TTCAN network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

Trigger Types

Tx_Ref_Trigger (TYPE = "0000") and Tx_Ref_Trigger_Gap (TYPE = "0001") cause the transmission of a reference message by a time master. A configuration error (TTOST0.EL = "11", severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronised time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the M_CAN synchronisation state is In_Gap (TTOST0.SYS = "10").

Tx_Trigger_Single (TYPE = "0010"), Tx_Trigger_Continous (TYPE = "0011"), Tx_Trigger_Arbitration (TYPE = "0100"), and Tx_Trigger_Merged (TYPE = "0101") cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit is reset.

Tx_Trigger_Continous starts a transmission in an exclusive time window when the message buffer's Transmission Request Pending bit is set. After successful transmission the Transmission Request Pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A Configuration Error (TTOST0.EL = "11", severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other

Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code has to be considered when the expected number of Tx_Triggers (TTMLM.ENTT) is calculated.

Watch_Trigger (TYPE = “0110”) and Watch_Trigger_Gap (TYPE = “0111”) check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the M_CAN synchronisation state is In_Gap (TTOST0.SYS = “10”).

Rx_Trigger (TYPE = “1000”) is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger shall be placed after the end of that message’s transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements which are referenced by Rx_Triggers should be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger’s time mark is reached.

Time_Base_Trigger (TYPE = “1001”) are used to generate internal/external events depending on the configuration of ASC, TMIN, and TMEX.

End_of_List (TYPE = “1010…1111”) is an illegal trigger type, a configuration error (TTOST0.EL = “11”, severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger or Watch_Trigger_Gap.

Restrictions for the Node’s Trigger List

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when TTOST0.SYS = “10”) will never become active. The watch triggers themselves will not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when TTOST0.SYS = “10”) must be set to trigger type End_of_List.

A typical trigger list for a potential time master will begin with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave will be the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger’s time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the Reference Trigger Offset TTOCF0.IRTO. In case error level 2 is reached (TTOST0.EL = “10”), the effective time mark is the sum of its time mark and 0x127. No other trigger elements should be placed in this range otherwise it may happen, that the time marks appear out of order

and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- Host clock frequency
- Speed and waiting time for Trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

Example for Trigger Handling

The example below shows how the trigger list is derived from a node's system matrix. Assumed node A is first time master and has knowledge of the section of the system matrix shown in **Table 375** below.

Table 375 System Matrix Node A

Cycle Count	Time Mark1	Time Mark2	Time Mark3	Time Mark4	Time Mark5	Time Mark6	Time Mark7
0	Tx7					TxRef	Error
1	Rx3		Tx2, Tx4			TxRef	Error
2						TxRef	Error
3	Tx7		Rx5			TxRef	Error
4	Tx7			Rx6		TxRef	Error

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the number of basic cycles in the system matrix is 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by TTMLM.CCM. The Cycle Code CC is composed of repeat factor (= value of most significant '1') and the number of the first basic cycle in the system matrix (= bit field after most significant '1').

Example:with a cycle code of 0b0010011 (repeat factor: 16, first basic cycle: 3) and a maximum cycle count of TTMLM.CCM = "0x3F"matches occur at cycle counts 3, 19, 35, 51

A trigger element consists of Time Mark TM, Cycle Code CC, Trigger Type TYPE, and Message Number MNR. For transmission MNR references the Tx Buffer number (0…31). For reception MNR references the number of the filter element (0…127) that matched during acceptance filtering. Depending on the configuration of the Filter Type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition a trigger element can be configured for Asynchronous Serial Communication ASC, generation of Time Mark Event Internal TMIN, and Time Mark Event External TMEX. The Message Status Count MSC holds the counter value (0…7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Table 376 Trigger List Node A

Trigger	Time Mark TM[15:0]	Cycle Code CC[6:0]	Trigger Type TYPE[3:0]	Mess. No. MNR[6:0]
0	Mark1	0b0000100	Tx_Trigger_Single	7
1	Mark1	0b1000000	Rx_Trigger	3
2	Mark1	0b1000011	Tx_Trigger_Single	7
3	Mark3	0b1000001	Tx_Trigger_Merged	2

Table 376 Trigger List Node A (cont'd)

Trigger	Time Mark TM[15:0]	Cycle Code CC[6:0]	Trigger Type TYPE[3:0]	Mess. No. MNR[6:0]
4	Mark3	0b1000011	Rx_Trigger	5
5	Mark4	0b1000001	Tx_Trigger_Arbitration	4
6	Mark4	0b1000100	Rx_Trigger	6
7	Mark6	n.a.	Tx_Ref_Trigger	0 (Ref)
8	Mark7	n.a.	Watch_Trigger	n.a.
9	n.a.	n.a.	End_of_List	n.a.

Tx_Trigger_Single, Tx_Trigger_Continous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from zero until a trigger with time mark > cycle time and with its Cycle Code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the Time Mark TM, the action defined by Trigger Type TYPE and Message Number MNR is started. There is an error in the configuration when End_of_List is reached.

At Mark6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the Watch Trigger at Mark7 is reached, the node was not able to transmit the reference message; error treatment is started.

Detection of Configuration Errors

A configuration error is signalled via TTOST0.EL = "11" (severity 3) when:

The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.

The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger.

The FSE of a node with TTOCF0.TM='0' (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.

Any time mark placed inside the Tx enable window (defined by TTMLM.TXEW) of a Tx_Trigger with a matching cycle code.

A time mark is placed near the time mark of a Tx_Ref_Trigger and the Reference Trigger Offset TTOST0.RTO causes a reversal of their sequential order measured in cycle time.

40.3.3.2.4 TTCAN Schedule Initialization

The synchronisation to the M_CAN's message schedule starts when CCCRi.INIT is reset. The M_CAN can operate strictly time-triggered (TTOCF0.GEN = '0') or external event-synchronized time-triggered (TTOCF0.GEN = '1'). All nodes start with cycle time zero at the beginning of their trigger list with TTOST0.SYS = "00" (out of synchronisation), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronised time-triggered operation mode will ignore Tx_Ref_Trigger and Watch_Trigger and will use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

Time Slaves

After configuration, a time slave will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag TTIR0.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). The first received reference message will restart the FSE and the cycle time.

Note: Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter which counts up to 0xFFFF = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it will assume a severe error (TTOST0.EL = “11”, severity 3), set interrupt flag TTIR0.WT, switch off its CAN bus output, and enter the bus monitoring mode (CCCRi.MON set to ‘1’). In the bus monitoring mode it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

Note: To leave the severe error state, the Host has to set CCCRi.INIT = ‘1’. After reset of CCCRi.INIT, the node restarts TTCAN communication.

When no error is encountered during synchronisation, the first reference message sets TTOST0.SYS = “01” (Synchronizing), the second sets the TTCAN synchronization state (depending on its Next_is_Gap bit) to TTOST0.SYS = “11” (In_Schedule) or TTOST0.SYS = “10” (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

Potential Time Masters

After configuration, a potential time master will start the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It will ignore its Watch_Trigger and Watch_Trigger_Gap when it did not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag TTIR0.IWT is set, the FSE is frozen, and the cycle time will become invalid, but the node will still be able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting TTIR0.IWT will re-enable the transmission of reference messages until next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE will be restarted by the reception of a reference message.

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it will assume a severe error (TTOST0.EL = “11”, severity 3), set interrupt flag TTIR0.WT, switch off its CAN bus output, and enter the bus monitoring mode (CCCRi.MON set to ‘1’). In bus monitoring mode, it is still able to receive messages, but it cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets TTOST0.SYS = “01” (synchronizing), the second sets the TTCAN synchronization state (depending on its Next_is_Gap bit) to TTOST0.SYS = “11” (In_Schedule) or TTOST0.SYS = “10” (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (TTOST0.MS = “11”) when it was the transmitter of the last reference message, else it is backup time master (TTOST0.MS = “10”).

When all potential time masters have finished configuration, the node with the highest time master priority in the network will become the current time master.

40.3.3.3 TTCAN Gap Control

All functions related to Gap control apply only when the M_CAN is operated in external event- synchronized time-triggered mode (TTOCF0.GEN = ‘1’). In this operation mode the TTCAN message schedule may be interrupted by inserting Gaps between the basic cycles of the system matrix. All nodes connected to the CAN network have to be configured for external event- synchronized time-triggered operation.

During a Gap, all transmissions are stopped and the CAN bus remains idle. A Gap is finished when the next reference message starts a new basic cycle. A Gap starts at the end of a basic cycle that itself was started by a reference message with bit Next_is_Gap = ‘1’ e.g. Gaps are initiated by the current time master.

The current time master has two options to initiate a Gap. A Gap can be initiated under software control when the application program writes TTOCN0.NIG = ‘1’. The Next_is_Gap bit will be transmitted as ‘1’ with the next reference message. A Gap can also be initiated under hardware control when the application program enables the event trigger input by writing TTOCN0.GCS = ‘1’. When a reference message is started and TTOCN0.GCS is set, a HIGH level at the event trigger will set Next_is_Gap = ‘1’.

As soon as that reference message is completed, the TTOST0.WFE bit will announce the Gap to the time master as well as to the time slaves. The current basic cycle will continue until its last time window. The time after the last time window is the Gap time.

For the actual time master and the potential time masters, TTOST0.GSI will be set when the last basic cycle has finished and the Gap time starts. In nodes that are time slaves, bit TTOST0.GSI will remain at ‘0’.

When a potential time master is in synchronization state In_Gap (TTOST0.SYS = “10”), it has four options to intentionally finish a Gap:

Under software control by writing TTOCN0.FGP = ‘1’.

Under hardware control (TTOCN0.GCS = ‘1’) an edge from HIGH to LOW at the event-trigger sets TTOCN0.FGP and restarts the schedule.

The third option is a time-triggered restart. When TTOCN0.TMG = ‘1’, the next register time mark interrupt (TTIR0.RTMI = ‘1’) will set TTOCN0.FGP and start the reference message.

Finally any potential time master will finish a Gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

Neither of these options can cause a basic cycle to be interrupted with a reference message.

Setting of TTOCN0.FGP after the Gap time has started will start the transmission of a reference message immediately and will thereby synchronize the message schedule. When TTOCN0.FGP is set before the Gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there will be no Gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = ‘1’ in the reference message will be ignored, as well as the event-trigger and the bits TTOCN0.NIG, TTOCN0.FGP, and TTOCN0.TMG.

40.3.3.4 Stop Watch

The stop watch function enables capturing of M_CAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first has to define local time, cycle time, or global time as stop watch source via TTOCN0.SWS. When TTOCN0.SWS is ≠ “00” and TT Interrupt Register flag TTIR0.SWE is ‘0’, the actual value of the time selected by TTOCN0.SWS will be copied into TTCPT.SWV on the next rising/falling edge (as configured via TTOCN0.SWP) on stop watch trigger. This will set interrupt flag TTIR0.SWE. After the application program has read TTCPT.SWV, it may enable the next stop watch event by resetting TTIR0.SWE to ‘0’.

40.3.3.5 Local Time, Cycle Time, Global Time, and External Clock Synchronization

There are two possible levels in time-triggered CAN: Level 1 and Level 2. Level 1 only provides time-triggered operation using cycle time. Level 2 additionally provides increased synchronisation quality, global time and external clock synchronisation. In both levels, all timing features are based on a local time base - the local time.

The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented 8 times each NTU. If the length of the NTU is shorter than 8 CAN clock periods (as may be configured in Level 1, or as a result of clock calibration in Level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only 4 times each NTU.

Figure 600 describes the synchronisation of the cycle time and global time, performed in the same manner by all TTCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message's frame synchronisation event. This frame synchronisation event occurs at the sample point of each Start of Frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the Cycle Sync Mark (Cycle Sync Mark = Sync_Mark - Ref_Mark) stored in register TTCSM. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (Cycle Time = Local Time - Ref_Mark).

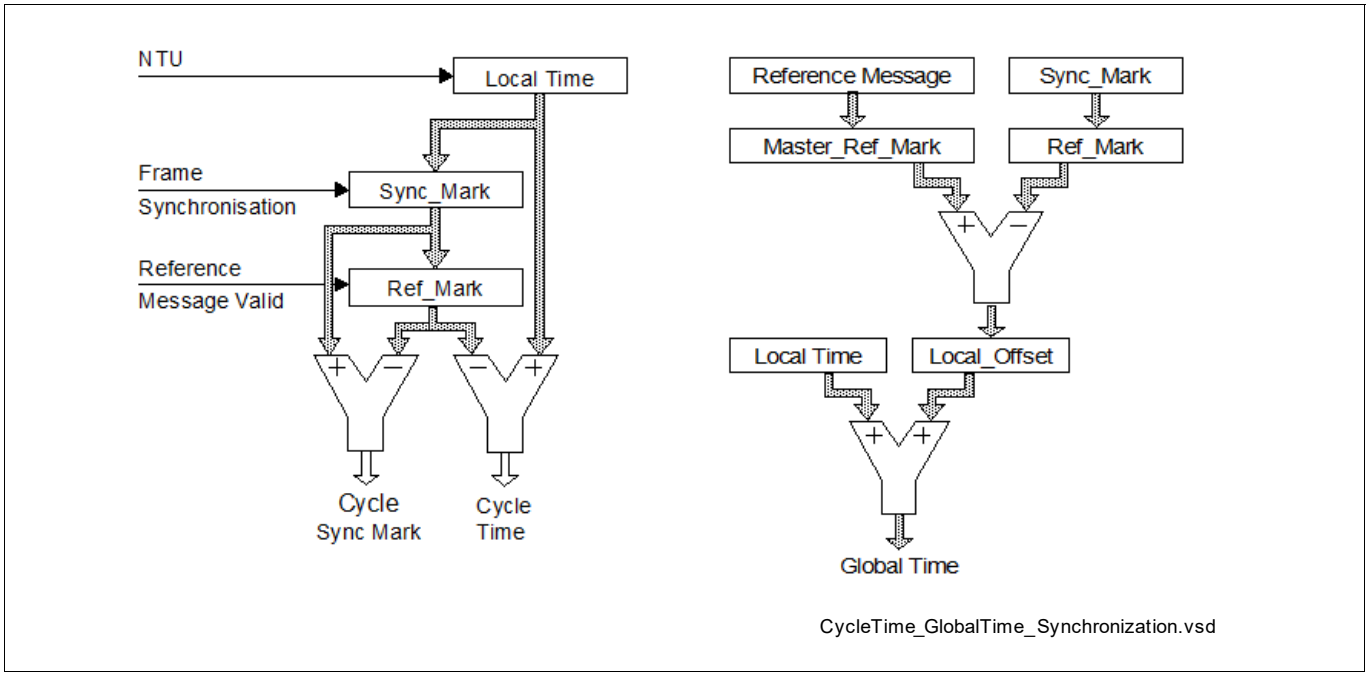


Figure 600 Cycle Time and Global Time Synchronization

The cycle time that can be read from TTCTC0.CT is the difference of the node's local time and Ref_Mark, both synchronized into the Host clock domain and truncated to 16 bits.

The global time exists for TTCAN Level 0 and Level 2 only, in Level 1 it is invalid. The node's view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master will use its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3, 4). The global time that can be read from TTLGT0.GT is the sum of the node's local time and its local offset, both synchronized into the Host clock domain and truncated to 16 bits. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see **Figure 600**). The node's view of the global time is local time + local offset. In a potential time master that has never received another time master's reference message, Local_Offset will be zero. When a node becomes the current time master after first having received other reference messages, Local_Offset will be frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signalled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

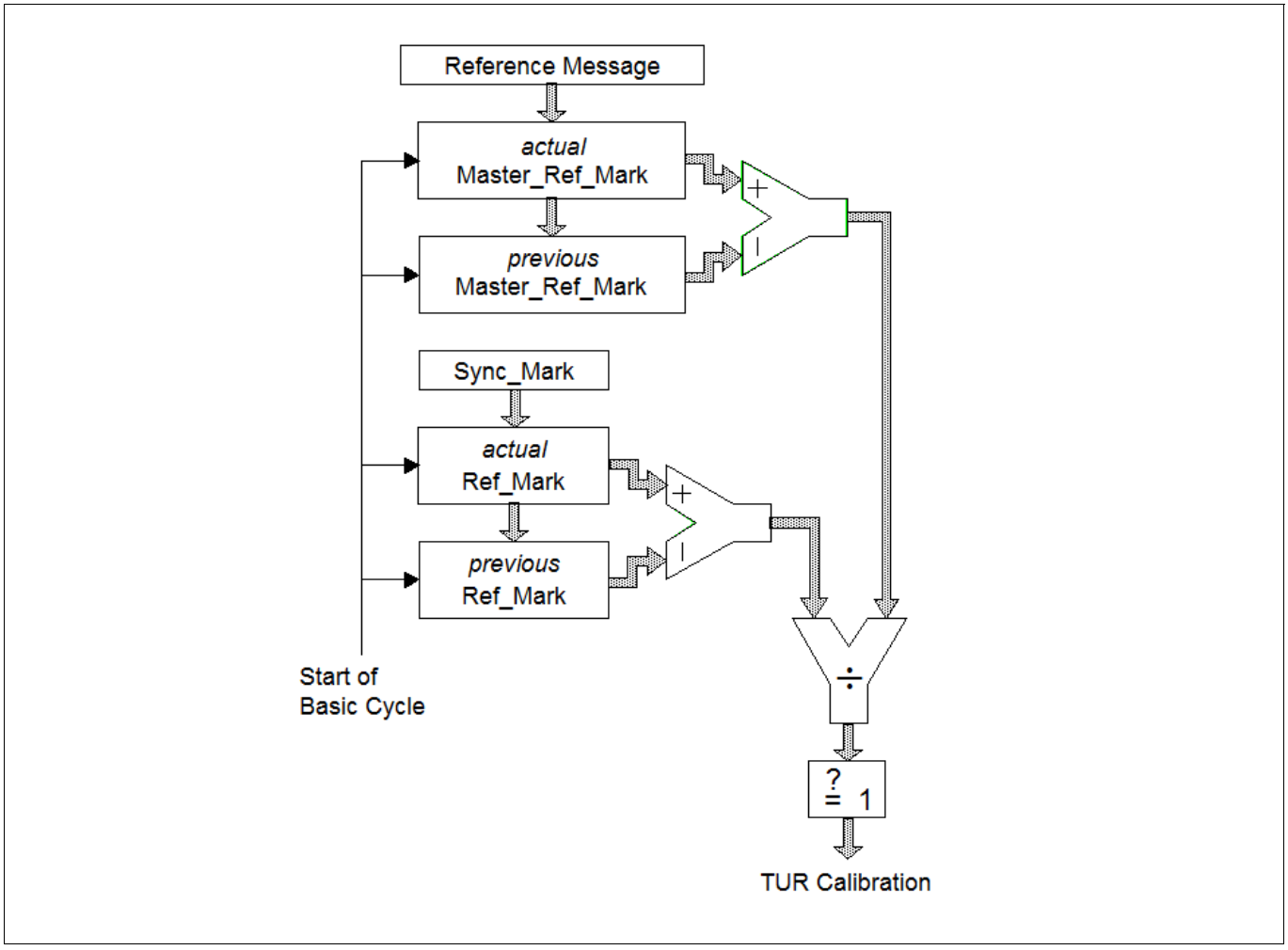


Figure 601 TTCAN Level 0 and Level 2 Drift Compensation

Figure 601 describes how in TTCAN Level 0,2 each time receiving node compensates the drift between its own local clock and the time master's clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for TURNA0.NAV is calculated. If the Synchronisation Deviation SD = |NC - TURNA0.NAV| ≤ SDL (Synchronisation Deviation Limit), the new value for TURNA0.NAV takes effect. Else the automatic drift compensation is suspended.

In TTCAN Level 0 and Level 2, TTOST0.QCS indicates whether the automatic drift compensation is active or suspended. In TTCAN Level 1, TTOST0.QCS is always '1'.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit TTOCF0.EECS.

The stop watch function (see **Section 40.3.3.4**) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated Numerator Configuration Low to TURCF0.NCL (TURCF0.DC cannot be updated during operation). The new value takes effect by writing TTOCN0.ECS to '1'.

The global time phase is adjusted by first writing the phase offset into the TT Global Time Preset register TTGTP0. The new value takes effect by writing TTOCN0.SGT to '1'. The first reference message transmitted after the global time phase adjustment will have the Disc_Bit set to '1'.

TTOST0.QGTP shows whether the node's global time is in phase with the time master's global time. TTOST0.QGTP is permanently '0' in TTCAN Level 1 and when the Synchronisation Deviation Limit is exceeded in TTCAN Level 0,2 (TTOST0.QCS = '0'). It is temporarily '0' while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = '1' or when TTOST0.QCS='0'.

40.3.3.6 TTCAN Error Level

The ISO 11898-4 specifies four levels of error severity:

S0 - No Error

S1 - Warning
Only notification of application, reaction application-specific.

S2 Error
Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the Reference Trigger Offset TTOST0.RTO set to the maximum value of 127.

S3 - Severe Error
Notification of application. All CAN bus operations are stopped, i.e. transmission of dominant bits is not allowed, and CCCRi.MON is set. The S3 error condition remains active until the application updates the configuration (set CCCRi.CCE).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by TTIR0.ELC. The error level is monitored by TTOST0.EL.

The M_CAN signals the following error conditions as required by ISO 11898-4:

Config_Error (S3)

Sets Error Level TTOST0.EL to "11" when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.

Watch_Trigger_Reached (S3)

Sets Error Level TTOST0.EL to "11" when a watch trigger was reached because the reference message is missing.

Application_Watchdog (S3)

Sets Error Level TTOST0.EL to "11" when the application failed to serve the application watchdog. The application watchdog is configured via TTOCF0.AWL. It is served by reading register TTOST0. When the watchdog is not served in time, bit TTOST0.AWE and interrupt flag TTIR0.AW are set, all TTCAN communication is stopped, and the M_CAN is set into bus monitoring mode (CCCRi.MON set to '1').

CAN_Bus_Off (S3)

Entering CAN_Bus_Off state sets error level TTOST0.EL to "11". CAN_Bus_Off state is signalled by PSRi.BO = '1' and CCCRi.INIT = '1'.

Scheduling_Error_2 (S2)

Sets Error Level TTOST0.EL to “10” if the MSC of one Tx_Trigger has reached 7. In addition interrupt flag TTIR0.SE2 is set. The Error Level TTOST0.EL is reset to “00” at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.

Tx_Overflow (S2)

Sets Error Level TTOST0.EL to “10” when the Tx count is equal or higher than the expected number of Tx_Triggers TTMLM.ENTT and a Tx_Trigger event occurs. In addition interrupt flag TTIR0.TXO is set. The Error Level TTOST0.EL is reset to “00” when the Tx count is no more than TTMLM.ENTT at the start of a new matrix cycle.

Scheduling_Error_1 (S1)

Sets Error Level TTOST0.EL to “01” if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than 2, or if one of the MSCs of an exclusive Rx_Trigger has reached 7. In addition interrupt flag TTIR0.SE1 is set. If within one matrix cycle none of these conditions is valid, the Error Level TTOST0.EL is reset to “00”.

Tx_Underflow (S1)

Sets Error Level TTOST0.EL to “01” when the Tx count is less than the expected number of Tx_Triggers TTMLM.ENTT at the start of a new matrix cycle. In addition interrupt flag TTIR0.TXU is set. The Error Level TTOST0.EL is reset to “00” when the Tx count is at least TTMLM.ENTT at the start of a new matrix cycle.

40.3.3.7 TTCAN Message Handling

40.3.3.7.1 Reference Message

For potential time masters the identifier of the reference message is configured via TTRMC.RID. No dedicated Tx Buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte (TTCAN Level 1) resp. the first four data bytes (TTCAN Level 0 and Level 2) will be provided by the FSE.

In case the reference message Payload Select TTRMC.RMPS is set, the rest of the reference message’s payload (Level 1: bytes 2-8, Level 0,2: bytes 5-6) is taken from Tx Buffer 0. In this case the data length DLC code from message buffer 0 is used.

Table 377 Number of Data Bytes transmitted with a reference messages

TTRMC.RMPS	TXBRPi.TRP0	Level 0	Level 1	Level 2
0	0	4	1	4
0	1	4	1	4
1	0	4	1	4
1	1	4 + MB0	1 + MB0	4 + MB0

To send additional payload with the reference message in Level 1 a DLC > 1 has to be configured, for Level 0,2 a DLC > 4 is required. In addition the transmission request pending bit TXBRPi.TRP0 of message buffer 0 must be set (see **Table 377**). In case bit TXBRPi.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the Reference Identifier TTRMC.RID is used.

40.3.3.7.2 Message Reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see **Chapter 40.3.2.4**).

The Message Status Count MSC is part of the corresponding trigger memory element and has to be initialized to zero during configuration. It is updated while the M_CAN is in synchronization states In_Gap or In_Schedule. The update happens at the message's Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger should be set to a value where it is ensured that reception and acceptance filtering for the targeted message has completed. This has to take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

40.3.3.7.3 Message Transmission

For time-triggered message transmission the M_CAN supplies 32 dedicated Tx buffers (see **Chapter 40.3.2.5.2**). A Tx FIFO or Tx queue is not available when the M_CAN is configured for time-triggered operation (TTOCF0.OM = “01” or “10”).

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program has to update the data regularly and on time, synchronized to the cycle time. The Host CPU is responsible that no partially updated messages are transmitted. To assure this the Host has to proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has completed by reading TXBTO
- Update the Tx buffer’s configuration and/or payload
- Issue an "Add Request" to set the Tx Buffer Request Pending bit

Tx_Trigger_Continous

- Issue a Cancellation Request to reset the Tx Buffer Request Pending bit
- Check whether the cancellation has finished by reading TXBCF
- Update Tx buffer’s configuration and/or payload
- Issue an "Add Request" to set the Tx Buffer Request Pending bit

The message’s MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission could not be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and could not be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message could have been started within its transmit enable window but was not started because transmission was disabled (M_CAN in Error Level S2 or Host has disabled this particular message).

The Tx buffers may be managed dynamically, i.e. several messages with different identifiers may share the same Tx buffer element. In this case the Host has to assure that no transmission request is pending for the Tx buffer element to be reconfigured by checking TXBRPi.

If a Tx buffer with pending transmission request should be updated, the Host first has to issue a cancellation request and check whether the cancellation has completed by reading TXBCF before it starts updating.

The Tx Handler will transfer a message from the Message RAM to its intermediate output buffer at the trigger element which becomes active immediately before the Tx_Trigger element which defines the beginning of the transmit window. During and after the transfer time the transmit message may not be updated and its TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be e.g. a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger has to be large enough to guarantee that the Tx Handler can read four words from the Message RAM even at high RAM access load from other modules.

Transmission in Exclusive Time Windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx Buffer Request Pending bit is reset. After successful transmission started by a Tx_Trigger_Continous the respective Tx Buffer Request Pending remains set. When the transmission was not successful due to disturbances, it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in Arbitrating Time Windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message has to arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it will be repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in Merged Arbitrating Time Windows

The purpose of a merged arbitrating time window is to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a single node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission will not be started if the corresponding Transmission Request Pending flag was reset by a successful Tx cancellation.

In single transmit windows, the Tx Handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmissions will be attempted in the sequence defined by the trigger list. If the time mark of a fourth message is read before the first is transmitted (or cancelled by the Host), the fourth request will be ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window will be started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if there is competing bus traffic. This has to be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx Buffer needs to be updated.

40.3.3.8 TTCAN Interrupt and Error Handling

The TT Interrupt Register TTIR consists of four segments. Each interrupt can be enabled separately by the corresponding bit in the TT Interrupt Enable register TTIE. The flags remain set until the Host clears them. A flag is cleared by writing a “1” to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a severe error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the M_CAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, e.g. by disturbances on the CAN bus, they will be handled by the TTCAN protocol’s failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (Level 0,2 only) that require a reaction by the application program. With a Stop Watch Event triggered by a rising/falling edge the internal time values are captured. The Trigger Time Mark Interrupt notifies the application that a specific Time_Base_Trigger is reached. The Register Time Mark Interrupt signals that the time referenced by TTOCN0.TMC (Cycle, Local, or Global) equals time mark TTTMK.TM. It can also be used to finish a Gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

40.3.3.9 Level 0

TTCAN Level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN Level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via TTOCF0.OM = “11”. In this mode the M_CAN operates in event-driven CAN communication, there is no fixed schedule, the configuration of TTOCF0.GEN is ignored. External event-synchronized operation is not available in Level 0. A synchronized time base is maintained by transmission of reference messages.

In Level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (TTIR0.TTMI) is set when the cycle time has reached TTOCF0.IRTO * 0x200, it reminds the Host to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (TTIR0.WT) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (TTIR0.RTMI) are also possible.

The reference message is configured as for Level 2 operation. Received reference messages are recognized by the identifier configured in register TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the Host sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- TTRMC
- TTOCF except EVTP, AWL, GEN
- TTMLM except ENTT, TXEW
- TURCF

Level 0 operation is controlled via:

- TTOCN except NIG, TMG, FGP, GCS, TTMIE
- TTGTP
- TTTMK
- TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- TTIR the following bits have changed function
  - TTMI not defined by trigger memory - activated at cycle time TTOCF0.IRTO * 0x200
  - WT not defined by trigger memory - activated at cycle time 0xFF00

Level 0 operation is signalled via:

- TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)

40.3.3.9.1 Synchronizing

Figure 602 below describes the states and state transitions in TTCAN Level 0 operation. Level 0 has no In_Gap state.

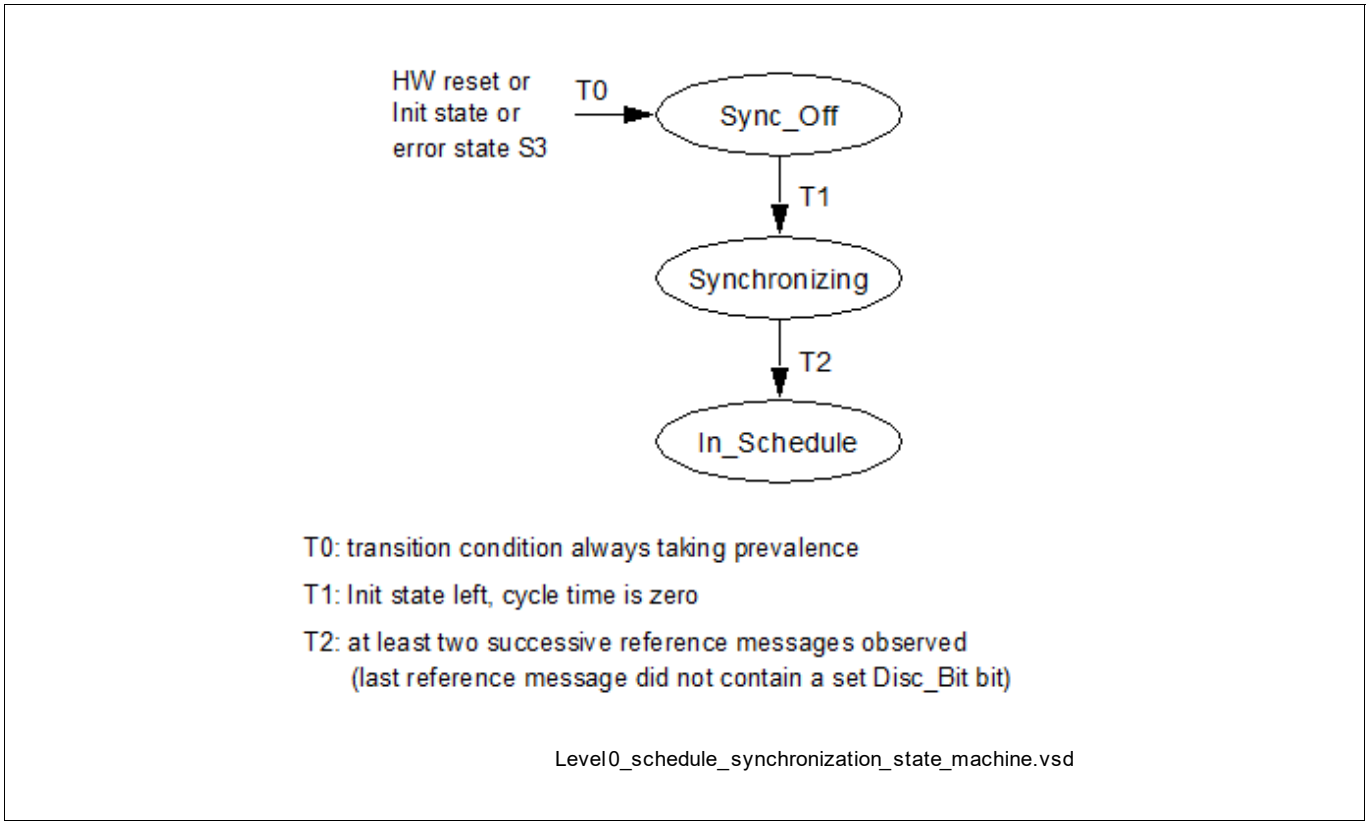


Figure 602 Level 0 schedule synchronization state machine

40.3.3.9.2 Handling of Error Levels

During Level 0 operation only the following error conditions may occur:

- Watch_Trigger_Reached (S3), reached cycle time $FF00_H$
- CAN_Bus_Off (S3)

Since no S1 and S2 error are possible, the error level can only switch between S0 (No Error) and S3 (Severe Error). In TTCAN Level 0 an S3 error is handled differently. When error level S3 is reached, both TTOST0.SYS and TTOST0.MS are reset, and interrupt flags TTIR0.GTE and TTIR0.GTD are set.

When error level S3 (TTOST0.EL = “11”) is entered, bus monitoring mode is, contrary to TTCAN Level 1 and Level 2, not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

40.3.3.9.3 Master Slave Relation

Figure 603 below describes the master slave relation in TTCAN Level 0. In case of an S3 error the M_CAN returns to state Master_Off.

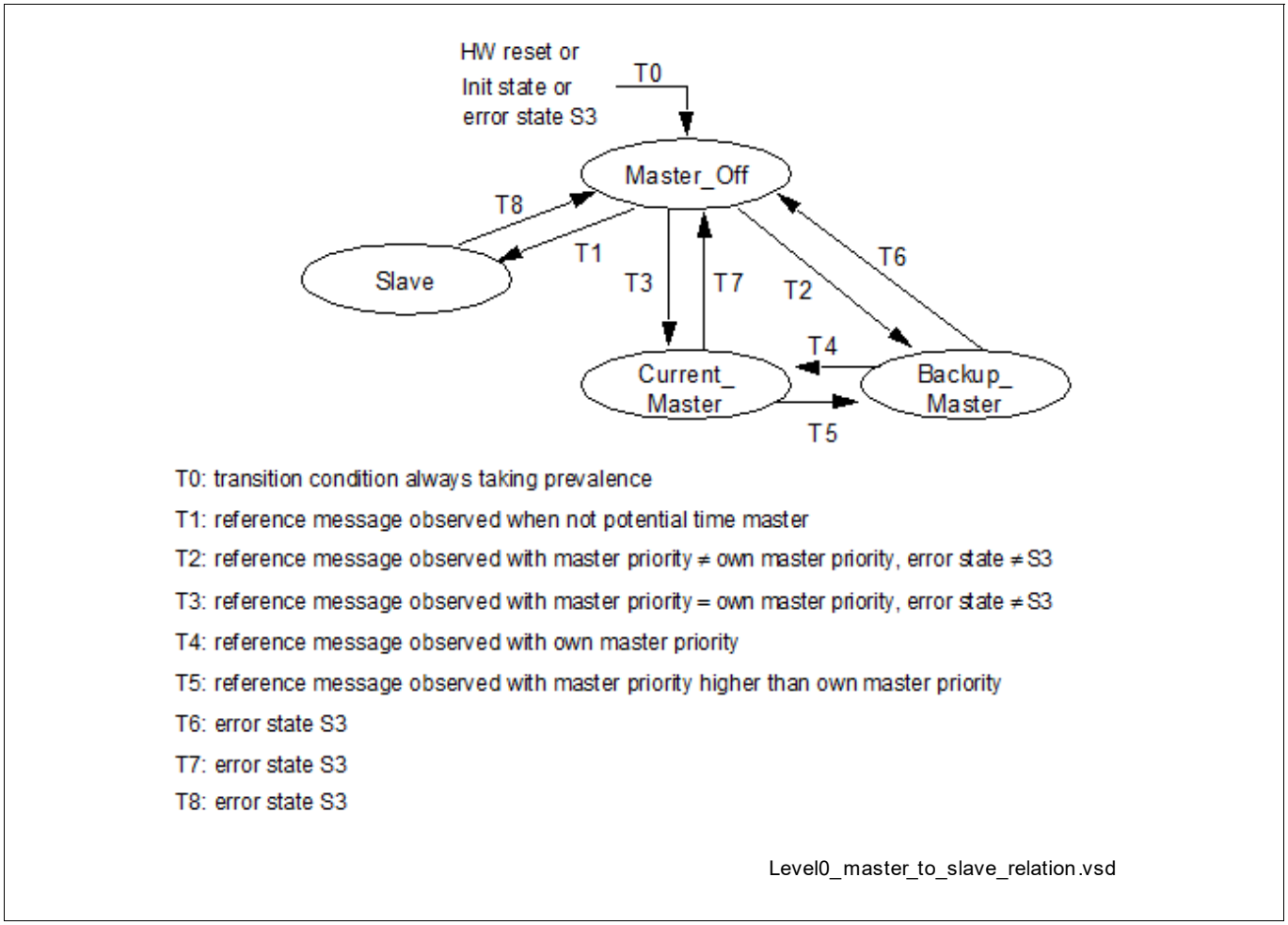


Figure 603 Level 0 master to slave relation

40.3.3.10 Synchronization to external Time Schedule

This feature can be used to synchronize the phase of the M_CAN's schedule to an external schedule (e.g. that of a second TTCAN network or FlexRay network). It is applicable only when the M_CAN is current time master (TTOST0.MS = "11").

External synchronization is controlled by event trigger input. If bit TTOCN0.ESCN is set, a rising edge at the event trigger the M_CAN compares its actual cycle time with the target phase value configured by TTGTP0.CTP.

Before setting TTOCN0.ESCN the Host has to adapt the phases of the two time schedules e.g. by using the TTCAN gap control (see **Section 40.3.3.3**). When the Host sets TTOCN0.ESCN, TTOST0.SPL is set.

If the difference between the cycle time and theTTIR0.target phase value TTGTP0.CTP at the rising edge at the event trigger is greater than 9 NTU, the phase lock bit TTOST0.SPL is reset, and interrupt flag TTIR0.CSM is set. TTOST0.SPL is also reset (and TTIR0.CSM is set), when another node becomes time master.

If both TTOST0.SPL and TTOCN0.ESCN are set, and if the difference between the cycle time and the target phase value TTGTP0.CTP at the rising edge at the event trigger is less or equal 9 NTU, the phase lock bit TTOST0.SPL remains set, and the measured difference is used as reference trigger offset value to adjust the phase at the next transmitted reference message.

Note: The rising edge detection at the event trigger is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with TTGTP0.CTP. All further edges until the beginning of the next basic cycle are ignored.

40.4 Registers

This section describes the registers within the MCMCAN module. All the registers are prefixed as CAN0_, CAN1_ or CAN2_ for MCMCAN modules, if the corresponding module exist for a particular product variant (Refer Appendix). The registers are grouped as three different sections:

- General Configuration Registers
- User Interface Registers
- Registers within M_CAN

The registers listed in **Figure 604** are not included in the MCMCAN module kernel (part of general module IP blocks), some registers must be programmed for proper operation of the MCMCAN module.

The additional ACCEN registers, are MCMCAN specific.

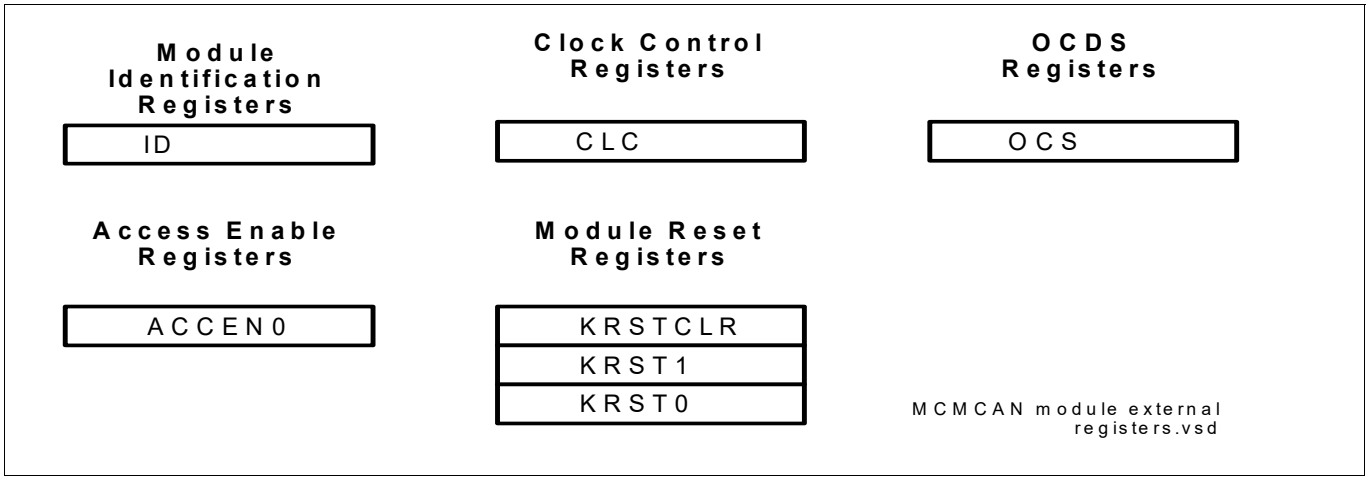


Figure 604 CAN Implementation-specific Special Function Registers

40.4.1 MCMCAN RAM address space

Table 378 Register Address Space - MCMCAN RAM

Module	Base Address	End Address	Note
CANRAM0	F020 0000$_H$	F020 7FFF$_H$	RAM Area CAN0
CANRAM1	F021 0000$_H$	F021 3FFF$_H$	RAM Area CAN1
CANRAM2	F022 0000$_H$	F022 3FFF$_H$	RAM Area CAN2

Note: Refer Appendix of a product variant for the register address space of MCMCAN and available RAM modules.

40.4.2 MCMCAN register overview

Table 379 Register Overview - CAN (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
RAM	Embedded SRAM for messages (008000_H Byte)	000000_H				
CLC	CAN Clock Control Register	008000_H	U,SV	SV,E,P	Application Reset	83
ID	Module Identification Register	008008_H	U,SV	nBE	Application Reset	70
MCR	Module Control Register	008030_H	U,SV	SV,U,P	Application Reset	70
BUFADR	Buffer receive address and transmit address	008034_H	U,SV	SV,U,P	Application Reset	73
MECR	Measure Control Register	008040_H	U,SV	SV,U,P	Application Reset	74
MESTAT	Measure Status Register	008044_H	U,SV	SV,U,P	Application Reset	75
ACCENCTR0	Access Enable Register Control 0	$0080DC_H$	U,SV	SV,SE,P	Application Reset	79
OCS	OCDS Control and Status	$0080E8_H$	U,SV	SV,P,OEN	Debug Reset	76
KRSTCLR	Kernel Reset Status Clear Register	$0080EC_H$	U,SV	SV,E,P	Application Reset	81
KRST1	Kernel Reset Register 1	$0080F0_H$	U,SV	SV,E,P	Application Reset	81
KRST0	Kernel Reset Register 0	$0080F4_H$	U,SV	SV,E,P	Application Reset	80
ACCEN0	Access Enable Register 0	$0080FC_H$	U,SV	SV,SE	Application Reset	78
ACCENNODEi0	Access Enable Register CAN Node i 0	$008100_H + i*400_H$	U,SV	SV,SE,P	Application Reset	79
STARTADRi	Start Address Node i	$008108_H + i*400_H$	U,SV	SV,SE,P	Application Reset	89
ENDADRi	End Address Node i	$00810C_H + i*400_H$	U,SV	SV,SE,P	Application Reset	89
ISREGi	Interrupt Signalling Register i	$008110_H + i*400_H$	U,SV	nBE	Application Reset	86
GRINT1i	Interrupt routing for Groups 1 i	$008114_H + i*400_H$	U,SV	SV,U,P	Application Reset	83
GRINT2i	Interrupt routing for Groups 2 i	$008118_H + i*400_H$	U,SV	SV,U,P	Application Reset	85

Table 379 Register Overview - CAN (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
NTCCRi	Node i Timer Clock Control Register	008120_H +i*400_H	U,SV	SV,U,P	Application Reset	90
NTATTRi	Node i Timer A Transmit Trigger Register	008124_H +i*400_H	U,SV	SV,U,P	Application Reset	91
NTBTTRi	Node i Timer B Transmit Trigger Register	008128_H +i*400_H	U,SV	SV,U,P	Application Reset	92
NTCTTRi	Node i Timer C Transmit Trigger Register	$00812C_H$ +i*400_H	U,SV	SV,U,P	Application Reset	92
NTRTRi	Node i Timer Receive Timeout Register	008130_H +i*400_H	U,SV	SV,U,P	Application Reset	93
NPCRi	Node i Port Control Register	008140_H +i*400_H	U,SV	SV,U,P	Application Reset	87
TTCRi	Time Trigger Control Register	$0081F0_H$	U,SV	SV,U,P	Application Reset	88
CRELi	Core Release Register i	008200_H +i*400_H	U,SV	nBE	See page 95	95
ENDNi	Endian Register i	008204_H +i*400_H	U,SV	nBE	Application Reset	95
DBTPi	Data Bit Timing & Prescaler Register i	$00820C_H$ +i*400_H	U,SV	SV,U,P	Application Reset	96
TESTi	Test Register i	008210_H +i*400_H	U,SV	SV,U,P	Application Reset	97
RWDi	RAM Watchdog i	008214_H +i*400_H	U,SV	SV,U,P	Application Reset	98
CCCRi	CC Control Register i	008218_H +i*400_H	U,SV	SV,U,P	Application Reset	99
NBTPi	Nominal Bit Timing & Prescaler Register i	$00821C_H$ +i*400_H	U,SV	SV,U,P	Application Reset	101
TSCCi	Timestamp Counter Configuration i	008220_H +i*400_H	U,SV	SV,U,P	Application Reset	103
TSCVi	Timestamp Counter Value i	008224_H +i*400_H	U,SV	SV,U,P	Application Reset	104
TOCCi	Timeout Counter Configuration i	008228_H +i*400_H	U,SV	SV,U,P	Application Reset	104
TOCVi	Timeout Counter Value i	$00822C_H$ +i*400_H	U,SV	SV,U,P	Application Reset	105
ECRi	Error Counter Register i	008240_H +i*400_H	U,SV	nBE	Application Reset	106
PSRi	Protocol Status Register i	008244_H +i*400_H	U,SV	nBE	Application Reset	107

Table 379 Register Overview - CAN (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
TDCRi	Transmitter Delay Compensation Register i	008248_H +i*400_H	U,SV	SV,U,P	Application Reset	111
IRi	Interrupt Register i	008250_H +i*400_H	U,SV	SV,U,P	Application Reset	112
IEi	Interrupt Enable i	008254_H +i*400_H	U,SV	SV,U,P	Application Reset	115
GFCi	Global Filter Configuration i	008280_H +i*400_H	U,SV	SV,U,P	Application Reset	117
SIDFCi	Standard ID Filter Configuration i	008284_H +i*400_H	U,SV	SV,U,P	Application Reset	118
XIDFCi	Extended ID Filter Configuration i	008288_H +i*400_H	U,SV	SV,U,P	Application Reset	119
XIDAMi	Extended ID AND Mask i	008290_H +i*400_H	U,SV	SV,U,P	Application Reset	120
HPMSi	High Priority Message Status i	008294_H +i*400_H	U,SV	nBE	Application Reset	120
NDAT1i	New Data 1 i	008298_H +i*400_H	U,SV	SV,U,P	Application Reset	121
NDAT2i	New Data 2 i	$00829C_H$ +i*400_H	U,SV	SV,U,P	Application Reset	122
RXF0Ci	Rx FIFO 0 Configuration i	$0082A0_H$ +i*400_H	U,SV	SV,U,P	Application Reset	122
RXF0Si	Rx FIFO 0 Status i	$0082A4_H$ +i*400_H	U,SV	nBE	Application Reset	123
RXF0Ai	Rx FIFO 0 Acknowledge i	$0082A8_H$ +i*400_H	U,SV	SV,U,P	Application Reset	124
RXBCi	Rx Buffer Configuration i	$0082AC_H$ +i*400_H	U,SV	SV,U,P	Application Reset	125
RXF1Ci	Rx FIFO 1 Configuration i	$0082B0_H$ +i*400_H	U,SV	SV,U,P	Application Reset	125
RXF1Si	Rx FIFO 1 Status i	$0082B4_H$ +i*400_H	U,SV	nBE	Application Reset	126
RXF1Ai	Rx FIFO 1 Acknowledge i	$0082B8_H$ +i*400_H	U,SV	SV,U,P	Application Reset	127
RXESCi	Rx Buffer/FIFO Element Size Configuration i	$0082BC_H$ +i*400_H	U,SV	SV,U,P	Application Reset	127
TXBCi	Tx Buffer Configuration i	$0082C0_H$ +i*400_H	U,SV	SV,U,P	Application Reset	129
TXFQSi	Tx FIFO/Queue Status i	$0082C4_H$ +i*400_H	U,SV	nBE	Application Reset	130

Table 379 Register Overview - CAN (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
TXESCi	Tx Buffer Element Size Configuration i	$0082C8_H$ +i*400_H	U,SV	SV,U,P	Application Reset	131
TXBRPi	Tx Buffer Request Pending i	$0082CC_H$ +i*400_H	U,SV	SV,U,P	Application Reset	132
TXBARi	Tx Buffer Add Request i	$0082D0_H$ +i*400_H	U,SV	SV,U,P	Application Reset	132
TXBCRi	Tx Buffer Cancellation Request i	$0082D4_H$ +i*400_H	U,SV	SV,U,P	Application Reset	133
TXBTOi	Tx Buffer Transmission Occurred i	$0082D8_H$ +i*400_H	U,SV	nBE	Application Reset	134
TXBCFi	Tx Buffer Cancellation Finished i	$0082DC_H$ +i*400_H	U,SV	nBE	Application Reset	134
TXBTIEi	Tx Buffer Transmission Interrupt Enable i	$0082E0_H$ +i*400_H	U,SV	SV,U,P	Application Reset	135
TXBCIEi	Tx Buffer Cancellation Finished Interrupt Enable i	$0082E4_H$ +i*400_H	U,SV	SV,U,P	Application Reset	135
TXEFCi	Tx Event FIFO Configuration i	$0082F0_H$ +i*400_H	U,SV	SV,U,P	Application Reset	136
TXEFSi	Tx Event FIFO Status i	$0082F4_H$ +i*400_H	U,SV	nBE	Application Reset	137
TXEFAi	Tx Event FIFO Acknowledge i	$0082F8_H$ +i*400_H	U,SV	SV,U,P	Application Reset	137
TTTMCi	TT Trigger Memory Configuration i	008300_H	U,SV	SV,U,P	Application Reset	138
TTRMCi	TT Reference Message Configuration i	008304_H	U,SV	SV,U,P	Application Reset	138
TTOCFi	TT Operation Configuration i	008308_H	U,SV	SV,U,P	Application Reset	139
TTMLMi	TT Matrix Limits i	$00830C_H$	U,SV	SV,U,P	Application Reset	141
TURCFi	TUR Configuration i	008310_H	U,SV	SV,U,P	Application Reset	142
TTOCNi	TT Operation Control i	008314_H	U,SV	SV,U,P	Application Reset	144
TTGTPi	TT Global Time Preset i	008318_H	U,SV	SV,U,P	Application Reset	146
TTTMKi	TT Time Mark i	$00831C_H$	U,SV	SV,U,P	Application Reset	147
TTIRi	TT Interrupt Register i	008320_H	U,SV	SV,U,P	Application Reset	148

Table 379 Register Overview - CAN (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
TTIEi	TT Interrupt Enable i	008324_H	U,SV	SV,U,P	Application Reset	151
TTOSTi	TT Operation Status i	$00832C_H$	U,SV	nBE	Application Reset	153
TURNAi	TUR Numerator Actual i	008330_H	U,SV	nBE	Application Reset	155
TTLGTi	TT Local & Global Time i	008334_H	U,SV	nBE	Application Reset	156
TTCTCi	TT Cycle Time & Count i	008338_H	U,SV	nBE	Application Reset	156
TTCPTi	TT Capture Time i	$00833C_H$	U,SV	nBE	Application Reset	157
TTCSMi	TT Cycle Sync Mark i	008340_H	U,SV	nBE	Application Reset	157

40.4.3 General Configuration Registers

This section describes the global module registers, system registers, access enable registers and kernal reset registers.

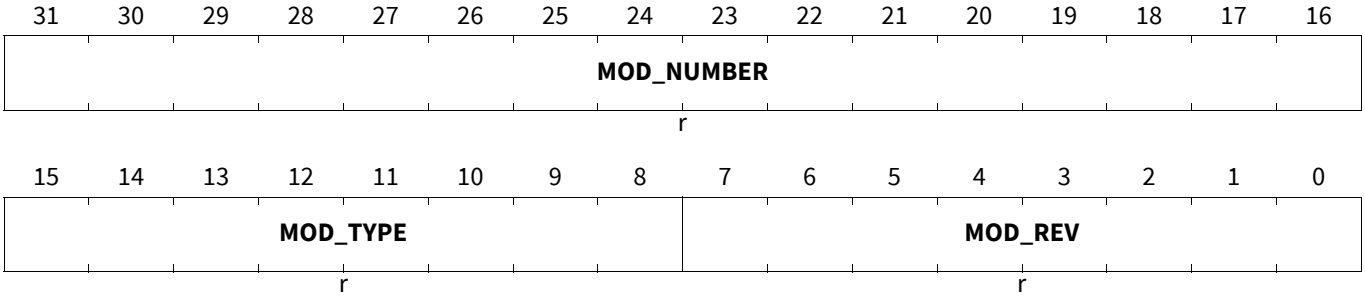
40.4.3.1 Global Module Registers

Global Module Registers, are registers, which are not part of M_CAN nodes.

Module Identification Register

The Module Identification Register ID contains read-only information about the module version.

ID
Module Identification Register **(008008_H)** **Application Reset Value: $00B8\ C0XX_H$**



Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision Number** MOD_REV defines the revision number. The value of a module revision starts with 01_H (first revision).
MOD_TYPE	15:8	r	**Module Type** $C0_H$ Define the module as a 32-bit module.
MOD_NUMBER	31:16	r	**Module Number Value** This bit field defines the MCMCAN module identification number $00B8_H$.

Module Control Register

The Module Control Register MCR contains basic settings that determine the operation of the MCMCAN module.

The write access to the lowest byte of the MCR register becomes only valid, if and only if, MCR.CCCE and MCR.CI are already set during write access. To switch the clocks on or off, the bits of MCR.CCCE and MCR.CI have to be reset afterwards. Before this sequence hasn't taken place, no write access to the corresponding nodes, can be done.

Note: If the baud rate logic is supplied from an unstable clock source, or no clock at all, the CAN functionality is not guaranteed.

To be able to change the clock settings the following programming sequence needs to be met:

```
uwTemp = CANn_MCR.U;
uwTemp |= (0xC0000000 | CLKSELx);
CANn_MCR.U = uwTemp;
uwTemp &= ~0xC0000000;
```

CANn_MCR.U = uwTemp;
The clock settings for CAN nodes becomes active.
To be able to start the RAM initialization, the following programming sequence need to be met:
CANn_MCR |= 0xC0000000;
Wait until CANn_MCR.RBUSY is 0b
Set CANn_MCR.RINIT to 0b
Set CANn_MCR.RINIT to 1b
Dummy read CANn_MCR
Wait until CANn_MCR.RBUSY is 0b
Set CANn_MCR.RINIT to 0b
CANn_MCR &= ~0xC0000000;
RAM initialization is finished

MCR
Module Control Register **(008030_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCCE	**CI**	**RINIT**	**RBUSY**	**DXCM**	**NODE**			**0**							
rw	rw	rw	rh	rw		rw		r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								**CLKSEL3**		**CLKSEL2**		**CLKSEL1**		**CLKSEL0**	
r								rw		rw		rw		rw	

Field	Bits	Type	Description
CLKSEL0	1:0	rw	**Clock Select 0** This bitfield is MCR.CI and MCR.CCCE protected. 00_B No clock supplied 01_B The asynchronous clock source is switched on 10_B The synchronous clock source is switched on 11_B Both clock sources are switched on
CLKSEL1	3:2	rw	**Clock Select 1** This bitfield is MCR.CI and MCR.CCCE protected. 00_B No clock supplied 01_B The asynchronous clock source is switched on 10_B The synchronous clock source is switched on 11_B Both clock sources are switched on
CLKSEL2	5:4	rw	**Clock Select 2** This bitfield is MCR.CI and MCR.CCCE protected. 00_B No clock supplied 01_B The asynchronous clock source is switched on 10_B The synchronous clock source is switched on 11_B Both clock sources are switched on

Field	Bits	Type	Description
CLKSEL3	7:6	rw	**Clock Select 3** This bitfield is MCR.CI and MCR.CCCE protected. 00_B No clock supplied 01_B The asynchronous clock source is switched on 10_B The synchronous clock source is switched on 11_B Both clock sources are switched on
NODE	26:24	rw	**Node** This bit field determines the CAN node i which is used for debug over CAN. This bitfield only exists on CAN0. 000_B Node 0 … 011_B Node 3
DXCM	27	rw	**Debug Over CAN Messages Enable** This bit enables the debug over serial connections between DAP and CAN0 module. If enabled the lowest receive/transmit message buffer is reserved for debugger communication. DXCM is described in detail in the OCDS chapter. This bit only exists on CAN0. 0_B DXCM disabled 1_B DXCM enabled
RBUSY	28	rh	**RAM BUSY** This bit shows that the RAM Initialization is running. This bit is set back to 0b by hardware when the RAM intialization is completed.
RINIT	29	rw	**RAM Init** This bit is MCR.CI and MCR.CCCE protected. This bit starts the initialization of the RAM block to all 0x0. The RAM initialization is started only when this bit is changed from 0b to 1b and also RBUSY is 0b.
CI	30	rw	**Change Init** Needs to be set to enable and disable clocks. 0_B Change Init disabled 1_B Change Init enabled (takes effect with CCCE:=1)
CCCE	31	rw	**Clock and RAM Change Enable** Needs to be set to enable and disable the clocks. 0_B Clock and RAM Change disabled 1_B Clock and RAM Change enabled (takes effect with CI:=1)
0	23:8	r	**Reserved** Shall read 0; shall be written with 0.

Debug over CAN (DXCM feature)

The MCMCAN controller supports debugging using standard CAN tool access in parallel to regular CAN bus traffic. This is achieved by transmitting DAP telegrams and replies as regular CAN messages (DXCM DAP over CAN Messages). DXCM uses the lowest message buffers and it is strongly recommended to use also the same CAN pins as for DXCPL (DAP over CAN Physical Layer). DXCM is enabled with the MCR.DXCM bit. Please refer to the OCDS chapter for more information about DAP, DXCM and DXCPL.

Debug over CAN shall be only available on CAN0. TX Buffer 0 will be the sending transmit object. For receive at least one message buffer has to be configured. Meaning that RX Buffer 0 will be used for receiving DAP telegrams. The starting address of the message buffer and the receiving address of the message buffer have to be configured within BUFADR register.

Assigning the buffer start address

The following register assigns the start address to all features needing the message buffers inside the corresponding M_CAN, which are for receive and transmit.

Buffer receive address and transmit address

BUFADR

Buffer receive address and transmit address (008034$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		RXBUF													
r		rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		TXBUF													
r		rw													

Field	Bits	Type	Description
TXBUF	13:0	rw	**Transmit Buffer start address** This is the start address of the first dedicated transmit buffer.
RXBUF	29:16	rw	**Receive Buffer start address** This is the start address of the first dedicated receive buffer.
0	15:14, 31:30	r	**Reserved** Shall read 0; shall be written with 0.

Pretended Networking

The registers above are intended to support Pretended Networking. As an application example, the SPB bus is clocked at 40MHz. The asynchronous module part is clocked either with 40MHz as well or even more power saving with direct drive from the oscillator. The cores are in idle mode.

Messages can be received and a receive interrupt can be generated. It is possible to trigger messages with or without changing the content by the timers provided. For example the network management message and two related messages can be triggered without any CPU interaction.

As mostly the operating system is still running, messages can be changed without any problem during the time, where the operating system is active.

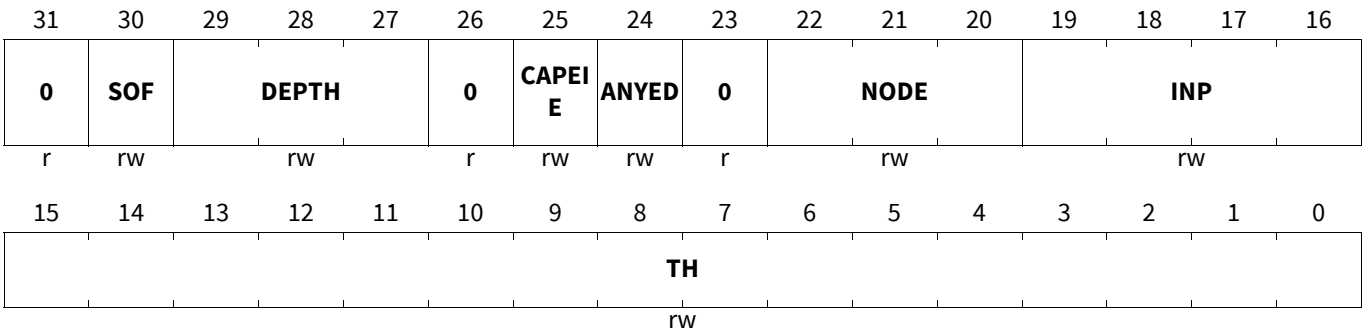
Oscillator calibration

The following registers support the oscillator calibration on which the decision is taken to increase or decrease the frequency. A detailed description will be provided as Application Node.

Measure Control Register

The Measure Control Register MECR controls the CAN edge timing measurement function for calibration purposes. This feature only exists on CAN0.

MECR
Measure Control Register **(008040_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TH	15:0	rw	**Threshold** This bit field contains the threshold value for the measurement timer. If TH = 0000_H, the timer is stopped and the capture function is disabled.
INP	19:16	rw	**Interrupt Node Pointer** INP selects the interrupt output line INT_Om (m = 0-15) for a capture event interrupt. 0_H Interrupt output line INT_O0 is selected … F_H Interrupt output line INT_O15 is selected
NODE	22:20	rw	**Node** This bit field determines the CAN node i whose input line RXDCANi is used for start and capture of the measurement timer. 000_B Node 0 … 011_B Node 3
ANYED	24	rw	**Any Edge** This bit enables capture on any edge of CAN input line specified by NODE. 0_B Capture on falling (dominant) edge only 1_B Capture on rising (recessive) or falling (dominant) edge
CAPEIE	25	rw	**Capture Event Interrupt Enable** This bit enables the capture event interrupt. Bit field INP selects the interrupt output line which becomes activated at this type of interrupt. 0_B Capture event interrupt is disabled 1_B Capture event interrupt is enabled

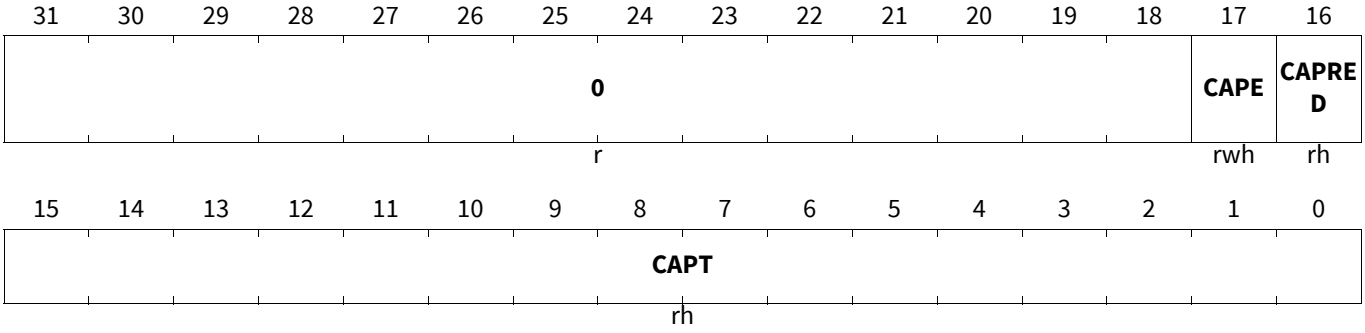
Field	Bits	Type	Description
DEPTH	29:27	rw	**Digital Glitch Filter Depth** DEPTH determines the number of input samples clocked with f_{SYNi} that are taken into account for the calculation of the floating average. The higher DEPTH is chosen to be, the longer the glitches that are suppressed and the longer the delay of the input signal introduced by this filter. 000_B off, default 001_B Filter depth of 8 cycles 010_B Filter depth of 16 cycles 011_B Filter depth of 32 cycles 100_B Filter depth of 64 cycles 101_B Filter depth of 128 cycles 110_B Filter depth of 255 cycles 111_B not allowed, reserved
SOF	30	rw	**Start Of Frame** This bit selects falling edge or any edge as measurement for start of frame detection. 0_B Measurement starts with any falling edge 1_B Measurement starts with falling Start of Frame edge. i.e any falling edge that occurs while the CAN node is in idle state
0	23, 26, 31	r	**Reserved** Shall read 0; shall be written with 0.

Measure Status Register

The Measure Status Register MESTAT contains the status information of the CAN edge timing measurement. This feature only exists on CAN0.

MESTAT

Measure Status Register **(008044_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CAPT	15:0	rh	**Captured Timer** This bit field contains the captured measurement timer content. The timer itself is cleared and started by the first falling (dominant) edge of a CAN frame on the input line of the CAN node specified by MECR.NODE. The timer is incremented by the module control clock f_{SYNi} and will be stopped when $FFFF_H$ is reached. If MECR.TH = 0000_H, the timer is always stopped. A capture will take place if all the following conditions are met: 1. MECR.TH > 0000_H 2. Timer is cleared and started by new frame 3. Timer reaches MECR.TH 4. This node is not sending and first edge (as specified by MECR.ANYED) after 3. occurs on input line Capture will be repeated for the following CAN frames until MECR.TH is cleared.
CAPRED	16	rh	**Captured Rising Edge** This bit indicates the type of edge that caused the last capture event. 0_B Capture occurred on falling (dominant) edge 1_B Capture occurred on rising (recessive) edge
CAPE	17	rwh	**Capture Event** This flag is set on a capture event. It must be reset by software. An interrupt request is generated if MECR.CAPEIE = 1. If CAPE=1 then no further measurement results are posted to MESTAT.CAPT and MESTAT.CAPRED. CAPE bit has to be cleared to re-enable update of MESTAT.CAPT and MESTAT.CAPRED. 0_B No capture event has occurred since last flag reset 1_B Capture event has occurred since last flag reset
0	31:18	r	**Reserved** Shall read 0; shall be written with 0.

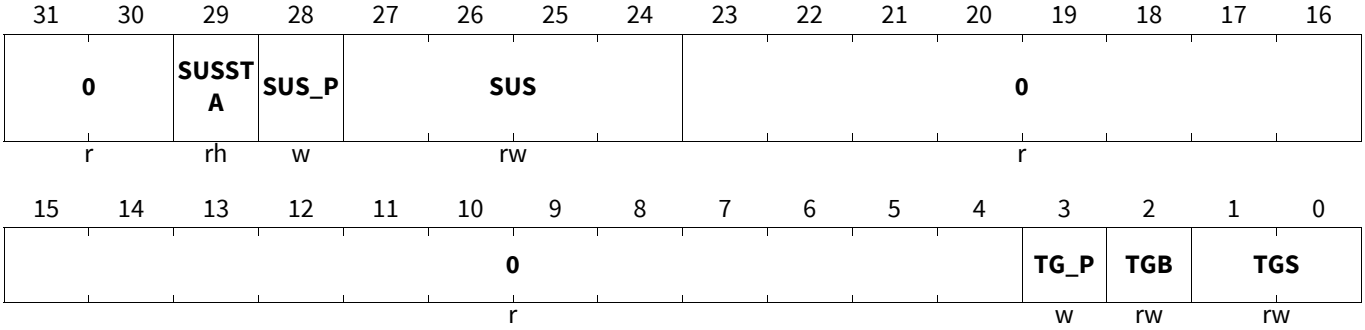
40.4.3.2 System Registers

OCDS Control and Status

OCDS Trigger Bus (OTGB) The OCDS Control and Status (OCS) register is cleared by Debug Reset. The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access requires Supervisor Mode.

OCS

OCDS Control and Status **(0080E8$_H$)** **Debug Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TGS	1:0	rw	**Trigger Set for OTGB0/1** 00_B No Trigger Set output 01_B TS16_CAN **others**, reserved
TGB	2	rw	**OTGB0/1 Bus Select** 0_B Trigger Set is output on OTGB0 1_B Trigger Set is output on OTGB1
TG_P	3	w	**TGS, TGB Write Protection** TGS and TGB are only written when TG_P is 1, otherwise unchanged. Read as 0.
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is off immediately. Do not use this mode in normal CAN applications, this mode is meant for debugging the peripheral IP. 2_H Soft suspend of CAN nodes. **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B CAN nodes are not (yet) suspended 1_B All CAN nodes are suspended
0	23:4, 31:30	r	**Reserved** Shall read 0; shall be written with 0.

OCDS Trigger Bus (OTGB) Interface

The MCMCAN Trigger Set is shown in **Table 380**. Its output is on OTGB0 or OTGB1 controlled by the **OCDS Control and Status** register. Links are only for CAN0, but the feature is available for all CAN modules.

Table 380 TS16_CAN Trigger Set MCMCAN

Values	Name	Description
i	AF	Acceptance filtering done for node i
i + 4	MR	Message successfully received on node i
i + 8	FDR	Fast Data Phase reception on node i
i + 12	FDT	Fast Data Phase transmission on node i.

40.4.3.3 Access Enable Registers ACCEN

The access enable bits control the access on the module itself.

Access Enable Register 0

The Access Enable Register 0 controls write access[1] for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 provides one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENy: EN0 -> TAG ID 000000_B, EN1 -> TAG ID 000001_B, … ,EN31 -> TAG ID 011111_B.

ACCEN0

Access Enable Register 0 ($0080FC_H$) Application Reset Value: $FFFF\ FFFF_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENy (y=0-31)	y	rw	**Access Enable for Master TAG ID y** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID y. 0_B Write access will not be executed 1_B Write access will be executed

40.4.3.4 Additional Access Enable Registers ACCENCTR and ACCENNODEi

The access enable bits help the application to control the access rights via bus master TAG ID. Inside the CAN node registers, an additional address range can be defined, called STARTADR and ENDADR.

The ACCENNODEi registers are protecting node i and the memory range defined in the STARTADRi and ENDADRi register. To disable the mechanism the STARTADRi has to be higher than the corresponding ENDADRi of the node.

1) The BPI_FPI Access Enable functionality controls only write transactions. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers-

Access Enable Register Control 0

The Access Enable Register Control 0 controls write access for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID ↔ master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCENCTR0 provides one enable bit for each possible 6-bit TAG ID encoding. The control registers (address range 8020_H to $804F_H$) are protected by this register.

Mapping of TAG IDs to ACCENCTR0.ENy: EN0 → TAG ID 000000_B, EN1 → TAG ID 000001_B, … ,EN31 → TAG ID 011111_B.

ACCENCTR0
Access Enable Register Control 0 ($0080DC_H$) Application Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENy (y=0-31)	y	rw	**Access Enable for Master TAG ID y** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID y 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register CAN Node i 0

The Access Enable Register CAN Node i Control 0 controls write access for transactions with the on chip bus master TAG ID 000000_B to 011111_B (see On Chip Bus chapter for the products TAG ID ↔ master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCENNODEi0 provides one enable bit for each possible 6-bit TAG ID encoding. This register controls node y registers, including the STARTADR0 and ENDADR0 register included in the node control area (8100_H+i*0400_H to 8400_H+i*0400_H-1). This means, that this area is including these registers and also the STARTADRi and ENDADRi. If the programming of the CPU leads to the fact, that it is disabling its own access, it will not be able to correct such a setting afterwards.

Mapping of TAG IDs to ACCENNODEi0.ENy: EN0 → TAG ID 000000_B, EN1 → TAG ID 000001_B, … , EN31 → TAG ID 011111_B.

ACCENNODEi0 (i=0-3)
Access Enable Register CAN Node i 0 (008100_H+i*400_H) Application Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENy (y=0-31)	y	rw	**Access Enable for Master TAG ID y** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID y 0_B Write access will not be executed 1_B Write access will be executed

40.4.3.5 Kernel Reset Registers

The Kernel Reset Registers give the user the possibility to reset the module without resetting the device.

Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers related to the module kernel reset.The RST bit will be reset by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is reset by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be reset to ´0´ by writing to KRSTCLR.CLR with ´1´.

During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **($0080F4_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RSTSTAT	RST
r														rh	rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (reset to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested

Field	Bits	Type	Description
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Shall read 0; shall be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers (KRST1.RST and KRST0.RST) related to the module kernel reset. The RST bit will be reset (cleared to ‘0’) by the BPI with the end of the BPI kernel reset sequence.

KRST1

Kernel Reset Register 1 **($0080F0_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															RST
r															rwh

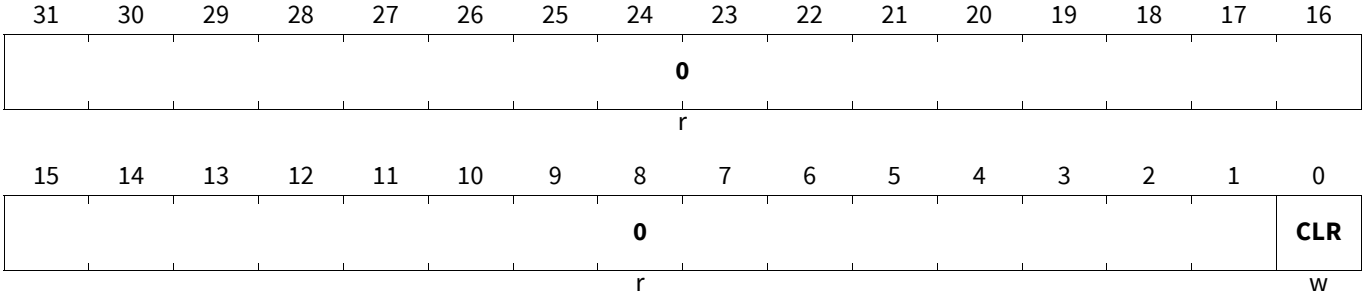
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (reset to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Shall read 0; shall be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Clear Register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **(0080EC$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Shall read 0; shall be written with 0.

40.4.4 MCMCAN User Interface Registers

This section describes the registers for clock control, port connections, interrupt control, and address decoding.

40.4.4.1 The Clock Control Register

CAN Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the standard interface for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the f_{SYN} and f_{ASYN} module clock signal, sleep mode and fast shut-off mode for the module.

CLC

CAN Clock Control Register **(008000_H)** **Application Reset Value: $0000\ 0003_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. The synchronous and asynchronous clock is switched on/off Note that no register access is possible to any register while module is disabled. A disable request is granted, if the M_CAN clock is disabled, or all M_CAN nodes acknowledge the disable request.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**Sleep Mode Disable Control** Used to control module's sleep mode.
0	2, 31:4	r	**Reserved** Shall read 0; shall be written with 0.

40.4.4.2 Interrupt Grouping and Signalling Registers

Interrupt routing for Groups 1 i

GRINT1i is the first of two grouping registers. In this register, the interrupt line within the module is fixed. Please be reminded, that the interrupt sources need to be enabled to be mapped. The total module has 16 interrupts and the interrupt node can be chosen within GRINT1i and GRINT2i.

Meaning:

0000_B Interrupt output line INT_O0 is selected.

0001_B Interrupt output line INT_O1 is selected.

$\cdots_B \cdots$

1110_B Interrupt output line INT_O14 is selected.

1111_B Interrupt output line INT_O15 is selected.

GRINT1i (i=0-3)

Interrupt routing for Groups 1 i **(008114_H+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LOI				BOFF				SAFE				MOER			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALRT				WATI				HPE				TEFIFO			
rw				rw				rw				rw			

Field	Bits	Type	Description
TEFIFO	3:0	rw	**Transmit Event FIFO Incidents** are mapped here. IR.TEFF (Transmit Event FIFO Full) and IR.TEFN (Transmit Event FIFO New Entry)
HPE	7:4	rw	**High Priority Events** are mapped here, giving IR.HPM an interrupt level
WATI	11:8	rw	**Watermark interrupts** are mapped here: IR.TEFW (Transmit FIFO warning interrupt reached), IR.RF1W (Receive FIFO 1 warning interrupt reached). IR.RF0W (Receive FIFO 0 warning interrupt reached)
ALRT	15:12	rw	**ALERTS** All kind of alerts are mapped here. IR.EW (warning status), IR.EP (error passive), IR.TSW (timestamp wrap around), IR.TEFL (Transmit Event FIFO Element Lost), IR.RF0L (Receive FIFO 0 Message Lost), IR.RF1L (Receive FIFO 1 Message Lost). The following TTCAN error messages and warnings are also shown here: TTIR. CER (Configuration Error), TTIR.AW Application Watchdog, TTIR.WT (Watch Trigger), TTIR.IWT Initialization Watch Trigger, TTIR.ELC (Error Level Changed), TTIR.SE2 (Scheduling Error 2), TTIR.SE1 (Scheduling Error), TTIR.TXO (Tx Count Overflow), TTIR.TXU (TX Count Underflow), TTIR.GTE (Global Time Error), TTIR.GTD (Global Time Discontinuity) and TTIR.GTW (Global Time Wrap)
MOER	19:16	rw	**Module errors** IR.WDI (watchdog interrupt) and IR.MRAF (message RAM access failure) are mapped here.
SAFE	23:20	rw	**Safety counter overflow** The interrupt node for IR.ELO showing a safety counter overflow
BOFF	27:24	rw	**Bus Off has been reached** Mapped to IRi.BO flag indication the change in Bus_Off status. To get out of bus off, the CCCRn.INIT bit has to be reset.

Field	Bits	Type	Description
LOI	31:28	rw	**Last Error Interrupts** The interrupt sources IR.PED (Protocol Error in Data Phase) and IR.PEA (Protocol Error in Arbitration Phase) are signalled here.

Interrupt routing for Groups 2 i

GRINT2i has the same functionality as GRINT1i, but for other interrupt sources. The interrupt sources need to be enabled to be mapped.

GRINT2i (i=0-3)
Interrupt routing for Groups 2 i **(008118_H+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRACO				**TRAQ**				**RETI**				**RxF0N**			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RxF1N				**RxF0F**				**RxF1F**				**REINT**			
rw				rw				rw				rw			

Field	Bits	Type	Description
REINT	3:0	rw	**Message stored in dedicated receive buffer interrupt (IR.DRX)** is assigned to interrupt node.
RxF1F	7:4	rw	**IR.RF1F** Receive FIFO1 full interrupt assigned to an interrupt node
RxF0F	11:8	rw	**IR.RF0F** Receive FIFO0 full interrupt assigned to an interrupt node
RxF1N	15:12	rw	**IR.RF1N** Receive FIFO1 new message assigned to an interrupt node
RxF0N	19:16	rw	**IR.RF0N** Receive FIFO0 new message assigned to an interrupt node
RETI	23:20	rw	**Receive Timeouts** can be assigned here. IR.TOO (time-out event) and TE (Timer Event)
TRAQ	27:24	rw	**Transmission Queue Events** can be assigned here. IR.TFE Transmission FIFO Empty
TRACO	31:28	rw	**Interrupts of the transmission control** can be assigned here. IR.TCF (Transmission Cancellation Finished) and IR.TF (Transmission Completed). As an additional information the copy of a local time event is shown here with TTIR.SWT (Stop Watch Event). Further on the TTIR.TTMI Trigger Time Event Internal, TTIR.RTMI (Register Time Mark), TTIR.SOG (Start of Gap), TTIR.CSM (Change of Synchronization Mode), TTIR.SMC (Start Matrix Cycle) and TTIR.SBC (Start of Basic Cycle) are shown here.

Interrupt Signalling Register i

The groups by the GRINT registers are also shown inside the ISREG (interrupt signalling register) register. Inside the interrupt signalling register a 1 means, that one of the corresponding bits inside the interrupt (status) register of the corresponding M_CAN node, at least one group member is showing an interrupt. ISREG is purely ORing the interrupt status bits of the group to enable SW to have proper handling of the bits. Writing to ISREGi has no effect.

If ISREGi is written, this shall have no effect on the interrupt status inside the M_CAN nodes. The bits have to be reset inside the corresponding M_CAN nodes, see register CANn_IRi.

ISREGi (i=0-3)

Interrupt Signalling Register i **(008110_H+i*400_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOI	BOFF	SAFE	MOER	ALRT	WATI	HPE	TEFIFO	TRACO	TRAQ	RETI	RxF0N	RxF1N	RxF0F	RxF1F	REINT
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
REINT	0	rh	**A message stored in a receive buffer interrupt**
RxF1F	1	rh	**Receive FIFO1 is full interrupt**
RxF0F	2	rh	**Receive FIFO0 is full interrupt**
RxF1N	3	rh	**Receive FIFO1 got a new message interrupt**
RxF0N	4	rh	**Receive FIFO0 got a new message interrupt**
RETI	5	rh	**A receive timeout event interrupt**
TRAQ	6	rh	**A transmission queue event interrupt**
TRACO	7	rh	**A transmission control event interrupt**
TEFIFO	8	rh	**A Transmit Event FIFO Incident interrupt**
HPE	9	rh	**A high priority event interrupt**
WATI	10	rh	**A watermark interrupt has been reached**
ALRT	11	rh	**An alert interrupt**
MOER	12	rh	**Module error interrupt**
SAFE	13	rh	**The safety counter interrupt ELO**
BOFF	14	rh	**Bus Off Interrupt**
LOI	15	rh	**Last Error Interrupt**
0	31:16	r	**Reserved** Shall read 0; shall be written with 0.

40.4.4.3 Node Port Control Register

Node i Port Control Register

The Node Port Control Register NPCRi configures the CAN bus transmit/receive ports.

NPCRi (i=0-3)

Node i Port Control Register **(008140_H+i*400_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

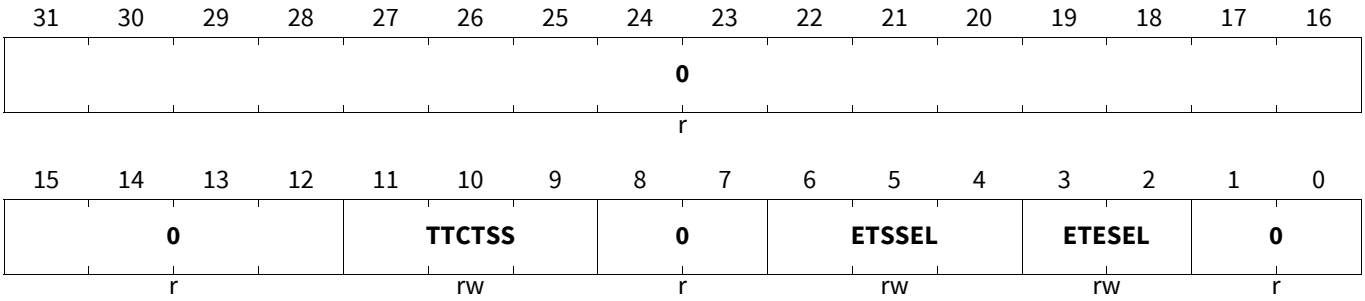
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					DELE	LOUT	LBM	0					RXSEL		
r					rw	rw	rw	r					rw		

Field	Bits	Type	Description
RXSEL	2:0	rw	**Receive Select** RXSEL selects one out of 8 possible receive inputs. The CAN receive signal is performed by the selected input. (see the device related chapter for RXSEL)
LBM	8	rw	**Loop-Back Mode** 0_B Loop-Back Mode is disabled. 1_B Loop-Back Mode is enabled. This node is connected to an internal (virtual) loop-back CAN bus. All CAN nodes which are in Loop-Back Mode are connected to this virtual CAN bus so that they can communicate with each other internally. The external transmit line is forced recessive in Loop-Back Mode.
LOUT	9	rw	**Loop Back Mode Out** The loop back bus is switched to the external CAN bus of the node.
DELE	10	rw	**Enable destructive read on ECRi.CEL** If this bit is set, the destructive read on ECRi.CEL and on the PSR register takes place. Meaning, that with read access on ECRi, the CEL is reset. The same is true for the PSR register, for the bits PXE, RFDF, RBRS, RESI, LEC and DLEC. After the destructive read it is advised to reset the bit again.
0	7:3, 31:11	r	**Reserved** Shall read 0; shall be written with 0.

40.4.4.4 Time Trigger Control Register

Time Trigger Control Register

TTCR0
Time Trigger Control Register **(0081F0$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
ETESEL	3:2	rw	**External Trigger Event Selection** This bit field defines the external trigger event that can be used to trigger the transmission of the reference message. The event causes the Event Trigger to be triggered. Control settings for this will not be influenced. 00_B The external event ECTTx does not trigger the transmission of the reference message. 01_B The reference message will be transmitted when a negative edge is detected at the selected input line ECTTx. 10_B The reference message will be transmitted when a positive edge is detected at the input line ECTTx. 11_B The reference message will be transmitted when a negative edge or a positive edge is detected at the input line ECTTx.
ETSSEL	6:4	rw	**External Trigger Source Selection** This bit fields selects the input source for the external reference message trigger. 000_B External trigger input line ECTT1 selected 001_B External trigger input line ECTT2 selected 010_B External trigger input line ECTT3 selected 011_B External trigger input line ECTT4 selected 100_B External trigger input line ECTT5 selected 101_B External trigger input line ECTT6 selected 110_B External trigger input line ECTT7 selected 111_B External trigger input line ECTT8 selected
TTCTSS	11:9	rw	**TTCapture Time Trigger Source Select** This bit selects the input source for the TT Capture Time (TTCPT) trigger. This register influences the stop watch event trigger 000_B No TTCPT trigger input allowed 001_B Local time register capture trigger input TTCPT_TRIG1 selected … 100_B Local time register capture trigger input TTCPT_TRIG4 selected **others**, Reserved; do not use this combination

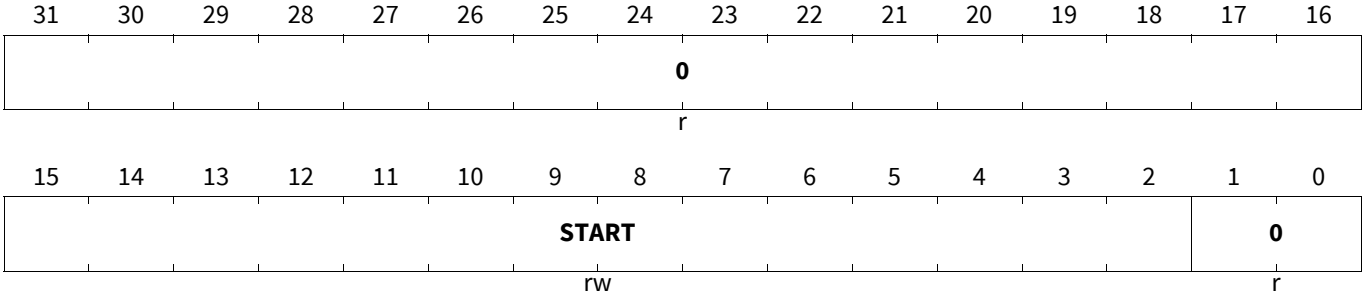
Field	Bits	Type	Description
0	1:0, 8:7, 31:12	r	**Reserved** Shall read 0; shall be written with 0.

40.4.4.5 Message RAM start address register

Start Address Node i

In case the RAM shall not be protected, the STARTADR has to be higher than the corresponding ENDADR of the node.

STARTADRi (i=0-3)
Start Address Node i **(008108_H+i*400_H)** **Application Reset Value: 0000 0000_H**

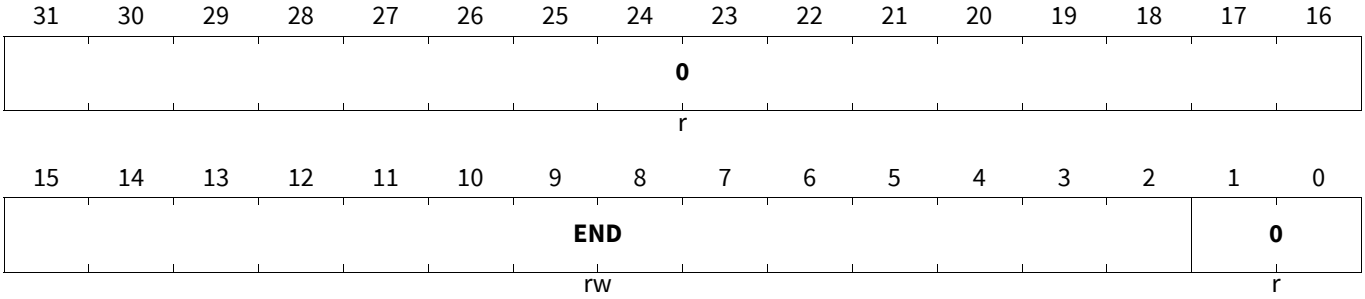


Field	Bits	Type	Description
START	15:2	rw	**Message RAM start** The address within the RAM area of the MCMCAN, of node i, where the message RAM to be protected starts
0	1:0, 31:16	r	**Reserved** Shall read 0; shall be written with 0.

40.4.4.6 Message RAM end address register

End Address Node i

ENDADRi (i=0-3)
End Address Node i **($00810C_H$+i*400_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
END	15:2	rw	**Message RAM end** The address within the RAM area of the MCMCAN, of node i, where the message RAM to be protected ends
0	1:0, 31:16	r	**Reserved** Shall read 0; shall be written with 0.

40.4.4.7 NTCCR

The Node i Timer Clock Control and Node i Timer A/B/C Transmit Trigger Registers offer additional timing functions for the node.

Node i Timer Clock Control Register

The Node i Timer Clock Control Register NTCCRi controls the functions of the node timer.

NTCCRi (i=0-3)
Node i Timer Clock Control Register **(008120_H+i*400_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											TRIGSRC			0	
r											rw			r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STSTART	STRESET	0		TPSC				0							
rw	rw	r		rw				r							

Field	Bits	Type	Description
TPSC	11:8	rw	**Timer Prescaler** The duration of one timer clock is given by (TPSC + 1) CAN bit times for all NTCCRi.TRIGSRC settings.
STRESET	14	rw	**Stamping Reset** This bit gives the possibility to reset the time stamp for CAN FD messages.
STSTART	15	rw	**Stamping Start** This bit starts the external timer used for CAN FD messages. The source and the prescaler are identical to the timers A/B/C.

Field	Bits	Type	Description
TRIGSRC	20:18	rw	**Trigger Source** This bit selects the trigger source for the different modes in the node timer. 000_B Node i Timer is decremented per f_{SYNi} prescaled by (TPSC + 1) timing to 0. 001_B System Timer (STM) trigger event enabled Node i Timer is decremented per STM trigger event prescaled by (TPSC + 1). 010_B General Timer (GTM) trigger event enabled Node i Timer is decremented per GTM trigger event prescaled by (TPSC + 1). **others**, Reserved, do not use
0	7:0, 13:12, 17:16, 31:21	r	**Reserved** Shall read 0; shall be written with 0.

40.4.4.8 CAN Node timers for pretended networking

Node i Timer A Transmit Trigger Register

The Node i Timer A Transmit Trigger Register NTATTRi controls the node timing functions for Transmit Trigger Mode.

NTATTRi (i=0-3)
Node i Timer A Transmit Trigger Register (008124_H+i*400_H) **Application Reset Value: 0001 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							STRT	TXMO							
r							rw	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD															
rw															

Field	Bits	Type	Description
RELOAD	15:0	rw	**Reload Value** This bit field contains the reload value for the timer. The timer will restart when RELOAD is written.
TXMO	23:16	r	**Transmit Message Object** This transmit trigger is fixed to transmit buffer 1
STRT	24	rw	**Timer Start** This bit field controls the operation of the timer. 0_B Timer is stopped. 1_B Timer is started.

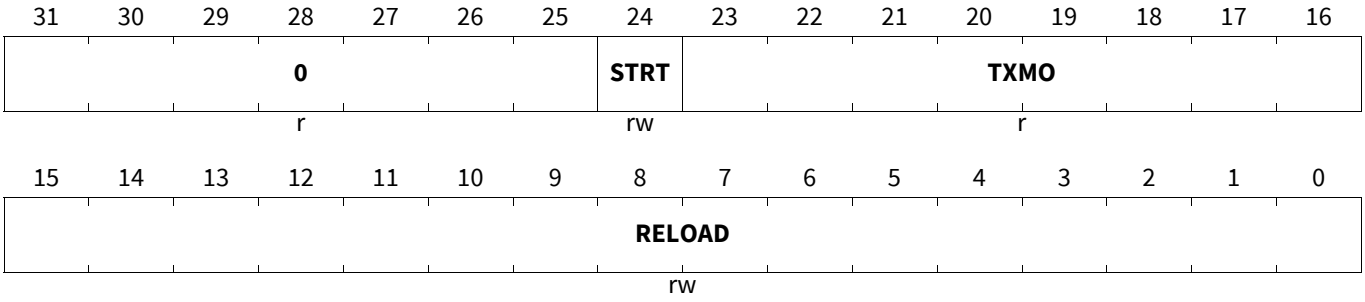
Field	Bits	Type	Description
0	31:25	r	**Reserved** Shall read 0; shall be written with 0.

Node i Timer B Transmit Trigger Register

The Node i Timer B Transmit Trigger Register NTBTTRi controls the node timing functions for Transmit Trigger Mode.

NTBTTRi (i=0-3)

Node i Timer B Transmit Trigger Register (008128$_H$+i*400$_H$) Application Reset Value: 0002 0000$_H$



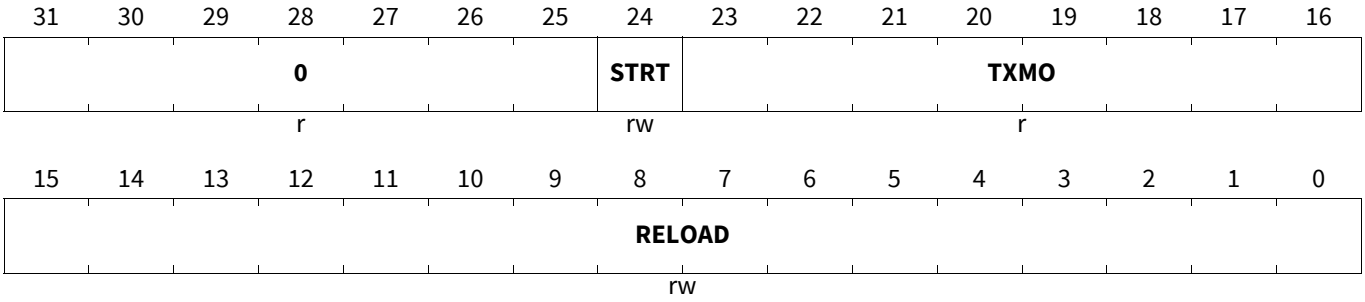
Field	Bits	Type	Description
RELOAD	15:0	rw	**Reload Value** This bit field contains the reload value for the timer. The timer will restart when RELOAD is written.
TXMO	23:16	r	**Transmit Message Object** This transmit object is fixed to transmit buffer 2
STRT	24	rw	**Timer Start** This bit field controls the operation of the timer. 0_B Timer is stopped. 1_B Timer is started.
0	31:25	r	**Reserved** Shall read 0; shall be written with 0.

Node i Timer C Transmit Trigger Register

The Node i Timer C Transmit Trigger Register NTCTTRi controls the node timing functions for Transmit Trigger Mode.

NTCTTRi (i=0-3)

Node i Timer C Transmit Trigger Register (00812C$_H$+i*400$_H$) Application Reset Value: 0003 0000$_H$



Field	Bits	Type	Description
RELOAD	15:0	rw	**Reload Value** This bit field contains the reload value for the timer. The timer will restart when RELOAD is written.
TXMO	23:16	r	**Transmit Message Object** This transmit trigger is fixed to transmit buffer 3
STRT	24	rw	**Timer Start** This bit field controls the operation of the timer. 0_B Timer is stopped. 1_B Timer is started.
0	31:25	r	**Reserved** Shall read 0; shall be written with 0.

40.4.4.9 Node Timer Receive Timerout Register

Node i Timer Receive Timeout Register

The Node i Timer Receive Timeout Register NTRTRi controls the node timing functions for Receive Timeout Mode. This feature is independent of Classical CAN and CAN FD.

This mode exists, to have for example network management supervision.

NTRTRi (i=0-3)
Node i Timer Receive Timeout Register **(008130_H+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								TE	TEIE	0					
r								rwh	rw	r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RELOAD															
rw															

Field	Bits	Type	Description
RELOAD	15:0	rw	**Reload Value** This bit field contains the reload value for the timer. The timer will start when RELOAD ≠ 0 is written. After half the time of the RELOAD value, the interrupt flags of the receive buffers will be cleared automatically, to ensure, that no message receive will be missed.
TEIE	22	rw	**Timer Event Interrupt Enable** This bit enables the node timer event interrupt of CAN node i. Bit field GRINT2.RETI selects the interrupt output line which becomes activated at this type of interrupt. 0_B Timer event interrupt is disabled 1_B Timer event interrupt is enabled

Field	Bits	Type	Description
TE	23	rwh	**Timer Event** This flag is set on a node timer transition from 1 to 0 in Receive Timeout Mode. This bit must be reset (i.e Write to '0') by software, writing a '1' has no effect. An interrupt request is generated if TEIE = 1. 0_B No timer event has occurred since last flag reset 1_B Timer event has occurred since last flag reset
0	21:16, 31:24	r	**Reserved** Shall read 0; shall be written with 0.

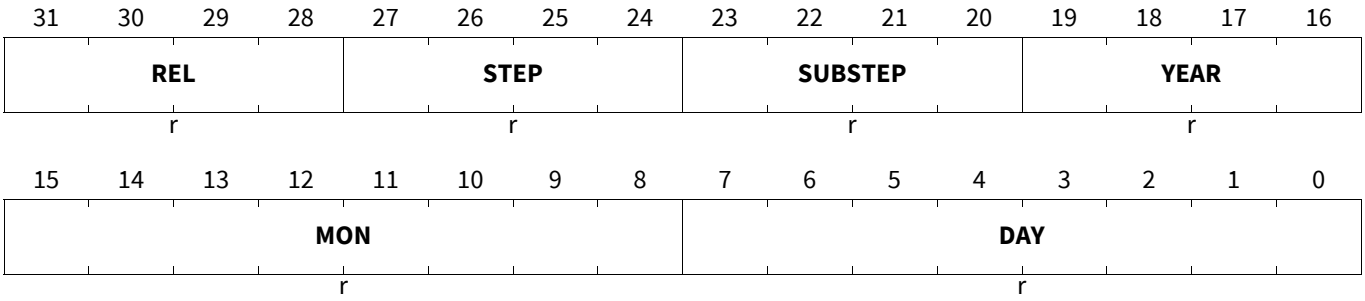
40.4.5 Registers within M_CAN

40.4.5.1 Standard Registers

Core Release Register i

CRELi (i=0-3)

Core Release Register i **(008200_H+i*400_H)** **Reset Value: Table 381**



Field	Bits	Type	Description
DAY	7:0	r	**Time Stamp Day**
MON	15:8	r	**Time Stamp Month**
YEAR	19:16	r	**Time Stamp Year**
SUBSTEP	23:20	r	**Sub-step of Core Release** One digit, BCD-coded.
STEP	27:24	r	**Step of Core Release** One digit, BCD-coded.
REL	31:28	r	**Core Release** One digit, BCD-coded.

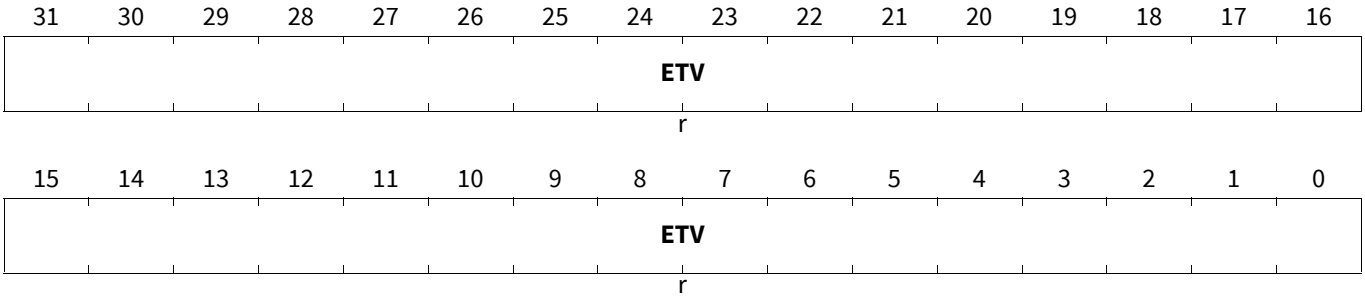
Table 381 Reset Values of CRELi (i=0-3)

Reset Type	Reset Value	Note
Application Reset	$3215\ 0323_H$	Node (i = 0) with TTCAN
Application Reset	$3215\ 0320_H$	Nodes (i > 0) without TTCAN

Endian Register i

ENDNi (i=0-3)

Endian Register i **(008204_H+i*400_H)** **Application Reset Value: $8765\ 4321_H$**



Field	Bits	Type	Description
ETV	31:0	r	**Endianness Test Value** The endianness test value is 0x87654321.

Data Bit Timing & Prescaler Register i

This register is only writable if bits CCCR.CCE and CCCR.INIT are set. The CAN bit time may be programmed in the range of 4 to 49 time quanta. The CAN time quantum may be programmed in the range of 1 to 32 clock cycles. t_q = (DBRP + 1) clock cycles.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [DTSEG1 + DTSEG2 + 3] t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

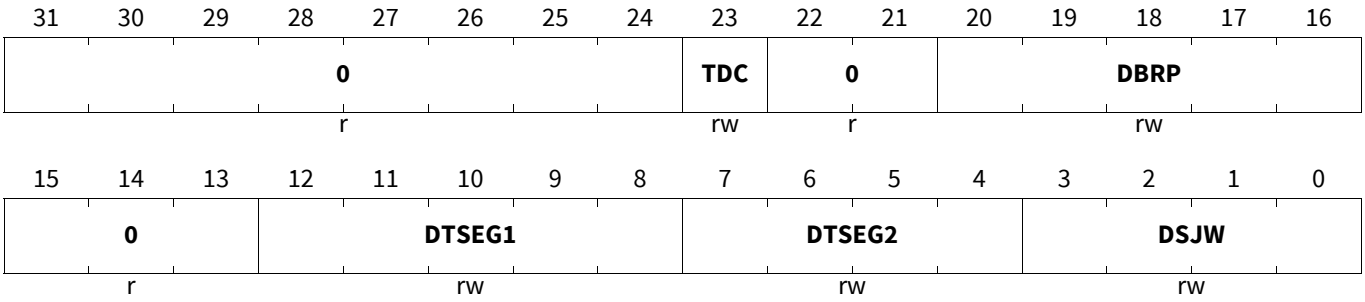
The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Notes

1. *With a CAN clock of 8 MHz, the reset value of 0x00000A33 configures the M_CAN for a fast bit rate of 500 kbit/s.*
2. *The bit rate configured for the CAN FD data phase via DBTP must be higher or equal to the bit rate configured for the arbitration phase via NBTP.*

DBTPi (i=0-3)

Data Bit Timing & Prescaler Register i **($00820C_H$+i*400_H)** **Application Reset Value: 0000 $0A33_H$**



Field	Bits	Type	Description
DSJW	3:0	rw	**Data (Re) Synchronization Jump Width** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
DTSEG2	7:4	rw	**Data time segment after sample point** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Field	Bits	Type	Description
DTSEG1	12:8	rw	**Data time segment before sample point** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
DBRP	20:16	rw	**Data Baud Rate Prescaler** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
TDC	23	rw	**Transmitter Delay Compensation** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Transmitter Delay Compensation disabled 1_B Transmitter Delay Compensation enabled
0	15:13, 22:21, 31:24	r	**Reserved** Shall read 0, shall be written with 0.

Test Register i

Write access to the Test Register has to be enabled by setting bit CCCR.TEST to ‘1’. All Test Register functions are set to their reset values when bit CCCR.TEST is reset.

Loop Back Mode and software control of transmit pin are hardware test modes. Programming of TEST.TX ≠ “00” may disturb the message transfer on the CAN bus.

TESTi (i=0-3)
Test Register i **(008210_H+i*400_H)** **Application Reset Value: 0000 00X0$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								RX	TX		LBCK	0	0	0	0
r								rh	rwh		rwh	r	r	r	r

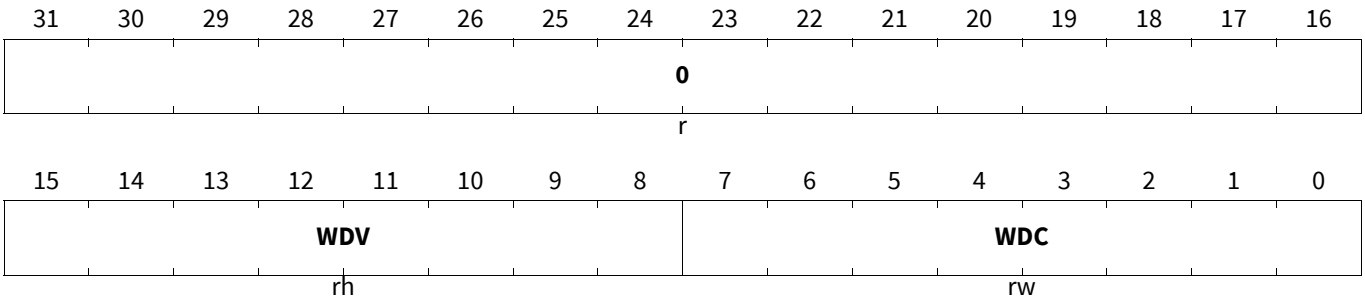
Field	Bits	Type	Description
LBCK	4	rwh	**Loop Back Mode** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. This is the external loop back mode, visible on the outside. 0_B Reset value, Loop Back Mode is disabled 1_B Loop Back Mode is enabled
TX	6:5	rwh	**Control of Transmit Pin** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 00_B Reset value, TX pin controlled by the CAN Core, updated at the end of the CAN bit time 01_B Sample Point can be monitored at the TX pin 10_B Dominant (‘0’) level at TX pin. 11_B Recessive (‘1’) at RX pin.
RX	7	rh	**Receive Pin** Monitors the actual value of RX pin. 0_B The CAN bus is dominant (RXD = ‘0’) 1_B The CAN bus is recessive (RXD = ‘1’)
0	0, 1, 2, 3, 31:8	r	**Reserved** Shall read 0, shall be written with 0.

RAM Watchdog i

The RAM Watchdog monitors the READY output of the Message RAM. A Message RAM access via the M_CAN’s Generic Master Interface starts the Message RAM Watchdog Counter with the value configured by RWD.WDC. The counter is reloaded with RWD.WDC when the Message RAM signals successful completion. In case there is no response from the Message RAM until the counter has counted down to zero, the counter stops and interrupt flag IR.WDI is set. The RAM Watchdog Counter is clocked by the Host clock.

RWDi (i=0-3)

RAM Watchdog i **(008214_H+i*400_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
WDC	7:0	rw	**Watchdog Configuration** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Start value of the Message RAM Watchdog Counter. With the reset value of “00” the counter is disabled. 00_H Watchdog disabled **others**, Start value of the Message RAM Watchdog Counter
WDV	15:8	rh	**Watchdog Value** Actual Message RAM Watchdog Counter Value.
0	31:16	r	**Reserved** Shall read 0, shall be written with 0.

CC Control Register i

The CCCRi register enables and disables CAN bus participation and basic protocol functions. Due to synchronization mechanisms between the clock domains, after a write operation to CCCRi, the register shall be read back, until the set values are written to the register. Please keep in mind, that the register also includes hardware influenced bits.

Note: After enabling the CAN clocks in MCR register, the application software has to wait for 10 hostclock cycles before accessing the kernel registers.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for rwh bits in this register.

CCCRi (i=0-3)
CC Control Register i **(008218_H+i*400_H)** **Application Reset Value: 0000 0001_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	0		BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw	r		rw	rw	rw	rw	rwh	rw	rh	rwh	rw	rwh

Field	Bits	Type	Description
INIT	0	rwh	**Initialization** *Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.* 0_B Normal Operation 1_B Initialization is started
CCE	1	rw	**Configuration Change Enable** 0_B The CPU has no write access to the protected configuration registers 1_B The CPU has write access to the protected configuration registers (while CCCR.INIT = '1')
ASM	2	rwh	**Restricted Operation Mode** Bit ASM can only be set by the Host when both CCE and INIT are set to '1'. In can also be set by the M_CAN. The bit can be reset by the Host at any time. For a description of the Restricted Operation Mode see paragraph Restricted Operation Mode. 0_B Normal CAN operation 1_B Restricted Operation Mode active
CSA	3	rh	**Clock Stop Acknowledge** 0_B No clock stop acknowledged 1_B M_CAN may be set in power down by stopping the synchronous and the asynchronous clock source
CSR	4	rw	**Clock Stop Request** 0_B No clock stop is requested 1_B Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.
MON	5	rwh	**Bus Monitoring Mode** Bit MON can only be set by the Host when both CCE and INIT are set to '1'. The bit can be reset by the Host at any time. 0_B Bus Monitoring Mode is disabled 1_B Bus Monitoring Mode is enabled
DAR	6	rw	**Disable Automatic Retransmission** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Automatic retransmission of messages not transmitted successfully enabled 1_B Automatic retransmission disabled

Field	Bits	Type	Description
TEST	7	rw	**Test Mode Enable** The TEST register can only be set, if CCE, INIT and TEST are set. Writes to test will only have effect, if all three bits are set. 0_B Normal operation, register TEST holds reset values 1_B Test Mode, write access to register TEST enabled
FDOE	8	rw	**FD Operation Enable** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B CAN FD frame format disabled. 1_B CAN FD frame format enabled.
BRSE	9	rw	**Bit Rate Switch Enable** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Bit rate switching for transmission disabled. 1_B Bit rate switching for transmission enabled.
PXHD	12	rw	**Protocol Exception Handling Disable** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Protocol exception handling enabled. 1_B Protocol exception handling disabled. (When protocol exception handling is disabled, the M_CAN will transmit an error frame when it detects a protocol exception condition.
EFBI	13	rw	**Edge Filtering during Bus Integration** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Edge filter disabled 1_B Two consecutive dominant tq required to detect an edge for hard synchronization.
TXP	14	rw	**Transmit Pause** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. If this bit is set, the M_CAN pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (see **Tx Handling**). 0_B Transmit pause disabled 1_B Transmit pause enabled
NISO	15	rw	**Non ISO Operation** If this bit is set, the M_CAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. 0_B CAN FD frame format according to ISO11898-1 1_B CAN FD frame format according to Bosch CAN FD Specification V1.0
0	11:10, 31:16	r	**Reserved** Shall read 0, shall be written with 0.

Nominal Bit Timing & Prescaler Register i

This register is only writable if bits CCCR.CCE and CCCR.INIT are set.

The CAN bit time may be programmed in the range of 4 to 385 time quanta. The CAN time quantum may be programmed in the range of 1 to 512 clock periods. t_q = (NBRP + 1) clock periods.

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2.

Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

The Information Processing Time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

Note: With a CAN clock of 8 MHz, the reset value of 0600_H0A03 configures the M_CAN for a bit rate of 500 kbit/s.

NBTPi (i=0-3)

Nominal Bit Timing & Prescaler Register i (00821C$_H$+i*400$_H$) **Application Reset Value: 0600 0A03$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW							NBRP								
rw							rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1								0	NTSEG2						
rw								r	rw						

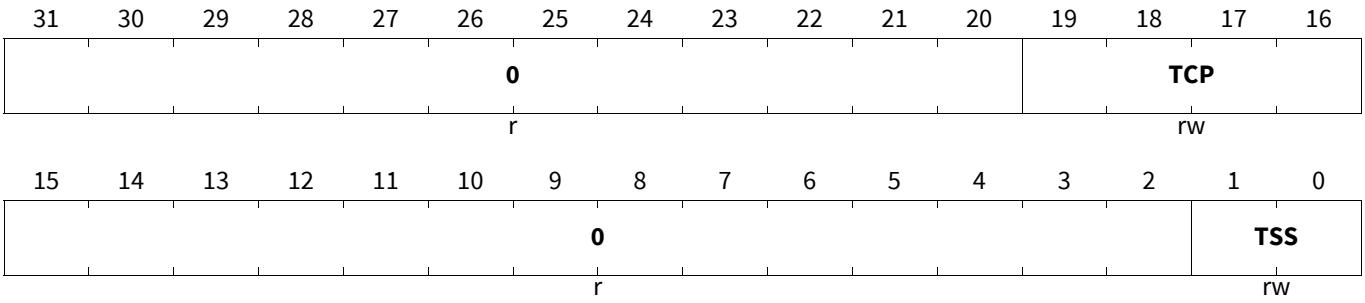
Field	Bits	Type	Description
NTSEG2	6:0	rw	**Nominal Time segment after sample point** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Valid values are 1 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
NTSEG1	15:8	rw	**Nominal Time segment before sample point** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.
NBRP	24:16	rw	**Baud Rate Prescaler** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
NSJW	31:25	rw	**(Re) Synchronization Jump Width** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Field	Bits	Type	Description
0	7	r	**Reserved** Shall read 0, shall be written with 0.

Timestamp Counter Configuration i

For a description of the Timestamp Counter see chapter Timestamp Generation

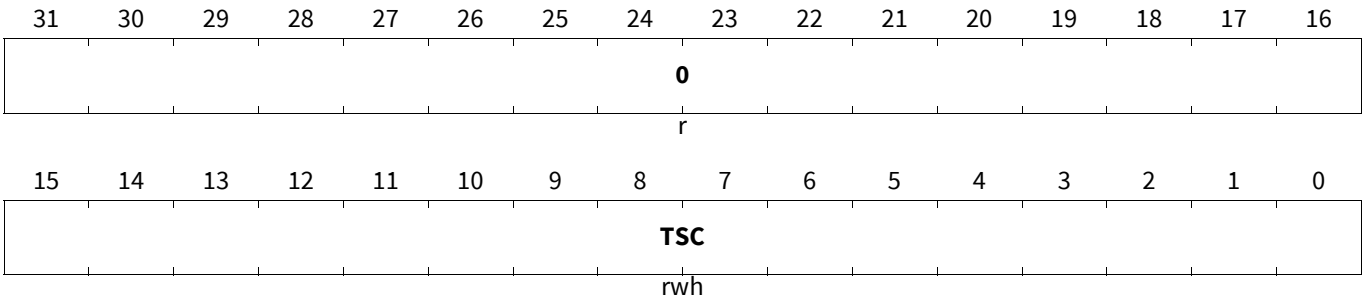
TSCCi (i=0-3)
Timestamp Counter Configuration i **(008220_H+i*400_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TSS	1:0	rw	**Time segment before sample point** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 01_B Timestamp counter value incremented according to TCP 10_B External timestamp counter value used, timer to be started in NTCCRy, the clock source as well as the chosen prescaler has to be configured before using this feature. **others**, Timestamp counter value always 0x0000
TCP	19:16	rw	**Timestamp Counter Prescaler** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. *Note: With CAN FD an external counter is required for timestamp generation (TSS = "10")*
0	15:2, 31:20	r	**Reserved** Shall read 0, shall be written with 0.

Timestamp Counter Value i

TSCVi (i=0-3)
Timestamp Counter Value i **(008224_H+i*400_H)** **Application Reset Value: 0000 0000_H**

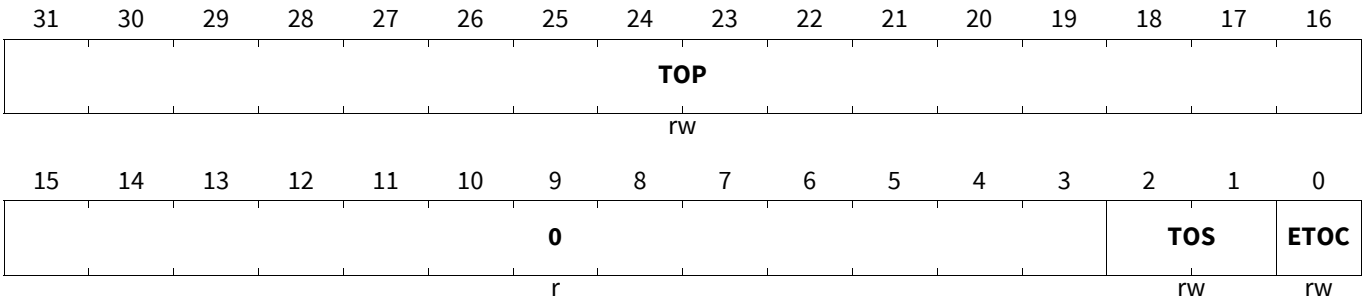


Field	Bits	Type	Description
TSC	15:0	rwh	**Timestamp Counter** The internal/external Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSCC.TSS = “01”, the Timestamp Counter is incremented in multiples of CAN bit times [1…16] depending on the configuration of TSCC.TCP. A wrap around sets interrupt flag IR.TSW. Write access resets the counter to zero. When TSCC.TSS = “10”, TSC reflects the external Timestamp Counter value. A write access has no impact. *Note: A “wrap around” is a change of the Timestamp Counter value from non-zero to zero not caused by write access to TSCV.*
0	31:16	r	**Reserved** Shall read 0, shall be written with 0.

Timeout Counter Configuration i

For a description of the Timeout Counter see **Timeout Counter**

TOCCi (i=0-3)
Timeout Counter Configuration i **(008228_H+i*400_H)** **Application Reset Value: FFFF 0000_H**

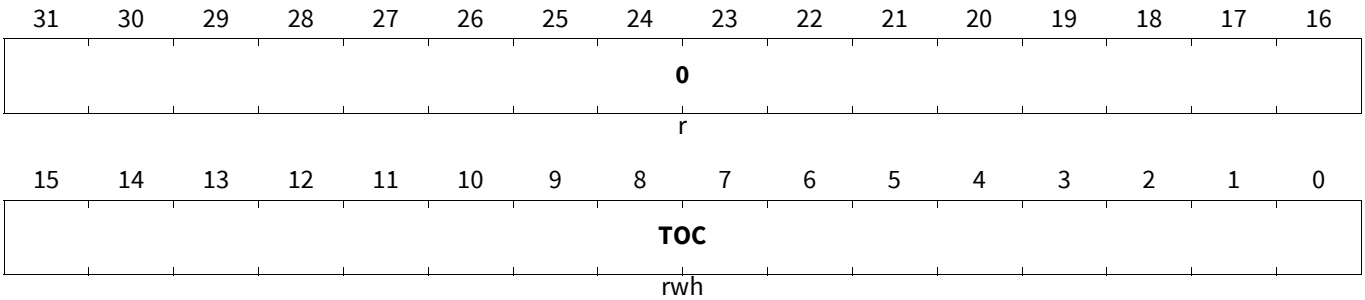


Field	Bits	Type	Description
ETOC	0	rw	**Enable Timeout Counter** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. *Note: For use of timeout function with CAN FD see chapter Timeout Counter.* 0_B Timeout Counter disabled 1_B Timeout Counter enabled
TOS	2:1	rw	**Timeout Select** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. When operating in Continuous mode, a write to TOCV presets the counter to the value configured by TOCC.TOP and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOCC.TOP. Down-counting is started when the first FIFO element is stored. 00_B Continuous operation 01_B Timeout controlled by Tx Event FIFO 10_B Timeout controlled by Rx FIFO 0 11_B Timeout controlled by Rx FIFO 1
TOP	31:16	rw	**Timeout Period** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
0	15:3	r	**Reserved** Shall read 0, shall be written with 0.

Timeout Counter Value i

TOCVi (i=0-3)

Timeout Counter Value i **($00822C_H$+i*400_H)** **Application Reset Value: 0000 FFFF$_H$**



Field	Bits	Type	Description
TOC	15:0	rwh	**Timeout Counter** The Timeout Counter is decremented in multiples of CAN bit times [1…16] depending on the configuration of TSCC.TCP. When decremented to zero, interrupt flag IR.TOO is set and the Timeout Counter is stopped. Start and reset/restart conditions are configured via TOCC.TOS. Any write access will lead to clearing of the counter.
0	31:16	r	**Reserved** Shall read 0, shall be written with 0.

Error Counter Register i

ECRi (i=0-3)
Error Counter Register i **(008240_H+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								CEL							
r								rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC							TEC							
rh	rh							rh							

Field	Bits	Type	Description
TEC	7:0	rh	**Transmit Error Counter** Actual state of the Transmit Error Counter, values between 0 and 255 *Note: When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.*
REC	14:8	rh	**Receive Error Counter** Actual state of the Receive Error Counter, values between 0 and 127 *Note: When CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.*
RP	15	rh	**Receive Error Passive** 0_B The Receive Error Counter is below the error passive level of 128 1_B The Receive Error Counter has reached the error passive level of 128

Field	Bits	Type	Description
CEL	23:16	rh	**CAN Error Logging** The counter is incremented each time when a CAN protocol error causes the Transmit Error Counter or the Receive Error Counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag IR.ELO. The counter is reset on read, if the bit NPCRi.DELE is set for the node.
0	31:24	r	**Reserved** Shall read 0, shall be written with 0.

Protocol Status Register i

PSRi (i=0-3)

Protocol Status Register i **(008244_H+i*400_H)** **Application Reset Value: 0000 0707_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									TDCV						
r									r						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PXE	RFDF	RBRS	RESI	DLEC			BO	EW	EP	ACT		LEC		
r	rh	rh	rh	rh	rh			rh	rh	rh	rh		rh		

Field	Bits	Type	Description
LEC	2:0	rh	**Last Error Code** The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. This bit field is set to 0x7 on read, if NPCRi.DELE is set. *Note: The Bus_Off recovery sequence (see ISO11898-1) cannot be shortened by setting or resetting CCCR.INIT. If the device goes Bus_Off, it will set CCCR.INIT of its own accord, stopping all bus activities. Once CCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR.REC is used to count these sequences.* 000_B No Error: No error occurred since LEC has been reset by successful reception or transmission. 001_B Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010_B Form Error: A fixed format part of a received frame has the wrong format. 011_B Ack Error: The message transmitted by the M_CAN was not acknowledged by another node. 100_B Bit1 Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant. 101_B Bit0 Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110_B CRC Error: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111_B No Change: Any read access to the Protocol Status Register re-initializes the LEC to '7'. When the LEC shows the value '7', no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

Field	Bits	Type	Description
ACT	4:3	rh	**Activity** Monitors the module's CAN communication state. *Note: ACT is set to "00" by a Protocol Exception Event.* 00_B Synchronizing - node is synchronizing on CAN communication 01_B Idle - node is neither receiver nor transmitter 10_B Receiver - node is operating as receiver 11_B Transmitter - node is operating as transmitter
EP	5	rh	**Error Passive** 0_B The M_CAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected 1_B The M_CAN is in the Error_Passive state
EW	6	rh	**Warning Status** 0_B Both error counters are below the Error_Warning limit of 96 1_B At least one of error counter has reached the Error_Warning limit of 96
BO	7	rh	**Bus_Off Status** 0_B The M_CAN is not in Bus_Off[1)] 1_B The M_CAN is in Bus_Off state
DLEC	10:8	rh	**Data Phase Last Error Code** Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error. This bit field is set to 0x7 on read, if NPCRi.DELE is set. *Note: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.*
RESI	11	rh	**ESI flag of last received CAN FD Message** This bit is set together with REDF, independent of acceptance filtering. This bit is reset after read access, if NPCRi.DELE is set. 0_B Last received CAN FD message did not have its ESI flag set 1_B Last received CAN FD message had its ESI flag set
RBRS	12	rh	**BRS flag of last received CAN FD Message** This bit is set together with REDF, independent of acceptance filtering. This bit is reset after read access, if NPCRi.DELE is set. 0_B Last received CAN FD message did not have its BRS flag set 1_B Last received CAN FD message had its BRS flag set

Field	Bits	Type	Description
RFDF	13	rh	**Received a CAN FD Message** This bit is set independent of acceptance filtering. This bit is reset after read access, if NPCRi.DELE is set. 0_B Since this bit was reset by the CPU, no CAN FD message has been received 1_B Message in CAN FD format with FDF flag set, has been received
PXE	14	rh	**Protocol Exception Event** This bit is reset after read access, if NPCRi.DELE is set. 0_B No protocol exception event occurred since last read access 1_B Protocol exception event occurred
TDCV	22:16	r	**Transmitter Delay Compensation Value** Position of the secondary sample point, defined by the sum of the measured delay from TX to RX and TDCR.TDCO. The SSP position is, in the data phase, the number of mtq between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.
0	15, 31:23	r	**Reserved** Shall read 0, shall be written with 0.

1) The Bus_Off recovery sequence (see ISO11898-1) cannot be shortened by setting or resetting CCCR.INIT. If the device goes Bus_Off, it will set CCCR.INIT of its own accord, stopping all bus activities. Once CCCR.INIT has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. ECR.REC is used to count these sequences.

Transmitter Delay Compensation Register i

TDCRi (i=0-3)

Transmitter Delay Compensation Register i(008248_H+i*400_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	TDCO							0	TDCF						
r	rw							r	rw						

Field	Bits	Type	Description
TDCF	6:0	rw	**Transmitter Delay Compensation Filter Window Length** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Defines the minimum value for the Secondary Sample Point position, dominant edges on RX that would result in an earlier Secondary Sample Point position are ignored for transmitter delay measurement. This feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are from 0 to 127 mtq.

Field	Bits	Type	Description
TDCO	14:8	rw	**Transmitter Delay Compensation Offset** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Offset value defining the distance between the measured delay from TX to RX and the secondary sample point. Valid values are 0 to 127 mtq. The duration of one mtq is equal to the fASYNi clock period.
0	7, 31:15	r	**Reserved** Shall read 0, shall be written with 0.

Interrupt Register i

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the Host clears them. A flag is cleared by writing a “1” to the corresponding bit position. Writing a “0” has no effect. The configuration of IE controls whether an interrupt is generated.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all rwh bits in this register.

IRi (i=0-3)
Interrupt Register i **(008250_H+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**0**	**PED**	**PEA**	**WDI**	**BO**	**EW**	**EP**	**ELO**	**0**	**0**	**DRX**	**TOO**	**MRAF**	**TSW**
r		r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r	r	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFL	**TEFF**	**TEFW**	**TEFN**	**TFE**	**TCF**	**TC**	**HPM**	**RF1L**	**RF1F**	**RF1W**	**RF1N**	**RF0L**	**RF0F**	**RF0W**	**RF0N**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
RF0N	0	rwh	**Rx FIFO 0 New Message** 0_B No new message written to Rx FIFO 0 1_B New message written to Rx FIFO 0
RF0W	1	rwh	**Rx FIFO 0 Watermark Reached** 0_B Rx FIFO 0 fill level below watermark 1_B Rx FIFO 0 fill level reached watermark
RF0F	2	rwh	**Rx FIFO 0 Full** 0_B Rx FIFO 0 not full 1_B Rx FIFO 0 full
RF0L	3	rwh	**Rx FIFO 0 Message Lost** 0_B No Rx FIFO 0 message lost 1_B Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
RF1N	4	rwh	**Rx FIFO 1 New Message** 0_B No new message written to Rx FIFO 1 1_B New message written to Rx FIFO 1

Field	Bits	Type	Description
RF1W	5	rwh	**Rx FIFO 1 Watermark Reached** 0_B Rx FIFO 1 fill level below watermark 1_B Rx FIFO 1 fill level reached watermark
RF1F	6	rwh	**Rx FIFO 1 Full** 0_B Rx FIFO 1 not full 1_B Rx FIFO 1 full
RF1L	7	rwh	**Rx FIFO 1 Message Lost** 0_B No Rx FIFO 1 message lost 1_B Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
HPM	8	rwh	**High Priority Message** 0_B No high priority message received 1_B High priority message received
TC	9	rwh	**Transmission Completed** 0_B No transmission completed 1_B Transmission completed
TCF	10	rwh	**Transmission Cancellation Finished** 0_B No transmission cancellation finished 1_B Transmission cancellation finished
TFE	11	rwh	**Tx FIFO Empty** 0_B Tx FIFO non-empty 1_B Tx FIFO empty
TEFN	12	rwh	**Tx Event FIFO New Entry** 0_B Tx Event FIFO unchanged 1_B Tx Handler wrote Tx Event FIFO element
TEFW	13	rwh	**Tx Event FIFO Watermark Reached** 0_B Tx Event FIFO fill level below watermark 1_B Tx Event FIFO fill level reached watermark
TEFF	14	rwh	**Tx Event FIFO Full** 0_B Tx Event FIFO not full 1_B Tx Event FIFO full
TEFL	15	rwh	**Tx Event FIFO Element Lost** 0_B No Tx Event FIFO element lost 1_B Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero
TSW	16	rwh	**Timestamp Wraparound** 0_B No timestamp counter wrap-around 1_B Timestamp counter wrapped around

Field	Bits	Type	Description
MRAF	17	rwh	**Message RAM Access Failure** The flag is set, when the Rx Handler • has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. • was not able to write a message to the Message RAM. In this case message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the M_CAN is switched into Restricted Operation Mode. To leave Restricted Operation Mode, the Host CPU has to reset CCCR.ASM. 0_B No Message RAM access failure occurred 1_B Message RAM access failure occurred
TOO	18	rwh	**Timeout Occurred** 0_B No timeout 1_B Timeout reached
DRX	19	rwh	**Message stored to Dedicated Rx Buffer** The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0_B No Rx Buffer updated 1_B At least one received message stored into an Rx Buffer
ELO	22	rwh	**Error Logging Overflow** 0_B CAN Error Logging Counter did not overflow 1_B Overflow of CAN Error Logging Counter occurred
EP	23	rwh	**Error Passive** 0_B Error_Passive status unchanged 1_B Error_Passive status changed
EW	24	rwh	**Warning Status** 0_B Error_Warning status unchanged 1_B Error_Warning status changed
BO	25	rwh	**Bus_Off Status** 0_B Bus_Off status unchanged 1_B Bus_Off status changed
WDI	26	rwh	**Watchdog Interrupt** 0_B No Message RAM Watchdog event occurred 1_B Message RAM Watchdog event due to missing READY

Field	Bits	Type	Description
PEA	27	rwh	**Protocol Error in Arbitration Phase** (Nominal Bit Time is used) 0_B No protocol error in arbitration phase 1_B Protocol error in arbitration phase detected (PSR.LEC ≠ 0,7)
PED	28	rwh	**Protocol Error in Data Phase** (Data Bit Time is used) 0_B No protocol error in data phase detected 1_B Protocol error in data phase detected (PSR.DLEC ≠ 0,7)
0	20, 21, 29, 31:30	r	**Reserved** Shall read 0, shall be written with 0.

Interrupt Enable i

The settings in the Interrupt Enable register determine which status changes in the Interrupt Register will be signalled on an interrupt line.

IEi (i=0-3)

Interrupt Enable i **(008254_H+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**0**	**PEDE**	**PEAE**	**WDIE**	**BOE**	**EWE**	**EPE**	**ELOE**	**0**	**0**	**DRXE**	**TOOE**	**MRAFE**	**TSWE**
r		r	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFLE	**TEFFE**	**TEFWE**	**TEFNE**	**TFEE**	**TCFE**	**TCE**	**HPME**	**RF1LE**	**RF1FE**	**RF1WE**	**RF1NE**	**RF0LE**	**RF0FE**	**RF0WE**	**RF0NE**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RF0NE	0	rw	**Rx FIFO 0 New Message Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
RF0WE	1	rw	**Rx FIFO 0 Watermark Reached Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
RF0FE	2	rw	**Rx FIFO 0 Full Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
RF0LE	3	rw	**Rx FIFO 0 Message Lost Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled

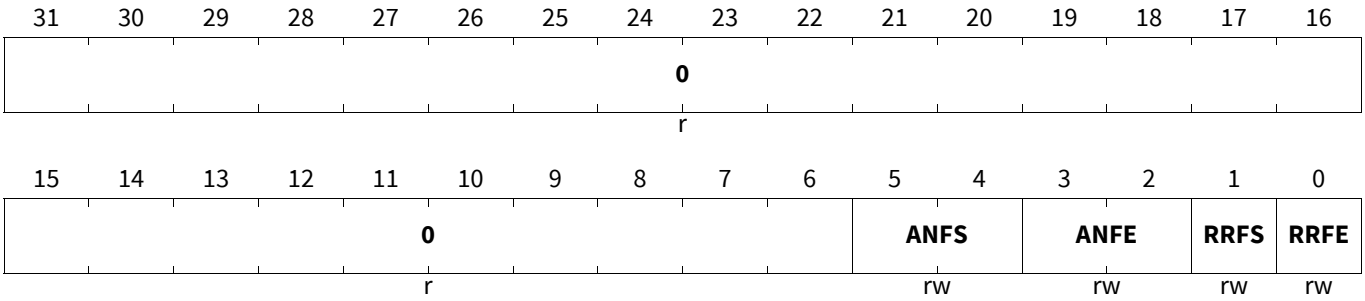
Field	Bits	Type	Description
RF1NE	4	rw	**Rx FIFO 1 New Message Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
RF1WE	5	rw	**Rx FIFO 1 Watermark Reached Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
RF1FE	6	rw	**Rx FIFO 1 Full Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
RF1LE	7	rw	**Rx FIFO 1 Message Lost Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
HPME	8	rw	**High Priority Message Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
TCE	9	rw	**Transmission Completed Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
TCFE	10	rw	**Transmission Cancellation Finished Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
TFEE	11	rw	**Tx FIFO Empty Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
TEFNE	12	rw	**Tx Event FIFO New Entry Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
TEFWE	13	rw	**Tx Event FIFO Watermark Reached Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
TEFFE	14	rw	**Tx Event FIFO Full Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
TEFLE	15	rw	**Tx Event FIFO Element Lost Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
TSWE	16	rw	**Timestamp Wraparound Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
MRAFE	17	rw	**Message RAM Access Failure Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
TOOE	18	rw	**Timeout Occurred Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled

Field	Bits	Type	Description
DRXE	19	rw	**Message stored to Dedicated Rx Buffer Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
ELOE	22	rw	**Error Logging Overflow Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
EPE	23	rw	**Error Passive Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
EWE	24	rw	**Warning Status Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
BOE	25	rw	**Bus_Off Status Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
WDIE	26	rw	**Watchdog Interrupt Enable** 0_B Interrupt disabled 1_B Interrupt enabled
PEAE	27	rw	**Protocol Error in Arbitration Phase Enable** 0_B Interrupt disabled 1_B Interrupt enabled
PEDE	28	rw	**Protocol Error in Data Phase Enable** 0_B Interrupt disabled 1_B Interrupt enabled
0	20, 21, 29, 31:30	r	**Reserved** Shall read 0, shall be written with 0.

Global Filter Configuration i

Global settings for Message ID filtering. The Global Filter Configuration controls the filter path for standard and extended messages as described in **Figure 594** and **Figure 595**.

GFCi (i=0-3)
Global Filter Configuration i **(008280_H+i*400_H)** **Application Reset Value: 0000 0000_H**



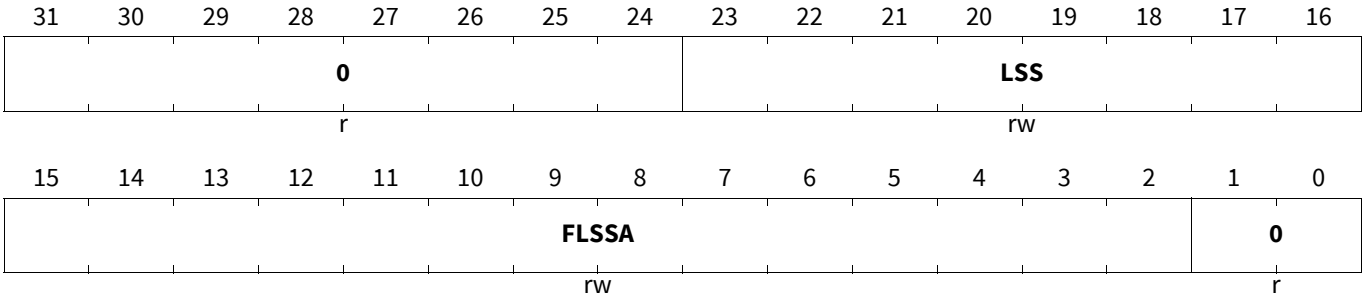
Field	Bits	Type	Description
RRFE	0	rw	**Reject Remote Frames Extended** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Filter remote frames with 29-bit extended IDs 1_B Reject all remote frames with 29-bit extended IDs
RRFS	1	rw	**Reject Remote Frames Standard** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Filter remote frames with 11-bit standard IDs 1_B Reject all remote frames with 11-bit standard IDs
ANFE	3:2	rw	**Accept Non-matching Frames Extended** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00_B Accept in Rx FIFO 0 01_B Accept in Rx FIFO 1 10_B Reject 11_B Reject
ANFS	5:4	rw	**Accept Non-matching Frames Standard** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00_B Accept in Rx FIFO 0 01_B Accept in Rx FIFO 1 10_B Reject 11_B Reject
0	31:6	r	**Reserved** Shall read 0, shall be written with 0.

Standard ID Filter Configuration i

Settings for 11-bit standard Message ID filtering. The Standard ID Filter Configuration controls the filter path for standard messages.

SIDFCi (i=0-3)

Standard ID Filter Configuration i **(008284_H+i*400_H)** **Application Reset Value: 0000 0000_H**

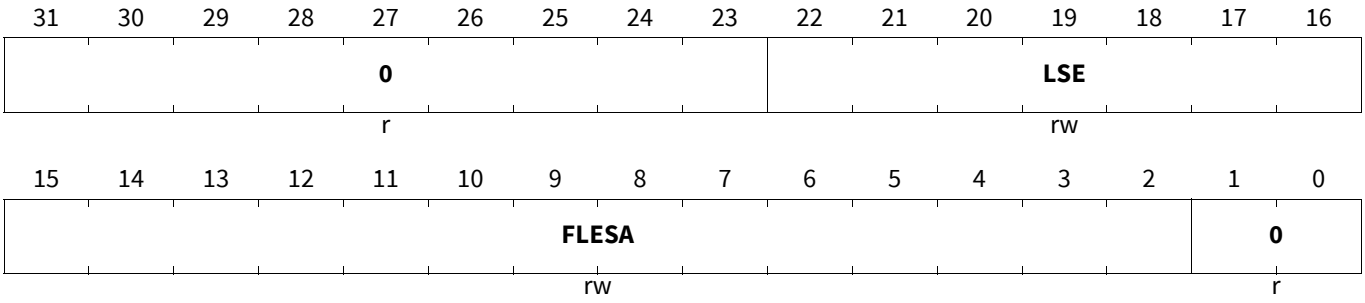


Field	Bits	Type	Description
FLSSA	15:2	rw	**Filter List Standard Start Address** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Start address of standard Message ID filter list (32-bit word address).
LSS	23:16	rw	**List Size Standard** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 00_H No standard Message ID filter 01_H 1 Message ID filter elements … 80_H 128 Message ID filter elements **others**, 128 Message ID filter elements
0	1:0, 31:24	r	**Reserved** Shall read 0, shall be written with 0.

Extended ID Filter Configuration i

Settings for 29-bit extended Message ID filtering. The Extended ID Filter Configuration controls the filter path for standard messages as described in **Figure 595**.

XIDFCi (i=0-3)
Extended ID Filter Configuration i **(008288_H+i*400_H)** **Application Reset Value: 0000 0000_H**



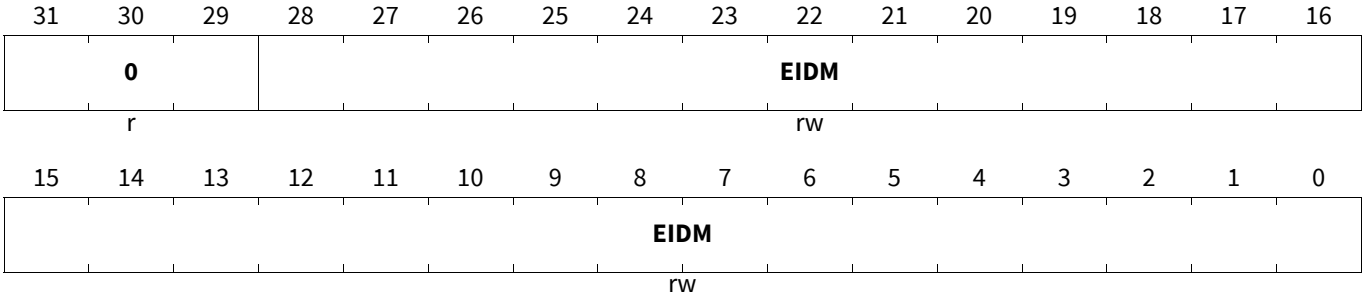
Field	Bits	Type	Description
FLESA	15:2	rw	**Filter List Extended Start Address** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Start address of extended Message ID filter list (32-bit word addess).
LSE	22:16	rw	**List Size Extended** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 00_H No standard Message ID filter 01_H 1 extended Message ID filter element … 40_H 64 extended Message ID filter element **others**, 64 extended Message ID filter elements

Field	Bits	Type	Description
0	1:0, 31:23	r	**Reserved** Shall read 0, shall be written with 0.

Extended ID AND Mask i

XIDAMi (i=0-3)

Extended ID AND Mask i **(008290_H+i*400_H)** **Application Reset Value: 1FFF FFFF$_H$**



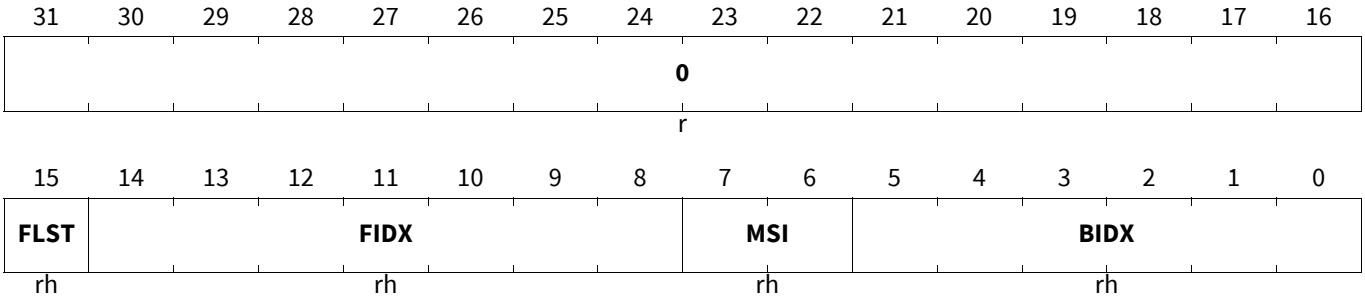
Field	Bits	Type	Description
EIDM	28:0	rw	**Extended ID Mask** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.
0	31:29	r	**Reserved** Shall read 0, shall be written with 0.

High Priority Message Status i

This register is updated every time a Message ID filter element configured to generate a priority event matches. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

HPMSi (i=0-3)

High Priority Message Status i **(008294_H+i*400_H)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
BIDX	5:0	rh	**Buffer Index** Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = '1'.
MSI	7:6	rh	**Message Storage Indicator** 00_B No FIFO selected 01_B FIFO message lost 10_B Message stored in FIFO 0 11_B Message stored in FIFO 1
FIDX	14:8	rh	**Filter Index** Index of matching filter element. Range is 0 to SIDFC.LSS - 1 resp. XIDFC.LSE - 1.
FLST	15	rh	**Filter List** Indicates the filter list of the matching filter element. 0_B Standard Filter List 1_B Extended Filter List
0	31:16	r	**Reserved** Shall read 0, shall be written with 0.

New Data 1 i

The register holds the New Data flags of Rx Buffers 0 to 31.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all rwh bits in this register.

NDAT1i (i=0-3)
New Data 1 i **(008298_H+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND31	**ND30**	**ND29**	**ND28**	**ND27**	**ND26**	**ND25**	**ND24**	**ND23**	**ND22**	**ND21**	**ND20**	**ND19**	**ND18**	**ND17**	**ND16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND15	**ND14**	**ND13**	**ND12**	**ND11**	**ND10**	**ND9**	**ND8**	**ND7**	**ND6**	**ND5**	**ND4**	**ND3**	**ND2**	**ND1**	**ND0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
NDy (y=0-31)	y	rwh	**New Data in Rx Buffer y - ND** The flag is set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a "1" to the corresponding bit position. Writing a "0" has no effect. 0_B Rx Buffer not updated 1_B Rx Buffer updated from new message

New Data 2 i

The register holds the New Data flags of Rx Buffers 32 to 63.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all rwh bits in this register.

NDAT2i (i=0-3)
New Data 2 i **($00829C_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
NDy (y=32-63)	y-32	rwh	**New Data in Rx Buffer y - ND** The flag is set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a “1” to the corresponding bit position. Writing a “0” has no effect. 0_B Rx Buffer not updated 1_B Rx Buffer updated from new message

Rx FIFO 0 Configuration i

RXF0Ci (i=0-3)
Rx FIFO 0 Configuration i **($0082A0_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F0OM	F0WM							0	F0S						
rw	rw							r	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F0SA														0	
rw														r	

Field	Bits	Type	Description
F0SA	15:2	rw	**Rx FIFO 0 Start Address** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Start address of Rx FIFO 0 in Message RAM (32-bit word address, see **Figure 605**).

Field	Bits	Type	Description
F0S	22:16	rw	**Rx FIFO 0 Size** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 00_H No Rx FIFO 0 01_H 1 Rx FIFO 0 elements … 40_H 64 Rx FIFO 0 elements **others**, 64 Rx FIFO 0 elements
F0WM	30:24	rw	**Rx FIFO 0 Watermark** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 01_H Level for Rx FIFO 0 watermark interrupt (IR.RF0W) … 40_H Level for Rx FIFO 0 watermark interrupt (IR.RF0W) **others**, Watermark interrupt disabled
F0OM	31	rw	**FIFO 0 Operation Mode** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. FIFO 0 can be operated in blocking or in overwrite mode (see **Rx FIFOs**). 0_B FIFO 0 blocking mode 1_B FIFO 0 overwrite mode
0	1:0, 23	r	**Reserved** Shall read 0, shall be written with 0.

Rx FIFO 0 Status i

RXF0Si (i=0-3)

Rx FIFO 0 Status i **($0082A4_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						RF0L	F0F	0		F0PI					
r						rh	rh	r		rh					

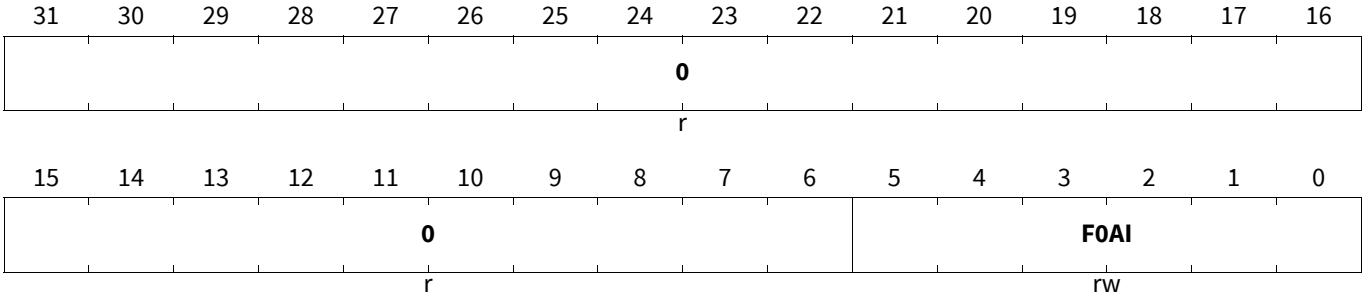
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		F0GI						0		F0FL					
r		rh						r		rh					

Field	Bits	Type	Description
F0FL	6:0	rh	**Rx FIFO 0 Fill Level** Number of elements stored in Rx FIFO 0, range 0 to 64.
F0GI	13:8	rh	**Rx FIFO 0 Get Index** Rx FIFO 0 read index pointer, range 0 to 63.
F0PI	21:16	rh	**Rx FIFO 0 Put Index** Rx FIFO 0 write index pointer, range 0 to 63.

Field	Bits	Type	Description
F0F	24	rh	**Rx FIFO 0 Full** 0_B Rx FIFO 0 not full 1_B Rx FIFO 0 full
RF0L	25	rh	**Rx FIFO 0 Message Lost** This bit is a copy of interrupt flag IR.RF0L. When IR.RF0L is reset, this bit is also reset. *Note: Overwriting the oldest message when RXF0C.F0OM = '1' will not set this flag.* 0_B No Rx FIFO 0 message lost 1_B Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero
0	7, 15:14, 23:22, 31:26	r	**Reserved** Shall read 0, shall be written with 0.

Rx FIFO 0 Acknowledge i

RXF0Ai (i=0-3)
Rx FIFO 0 Acknowledge i **($0082A8_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

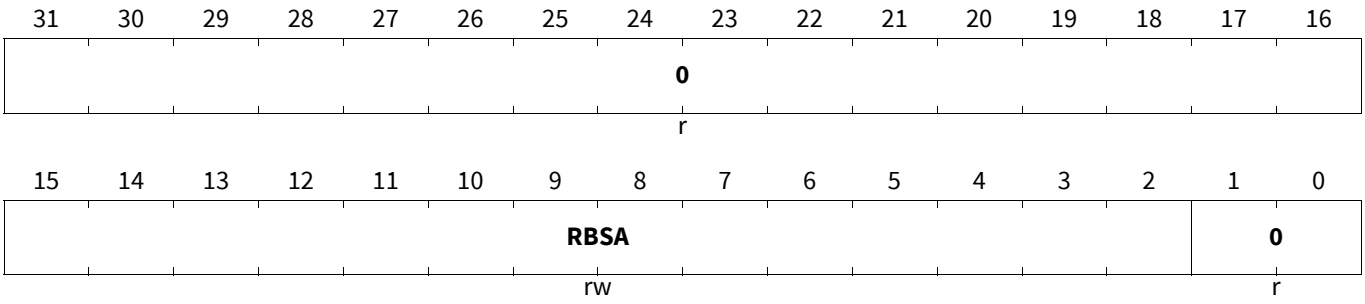


Field	Bits	Type	Description
F0AI	5:0	rw	**Rx FIFO 0 Acknowledge Index** After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index RXF0S.F0GI to F0AI + 1 and update the FIFO 0 Fill Level RXF0S.F0FL.
0	31:6	r	**Reserved** Shall read 0, shall be written with 0.

Rx Buffer Configuration i

RXBCi (i=0-3)

Rx Buffer Configuration i **(0082AC$_H$+i*400$_H$)** **Application Reset Value: 0000 0000$_H$**

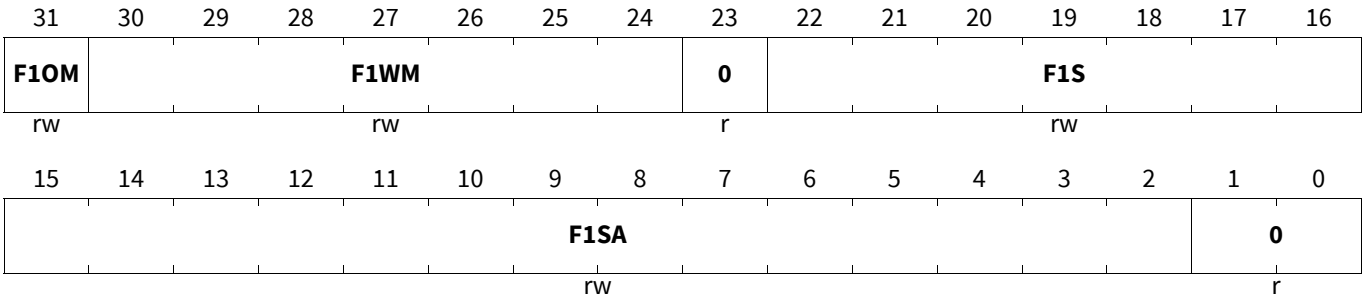


Field	Bits	Type	Description
RBSA	15:2	rw	**Rx Buffer Start Address** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address).
0	1:0, 31:16	r	**Reserved** Shall read 0, shall be written with 0.

Rx FIFO 1 Configuration i

RXF1Ci (i=0-3)

Rx FIFO 1 Configuration i **(0082B0$_H$+i*400$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
F1SA	15:2	rw	**Rx FIFO 1 Start Address** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Start address of Rx FIFO 1 in Message RAM (32-bit word address).

Field	Bits	Type	Description
F1S	22:16	rw	**Rx FIFO 1 Size** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 00_H No Rx FIFO 1 01_H 1 Rx FIFO 1 elements … 40_H 64 Rx FIFO 1 elements **others**, 64 Rx FIFO 1 elements
F1WM	30:24	rw	**Rx FIFO 1 Watermark** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 01_H Level for Rx FIFO 1 watermark interrupt (IR.RF1W) … 40_H Level for Rx FIFO 1 watermark interrupt (IR.RF1W) **others**, Watermark interrupt disabled
F1OM	31	rw	**FIFO 1 Operation Mode** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. FIFO 1 can be operated in blocking or in overwrite mode. 0_B FIFO 1 blocking mode 1_B FIFO 1 overwrite mode
0	1:0, 23	r	**Reserved** Shall read 0, shall be written with 0.

Rx FIFO 1 Status i

RXF1Si (i=0-3)

Rx FIFO 1 Status i **(0082B4$_H$+i*400$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**0**				**RF1L**	**F1F**	**0**		**F1PI**					
r		r				rh	rh	r		rh					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**F1GI**						**0**		**F1FL**					
r		rh						r		rh					

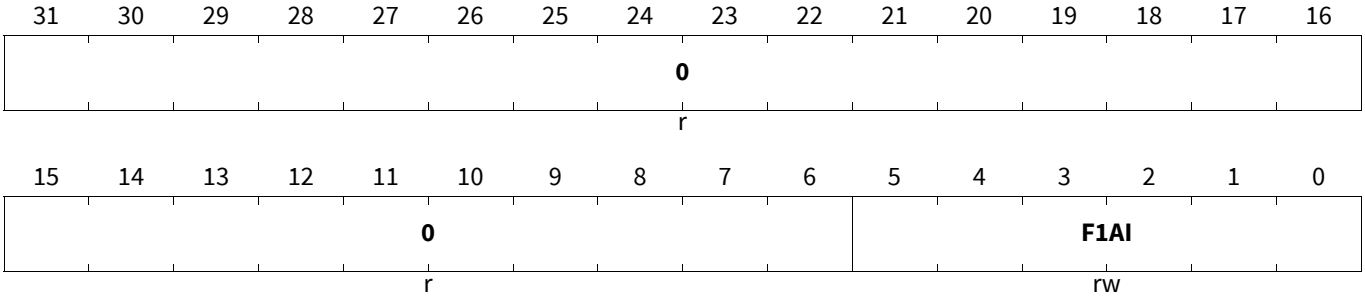
Field	Bits	Type	Description
F1FL	6:0	rh	**Rx FIFO 1 Fill Level** Number of elements stored in Rx FIFO 1, range 0 to 64.
F1GI	13:8	rh	**Rx FIFO 1 Get Index** Rx FIFO 1 read index pointer, range 0 to 63.
F1PI	21:16	rh	**Rx FIFO 1 Put Index** Rx FIFO 1 write index pointer, range 0 to 63.

Field	Bits	Type	Description
F1F	24	rh	**Rx FIFO 1 Full** 0_B Rx FIFO 1 not full 1_B Rx FIFO 1 full
RF1L	25	rh	**Rx FIFO 1 Message Lost** This bit is a copy of interrupt flag IR.RF1L. When IR.RF1L is reset, this bit is also reset. *Note: Overwriting the oldest message when RXF1C.F1OM = '1' will not set this flag.* 0_B No Rx FIFO 1 message lost 1_B Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero
0	7, 15:14, 23:22, 29:26, 31:30	r	**Reserved** Shall read 0, shall be written with 0.

Rx FIFO 1 Acknowledge i

RXF1Ai (i=0-3)

Rx FIFO 1 Acknowledge i **($0082B8_H$+i*400_H)** **Application Reset Value: 0000 0000_H**



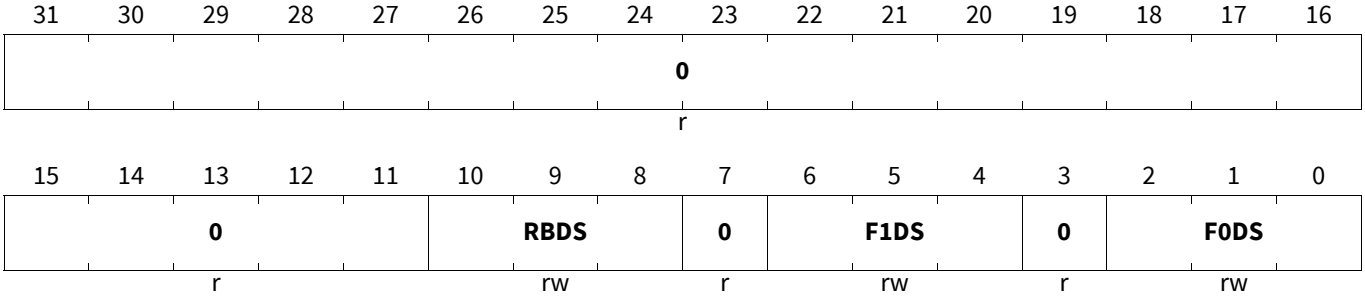
Field	Bits	Type	Description
F1AI	5:0	rw	**Rx FIFO 1 Acknowledge Index** After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index RXF1S.F1GI to F1AI + 1 and update the FIFO 1 Fill Level RXF1S.F1FL
0	31:6	r	**Reserved** Shall read 0, shall be written with 0.

Rx Buffer/FIFO Element Size Configuration i

Configures the number of data bytes belonging to an Rx Buffer / Rx FIFO element. Data field sizes > 8 bytes are intended for CAN FD operation only.

RXESCi (i=0-3)

Rx Buffer/FIFO Element Size Configuration i(0082BC$_H$+i*400$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
F0DS	2:0	rw	**Rx FIFO 0 Data Field Size** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. *Note: In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by RXESC are stored to the Rx Buffer resp. Rx FIFO element. The rest of the frame's data field is ignored.* 000_B 8-byte data field 001_B 12-byte data field 010_B 16-byte data field 011_B 20-byte data field 100_B 24-byte data field 101_B 32-byte data field 110_B 48-byte data field 111_B 64-byte data field
F1DS	6:4	rw	**Rx FIFO 1 Data Field Size** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 000_B 8-byte data field 001_B 12-byte data field 010_B 16-byte data field 011_B 20-byte data field 100_B 24-byte data field 101_B 32-byte data field 110_B 48-byte data field 111_B 64-byte data field

Field	Bits	Type	Description
RBDS	10:8	rw	**Rx Buffer Data Field Size** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 000_B 8-byte data field 001_B 12-byte data field 010_B 16-byte data field 011_B 20-byte data field 100_B 24-byte data field 101_B 32-byte data field 110_B 48-byte data field 111_B 64-byte data field
0	3, 7, 31:11	r	**Reserved** Shall read 0, shall be written with 0.

Tx Buffer Configuration i

TXBCi (i=0-3)
Tx Buffer Configuration i **($0082C0_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**TFQM**		**TFQS**				**0**		**NDTB**					
r		rw		rw				r		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBSA														**0**	
rw														r	

Field	Bits	Type	Description
TBSA	15:2	rw	**Tx Buffers Start Address** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Start address of Tx Buffers section in Message RAM (32-bit word address).
NDTB	21:16	rw	**Number of Dedicated Transmit Buffers** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. *Note: Be aware that the sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers.* 00_H No Dedicated Tx Buffers 01_H 1 Dedicated Tx Buffers … 20_H 32 Dedicated Tx Buffers **others**, 32 Dedicated Tx Buffers

Field	Bits	Type	Description
TFQS	29:24	rw	**Transmit FIFO/Queue Size** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 00_H No Tx FIFO/Queue 01_H 1 Tx Buffers used for Tx FIFO/Queue … 20_H 32 Tx Buffers used for Tx FIFO/Queue **others**, 32 Tx Buffers used for Tx FIFO/Queue
TFQM	30	rw	**Tx FIFO/Queue Mode** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Tx FIFO operation 1_B Tx Queue operation
0	1:0, 23:22, 31	r	**Reserved** Shall read 0, shall be written with 0.

Tx FIFO/Queue Status i

The Tx FIFO/Queue status is related to the pending Tx requests listed in register TXBRP. Therefore the effect of Add/Cancellation requests may be delayed due to a running Tx scan (TXBRP not yet updated).

TXFQSi (i=0-3)
Tx FIFO/Queue Status i **($0082C4_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										TFQF	TFQPI				
r										rh	rh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			TFGI					0		TFFL					
r			rh					r		rh					

Field	Bits	Type	Description
TFFL	5:0	rh	**Tx FIFO Free Level** Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TXBC.TFQM = '1') *Note: In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.*

Field	Bits	Type	Description
TFGI	12:8	rh	**Tx FIFO Get Index** Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TXBC.TFQM = '1').
TFQPI	20:16	rh	**Tx FIFO/Queue Put Index** Tx FIFO/Queue write index pointer, range 0 to 31.
TFQF	21	rh	**Tx FIFO/Queue Full** 0_B Tx FIFO/Queue not full 1_B Tx FIFO/Queue full
0	7:6, 15:13, 31:22	r	**Reserved** Shall read 0, shall be written with 0.

Tx Buffer Element Size Configuration i

Configures the number of data bytes belonging to a Tx Buffer element. Data field sizes > 8 bytes are intended for CAN FD operation only.

TXESCi (i=0-3)

Tx Buffer Element Size Configuration i **($0082C8_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													TBDS		
r													rw		

Field	Bits	Type	Description
TBDS	2:0	rw	**Tx Buffer Data Field Size** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. *Note: In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TXESC.TBDS, the bytes not defined by the Tx Buffer are transmitted as "0xCC" (padding bytes).* 000_B 8-byte data field 001_B 12-byte data field 010_B 16-byte data field 011_B 20-byte data field 100_B 24-byte data field 101_B 32-byte data field 110_B 48-byte data field 111_B 64-byte data field

Field	Bits	Type	Description
0	31:3	r	**Reserved** Shall read 0, shall be written with 0.

Tx Buffer Request Pending i

Each Tx Buffer has its own Transmission Request Pending bit. The bits are set via register TXBAR. The bits are reset after a requested transmission has completed or has been cancelled via register TXBCR.

TXBRP bits are set only for those Tx Buffers configured via TXBC. After a TXBRP bit has been set, a Tx scan (Tx Handling) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID).

A cancellation request resets the corresponding transmission request pending bit of register TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signalled via TXBCF

- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding TXBCF bit is set for all unsuccessful transmissions.

Note: TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this "Add Request" is cancelled immediately, the corresponding TXBRP bit is reset.

TXBRPi (i=0-3)
Tx Buffer Request Pending i **($0082CC_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRP31	**TRP30**	**TRP29**	**TRP28**	**TRP27**	**TRP26**	**TRP25**	**TRP24**	**TRP23**	**TRP22**	**TRP21**	**TRP20**	**TRP19**	**TRP18**	**TRP17**	**TRP16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRP15	**TRP14**	**TRP13**	**TRP12**	**TRP11**	**TRP10**	**TRP9**	**TRP8**	**TRP7**	**TRP6**	**TRP5**	**TRP4**	**TRP3**	**TRP2**	**TRP1**	**TRP0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TRPz (z=0-31)	z	rh	**Transmission Request Pending Tx Buffer z - TRP** 0_B No transmission request pending 1_B Transmission request pending

Tx Buffer Add Request i

Each Tx Buffer has its own "Add Request" bit. Writing a ‘1’ will set the corresponding "Add Request" bit; writing a ‘0’ has no impact. This enables the Host to set transmission requests for multiple Tx Buffers with one write to

TXBAR. TXBAR bits are set only for those Tx Buffers configured via TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.

Note: If an add request is applied for a Tx Buffer with pending transmission request (corresponding TXBRP bit already set), this add request is ignored. LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all rwh bits in this register.

TXBARi (i=0-3)

Tx Buffer Add Request i **($0082D0_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AR31	**AR30**	**AR29**	**AR28**	**AR27**	**AR26**	**AR25**	**AR24**	**AR23**	**AR22**	**AR21**	**AR20**	**AR19**	**AR18**	**AR17**	**AR16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR15	**AR14**	**AR13**	**AR12**	**AR11**	**AR10**	**AR9**	**AR8**	**AR7**	**AR6**	**AR5**	**AR4**	**AR3**	**AR2**	**AR1**	**AR0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
ARz (z=0-31)	z	rwh	**Add Request Tx Buffer z - AR** 0_B No transmission request added 1_B Transmission requested added

Tx Buffer Cancellation Request i

Each Tx Buffer has its own Cancellation Request bit. Writing a ‘1’ will set the corresponding Cancellation Request bit; writing a ‘0’ has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to TXBCR. TXBCR bits are set only for those Tx Buffers configured via TXBC. The bits remain set until the corresponding bit of TXBRP is reset.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all rwh bits in this register.

TXBCRi (i=0-3)

Tx Buffer Cancellation Request i **($0082D4_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CR31	**CR30**	**CR29**	**CR28**	**CR27**	**CR26**	**CR25**	**CR24**	**CR23**	**CR22**	**CR21**	**CR20**	**CR19**	**CR18**	**CR17**	**CR16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR15	**CR14**	**CR13**	**CR12**	**CR11**	**CR10**	**CR9**	**CR8**	**CR7**	**CR6**	**CR5**	**CR4**	**CR3**	**CR2**	**CR1**	**CR0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CRz (z=0-31)	z	rwh	**Cancellation Request Tx Buffer z - CR** 0_B No cancellation pending 1_B Cancellation pending

Tx Buffer Transmission Occurred i

Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR.

TXBTOi (i=0-3)
Tx Buffer Transmission Occurred i **($0082D8_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TO31	**TO30**	**TO29**	**TO28**	**TO27**	**TO26**	**TO25**	**TO24**	**TO23**	**TO22**	**TO21**	**TO20**	**TO19**	**TO18**	**TO17**	**TO16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TO15	**TO14**	**TO13**	**TO12**	**TO11**	**TO10**	**TO9**	**TO8**	**TO7**	**TO6**	**TO5**	**TO4**	**TO3**	**TO2**	**TO1**	**TO0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TOz (z=0-31)	z	rh	**Transmission Occurred Tx Buffer z - TO** 0_B No transmission occurred 1_B Transmission occurred

Tx Buffer Cancellation Finished i

Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding TXBRP bit is cleared after a cancellation was requested via TXBCR. In case the corresponding TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a '1' to the corresponding bit of register TXBAR.

TXBCFi (i=0-3)
Tx Buffer Cancellation Finished i **($0082DC_H$+i*400_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF31	**CF30**	**CF29**	**CF28**	**CF27**	**CF26**	**CF25**	**CF24**	**CF23**	**CF22**	**CF21**	**CF20**	**CF19**	**CF18**	**CF17**	**CF16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF15	**CF14**	**CF13**	**CF12**	**CF11**	**CF10**	**CF9**	**CF8**	**CF7**	**CF6**	**CF5**	**CF4**	**CF3**	**CF2**	**CF1**	**CF0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CFz (z=0-31)	z	rh	**Cancellation Finished Tx Buffer z - CF** 0_B No transmit buffer cancellation 1_B Transmit buffer cancellation finished

Tx Buffer Transmission Interrupt Enable i

Each Tx Buffer has its own Transmission Interrupt Enable bit.

TXBTIEi (i=0-3)

Tx Buffer Transmission Interrupt Enable i (0082E0$_H$+i*400$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIE31	**TIE30**	**TIE29**	**TIE28**	**TIE27**	**TIE26**	**TIE25**	**TIE24**	**TIE23**	**TIE22**	**TIE21**	**TIE20**	**TIE19**	**TIE18**	**TIE17**	**TIE16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIE15	**TIE14**	**TIE13**	**TIE12**	**TIE11**	**TIE10**	**TIE9**	**TIE8**	**TIE7**	**TIE6**	**TIE5**	**TIE4**	**TIE3**	**TIE2**	**TIE1**	**TIE0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TIEz (z=0-31)	z	rw	**Transmission Interrupt Enable Tx Buffer z - TIE** 0_B Transmission interrupt disabled 1_B Transmission interrupt enable

Tx Buffer Cancellation Finished Interrupt Enable i

Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit.

TXBCIEi (i=0-3)

Tx Buffer Cancellation Finished Interrupt Enable i(0082E4$_H$+i*400$_H$) **Application Reset Value: 0000 0000$_H$**

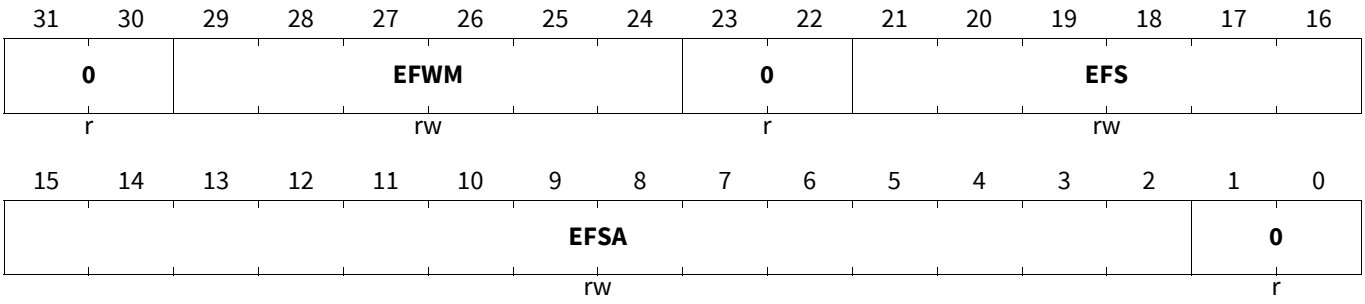
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFIE31	**CFIE30**	**CFIE29**	**CFIE28**	**CFIE27**	**CFIE26**	**CFIE25**	**CFIE24**	**CFIE23**	**CFIE22**	**CFIE21**	**CFIE20**	**CFIE19**	**CFIE18**	**CFIE17**	**CFIE16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFIE15	**CFIE14**	**CFIE13**	**CFIE12**	**CFIE11**	**CFIE10**	**CFIE9**	**CFIE8**	**CFIE7**	**CFIE6**	**CFIE5**	**CFIE4**	**CFIE3**	**CFIE2**	**CFIE1**	**CFIE0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CFIEz (z=0-31)	z	rw	**Cancellation Finished Interrupt Enable Tx Buffer z - CFIE** 0_B Cancellation finished interrupt disabled 1_B Cancellation finished interrupt enabled

Tx Event FIFO Configuration i

TXEFCi (i=0-3)

Tx Event FIFO Configuration i **(0082F0$_H$+i*400$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
EFSA	15:2	rw	**Event FIFO Start Address** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Start address of Tx Event FIFO in Message RAM (32-bit word address).
EFS	21:16	rw	**Event FIFO Size** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. The Tx Event FIFO elements are indexed from 0 to EFS - 1 00_H Tx Event FIFO disabled 01_H 1 Tx Event FIFO elements … 20_H 32 Tx Event FIFO elements **others**, 32 Tx Event FIFO elements
EFWM	29:24	rw	**Event FIFO Watermark** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 01_H Level for Tx Event FIFO watermark interrupt (IR.TEFW) … 20_H Level for Tx Event FIFO watermark interrupt (IR.TEFW) **others**, Watermark interrupt disabled
0	1:0, 23:22, 31:30	r	**Reserved** Shall read 0, shall be written with 0.

Tx Event FIFO Status i

TXEFSi (i=0-3)

Tx Event FIFO Status i **($0082F4_H$+i*400_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						TEFL	EFF	0			EFPI				
r						rh	rh	r			rh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			EFGI					0		EFFL					
r			rh					r		rh					

Field	Bits	Type	Description
EFFL	5:0	rh	**Event FIFO Fill Level** Number of elements stored in Tx Event FIFO, range 0 to 32.
EFGI	12:8	rh	**Event FIFO Get Index** Tx Event FIFO read index pointer, range 0 to 31.
EFPI	20:16	rh	**Event FIFO Put Index** Tx Event FIFO write index pointer, range 0 to 31.
EFF	24	rh	**Event FIFO Full** 0_B Tx Event FIFO not full 1_B Tx Event FIFO full
TEFL	25	rh	**Tx Event FIFO Element Lost** This bit is a copy of interrupt flag IR.TEFL. When IR.TEFL is reset, this bit is also reset. 0_B No Tx Event FIFO element lost 1_B Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
0	7:6, 15:13, 23:21, 31:26	r	**Reserved** Shall read 0, shall be written with 0.

Tx Event FIFO Acknowledge i

TXEFAi (i=0-3)

Tx Event FIFO Acknowledge i **($0082F8_H$+i*400_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													EFAI		
r													rw		

Field	Bits	Type	Description
EFAI	4:0	rw	**Event FIFO Acknowledge Index** After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index TXEFS.EFGI to EFAI + 1 and update the FIFO 0 Fill Level TXEFS.EFFL.
0	31:5	r	**Reserved** Shall read 0, shall be written with 0.

TT Trigger Memory Configuration 0

The TTCAN function exists only on CAN0 node 0.

TTTMC0
TT Trigger Memory Configuration 0 (008300_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									**TME**						
r									rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMSA														**0**	
rw														r	

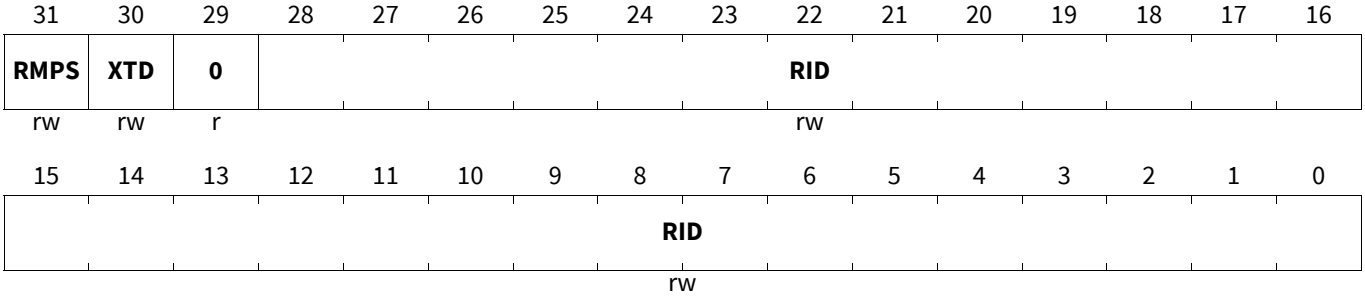
Field	Bits	Type	Description
TMSA	15:2	rw	**Trigger Memory Start Address** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Start address of Trigger Memory in Message RAM.
TME	22:16	rw	**Trigger Memory Elements** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 00_H No Trigger Memory 01_H 1 Trigger Memory element … 40_H 64 Trigger Memory element **others**, 64 Trigger Memory elements
0	1:0, 31:23	r	**Reserved** Shall read 0, shall be written with 0.

TT Reference Message Configuration 0

The TTCAN function exists only on CAN0 node 0. For details about handling of reference messages.

TTRMC0

TT Reference Message Configuration 0 (008304$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RID	28:0	rw	**Reference Identifier** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Identifier transmitted with reference message and used for reference message filtering. Standard or extended reference identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
XTD	30	rw	**Extended Identifier** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B 11-bit standard identifier 1_B 29-bit extended identifier
RMPS	31	rw	**Reference Message Payload Select** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Ignored in case of time slaves. 0_B Reference message has no additional payload 1_B The following elements are taken from Tx Buffer 0: Message Marker MM, Event FIFO Control EFC, Data Length Code DLC, Data Bytes DB (Level 1: bytes 2-8, Level 0,2: bytes 5-8)
0	29	r	**Reserved** Shall read 0, shall be written with 0.

TT Operation Configuration 0

The TTCAN function exists only on CAN0 node 0.

TTOCF0

TT Operation Configuration 0 **(008308$_H$)** **Application Reset Value: 0001 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					EVTP	ECC	EGTF	AWL							
r					rw	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EECS	IRTO							LDSDL			TM	GEN	0	OM	
rw	rw							rw			rw	rw	r	rw	

Field	Bits	Type	Description
OM	1:0	rw	**Operation Mode** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 00_B Event-driven CAN communication, default 01_B TTCAN level 1 10_B TTCAN level 2 11_B TTCAN level 0
GEN	3	rw	**Gap Enable** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Strictly time-triggered operation 1_B External event-synchronized time-triggered operation
TM	4	rw	**Time Master** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Time Master function disabled 1_B Potential Time Master
LDSDL	7:5	rw	**LD of Synchronization Deviation Limit** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. The Synchronization Deviation Limit SDL is configured by its dual logarithm LDSDL with SDL = $2^{(LDSDL+5)}$. It should not exceed the clock tolerance given by the CAN bit timing configuration. LD of Synchronization Deviation Limit (SDL ≤ 32…4096)
IRTO	14:8	rw	**Initial Reference Trigger Offset** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Positive offset, range from 0 to 127

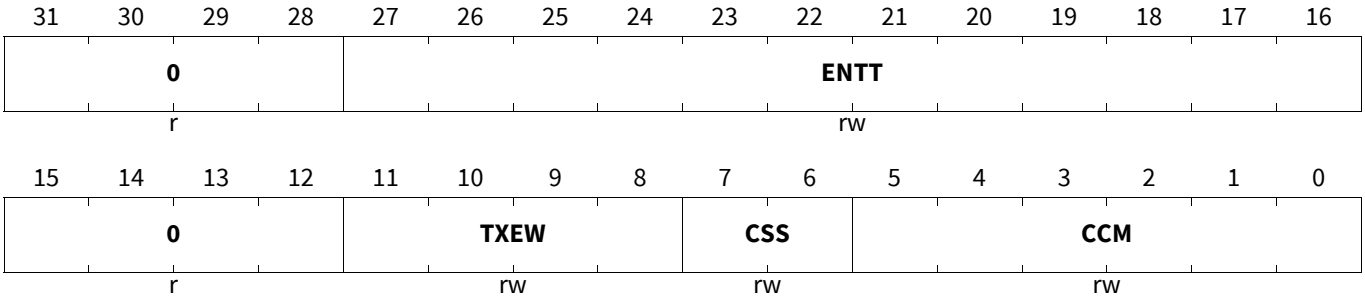
Field	Bits	Type	Description
EECS	15	rw	**Enable External Clock Synchronization** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. If enabled, TUR configuration (TURCF.NCL only) may be updated during TTCAN operation. 0_B External clock synchronization in TTCAN Level 0,2 disabled 1_B External clock synchronization in TTCAN Level 0,2 enabled
AWL	23:16	rw	**Application Watchdog Limit** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. The application watchdog can be disabled by programming AWL to 0x00. Maximum time after which the application has to serve the application watchdog. The application watchdog is incremented once each 256 NTUs.
EGTF	24	rw	**Enable Global Time Filtering** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Global time filtering in TTCAN Level 0,2 is disabled 1_B Global time filtering in TTCAN Level 0,2 is enabled
ECC	25	rw	**Enable Clock Calibration** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Automatic clock calibration in TTCAN Level 0,2 is disabled 1_B Automatic clock calibration in TTCAN Level 0,2 is enabled
EVTP	26	rw	**Event Trigger Polarity** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0_B Rising edge trigger 1_B Falling edge trigger
0	2, 31:27	r	**Reserved** Shall read 0, shall be written with 0.

TT Matrix Limits 0

The TTCAN function exists only on CAN0 node 0.

TTMLM0

TT Matrix Limits 0 **($00830C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CCM	5:0	rw	**Cycle Count Max** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. *Note: ISO 11898-4, 5.2.1 requires, that only the listed cycle count values are configured. Other values are possible but may lead to inconsistent matrix cycles.* 00_H 1 Basic Cycle per Matrix Cycle 01_H 2 Basic Cycles per Matrix Cycle 03_H 4 Basic Cycles per Matrix Cycle 07_H 8 Basic Cycles per Matrix Cycle $0F_H$ 16 Basic Cycles per Matrix Cycle $1F_H$ 32 Basic Cycles per Matrix Cycle $3F_H$ 64 Basic Cycles per Matrix Cycle **others**, Reserved
CSS	7:6	rw	**Cycle Start Synchronization** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Enables sync pulse output at start of cycle. 00_B No sync pulse 01_B Sync pulse at start of basic cycle 10_B Sync pulse at start of matrix cycle 11_B Reserved
TXEW	11:8	rw	**Tx Enable Window** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Length of Tx enable window, 1-16 NTU cycles
ENTT	27:16	rw	**Expected Number of Tx Triggers** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Expected number of Tx Triggers in one Matrix Cycle
0	15:12, 31:28	r	**Reserved** Shall read 0, shall be written with 0.

TUR Configuration 0

The TTCAN function exists only on CAN0 node 0.

The length of the NTU is given by: NTU = CAN Clock Period x NC / DC

NC is an 18-bit value. Its high part, NCH[17:16] is hard wired to 0b01. Therefore the range of NC is 10000_H…$1FFFF_H$. The value configured by NCL is the initial value for TURNA.NAV[15:0]. DC is set to 1000_H by hardware reset and it may not be written to 0000_H.

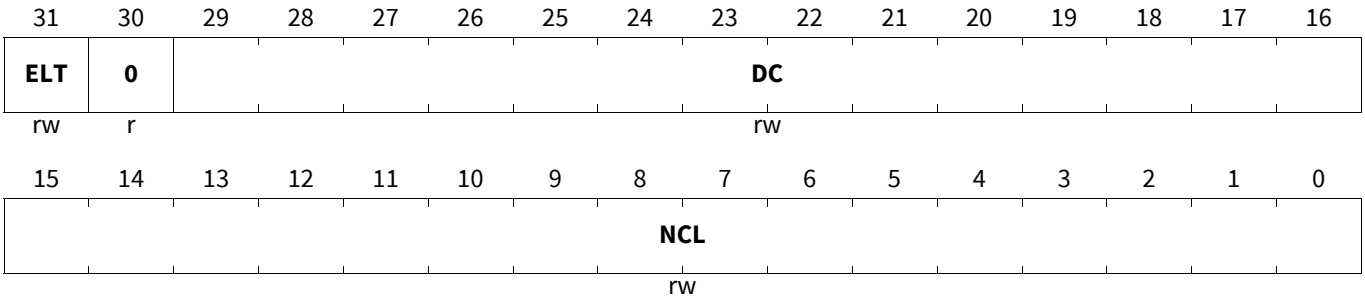
Level 1: NC ≥ 4 x DC and NTU = CAN bit time

Level 0,2: NC ≥ 8 x DC

The actual value of TUR may be changed by the clock drift compensation function of TTCAN Level 0 and Level 2 in order to adjust the node's local view of the NTU to the time master's view of the NTU. DC will not be changed

by the automatic drift compensation, TURNA.NAV may be adjusted around NC in the range of the Synchronisation Deviation Limit given by TTOCF.LDSDL. NC and DC should be programmed to the largest suitable values in order to allow the best computational accuracy for the drift compensation process.

TURCF0
TUR Configuration 0 **(008310_H)** **Application Reset Value: $1000\ 0000_H$**



Field	Bits	Type	Description
NCL	15:0	rw	**Numerator Configuration Low** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. Write access to the TUR Numerator Configuration Low is only possible during configuration with TURCF.ELT = ‘0’ or if TTOCF.EECS (external clock synchronization enabled) is set. When a new value for NCL is written outside TT Configuration Mode, the new value takes effect when TTOST.WECS is cleared to ‘0’. NCL is locked TTOST.WECS is ‘1’. *Note: If NC < 7 x DC in TTCAN Level 1, then it is required that subsequent time marks in the Trigger Memory must differ by at least 2 NTU.* 0000_HNumerator Configuration Low … $FFFF_H$Numerator Configuration Low
DC	29:16	rw	**Denominator Configuration** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. 0000_HIllegal value 0001_HDenominator Configuration … $3FFF_H$Denominator Configuration

Field	Bits	Type	Description
ELT	31	rw	**Enable Local Time** This bitfield is CCE and INIT protected. Writes will only have effect, if both bits are set. *Note: Local time is started by setting ELT. It remains active until ELT is reset or until the next hardware reset. TURCF.DC is locked when TURCF.ELT = '1'. If ELT is written to '0', the readable value will stay at '1' until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register remains locked.* 0_B Local time is stopped, default 1_B Local time is enabled
0	30	r	**Reserved** Shall read 0, shall be written with 0.

TT Operation Control 0

The TTCAN function exists only on CAN0 node 0.

TTOCN0

TT Operation Control 0 **(008314_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCKC	**0**	**ESCN**	**NIG**	**TMG**	**FGP**	**GCS**	**TTIE**	**TMC**		**RTIE**	**SWS**		**SWP**	**ECS**	**SGT**
rh	r	rw	rwh	rw	rwh	rw	rw	rw		rw	rw		rw	rwh	rwh

Field	Bits	Type	Description
SGT	0	rwh	**Set Global time** Writing a '1' to SGT sets TTOST.WGDT if the node is the actual Time Master. SGT is reset after one Host clock period. The global time preset takes effect when the node transmits the next reference message with the Master_Ref_Mark modified by the preset value written to TTGTP.
ECS	1	rwh	**External Clock Synchronization** Writing a '1' to ECS sets TTOST.WECS if the node is the actual Time Master. ECS is reset after one Host clock period. The external clock synchronization takes effect at the start of the next basic cycle.
SWP	2	rw	**Stop Watch Polarity** 0_B Rising edge trigger 1_B Falling edge trigger

Field	Bits	Type	Description
SWS	4:3	rw	**Stop Watch Source** 00_B Stop Watch disabled 01_B Actual value of cycle time is copied to TTCPT.SWV 10_B Actual value of local time is copied to TTCPT.SWV 11_B Actual value of global time is copied to TTCPT.SWV
RTIE	5	rw	**Register Time Mark Interrupt Pulse Enable** Register time mark interrupts are configured by register TTTMK. A register time mark interrupt pulse with the length of one TTCAN clock period is generated when the time referenced by TTOCN.TMC (cycle, local, or global) equals TTTMK.TM, independent of the synchronization state. 0_B Register Time Mark Interrupt output disabled 1_B Register Time Mark Interrupt output enabled
TMC	7:6	rw	**Register Time Mark Compare** *Note: When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = "00", then reconfigure TTTMK, and finally set TMC to the intended time reference.* 00_B No Register Time Mark Interrupt generated 01_B Register Time Mark Interrupt if Time Mark = cycle time 10_B Register Time Mark Interrupt if Time Mark = local time 11_B Register Time Mark Interrupt if Time Mark = global time
TTIE	8	rw	**Trigger Time Mark Interrupt Pulse Enable** External time mark events are configured by trigger memory element TMEX. A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the M_CAN is in synchronization state In_Schedule or In_Gap. 0_B Trigger Time Mark Interrupt output disabled 1_B Trigger Time Mark Interrupt output enabled
GCS	9	rw	**Gap Control Select** 0_B Gap control independent from event trigger 1_B Gap control by the event trigger
FGP	10	rwh	**Finish Gap** Set by the CPU, reset by each reference message 0_B No reference message requested 1_B Application requested start of reference message
TMG	11	rw	**Time Mark Gap** 0_B Reset by each reference message 1_B Next reference message started when Register Time Mark interrupt TTIR.RTMI is activated

Field	Bits	Type	Description
NIG	12	rwh	**Next is Gap** This bit can only be set when the M_CAN is the actual Time Master and when it is configured for external event-synchronized time-triggered operation (TTOCF.GEN = '1') 0_B No action, reset by reception of any reference message 1_B Transmit next reference message with Next_is_Gap = '1'
ESCN	13	rw	**External Synchronization Control** If enabled the M_CAN synchronizes its cycle time phase to an external event signalled by a rising edge at the event trigger. 0_B External synchronization disabled 1_B External synchronization enabled
LCKC	15	rh	**TT Operation Control Register Locked** Set by a write access to register TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain. 0_B Write access to TTOCN enabled 1_B Write access to TTOCN locked
0	14, 31:16	r	**Reserved** Shall read 0, shall be written with 0.

TT Global Time Preset 0

If TTOST.WGDT is set, the next reference message will be transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = '1', presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a reference message with Disc_Bit = '1' becomes valid or if the node is not the current Time Master. TP is locked while TTOST.WGTD = '1' after setting TTOCN.SGT until the reference message with Disc_Bit = '1' becomes valid or until the node is no longer the current Time Master.

The TTCAN function exists only on CAN0 node 0.

TTGTP0

TT Global Time Preset 0 **(008318_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTP															
rwh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TP															
rwh															

Field	Bits	Type	Description
TP	15:0	rwh	**Time Preset** CTP is write-protected while TTOCN.ESCN or TTOST.SPL are set. 0000_H Next Master Reference Mark = Master Reference Mark + TP … $7FFF_H$ Next Master Reference Mark = Master Reference Mark + TP 8000_H Reserved 8001_H Next Master Reference Mark = Master Reference Mark - (0x10000 - TP) … $FFFF_H$ Next Master Reference Mark = Master Reference Mark - (0x10000 - TP)
CTP	31:16	rwh	**Cycle Time Target Phase** CTP is write-protected while TTOCN.ESCN or TTOST.SPL are set. 0000_H Defines target value of cycle time when a rising edge of the event trigger is expected … $FFFF_H$ Defines target value of cycle time when a rising edge of the event trigger is expected

TT Time Mark 0

A time mark interrupt (TTIR.RTMI = ‘1’) is generated when the time base indicated by TTOCN.TMC (cycle time, local time, or global time) has the same value as TM.

The TTCAN function exists only on CAN0 node 0.

TTTMK0

TT Time Mark 0 **($00831C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCKM				**0**					**TICC**						
rh				r					rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TM															
rw															

Field	Bits	Type	Description
TM	15:0	rw	**Time Mark** *Note: When using byte access to register TTTMK it is recommended to first disable the time mark compare function (TTOCN.TMC = “00”) to avoid compares on inconsistent register values.* 0000_H Time Mark … $FFFF_H$ Time Mark

Field	Bits	Type	Description
TICC	22:16	rw	**Time Mark Cycle Code** Cycle count for which the time mark is valid. 00_H valid for all cycles 01_H valid for all cycles 02_H valid every second cycle at cycle count mod2 = 0 03_H valid every second cycle at cycle count mod2 = 1 04_H valid every fourth cycle at cycle count mod4 = 0 … 07_H valid every fourth cycle at cycle count mod4 = 3 08_H valid every eighth cycle at cycle count mod8 = 0 … $0F_H$ valid every eighth cycle at cycle count mod8 = 7 10_H valid every sixteenth cycle at cycle count mod16 = 0 … $1F_H$ valid every sixteenth cycle at cycle count mod16 = 15 20_H valid every thirty-second cycle at cycle count mod32 = 0 … $3F_H$ valid every thirty-second cycle at cycle count mod32 = 31 40_H valid every sixty-fourth cycle at cycle count mod64 = 0 … $7F_H$ valid every sixty-fourth cycle at cycle count mod64 = 63
LCKM	31	rh	**TT Time Mark Register Locked** Always set by a write access to registers TTOCN. Set by write access to register TTTMK when TTOCN.TMC ≠ "00”. Reset when the registers have been synchronized into the CAN clock domain. 0_B Write access to TTTMK enabled 1_B Write access to TTTMK locked
0	30:23	r	**Reserved** Shall read 0, shall be written with 0.

TT Interrupt Register 0

The interrupt register for TTCAN related events.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all rwh bits in this register.

The TTCAN function exists only on CAN0 node 0.

TTIR0

TT Interrupt Register 0 **(008320$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													CER	AW	WT
r													rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWT	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SBC	0	rwh	**Start of Basic Cycle** 0_B No Basic Cycle started since bit has been reset 1_B Basic Cycle started
SMC	1	rwh	**Start of Matrix Cycle** 0_B No Matrix Cycle started since bit has been reset 1_B Matrix Cycle started
CSM	2	rwh	**Change of Synchronization Mode** 0_B No change in master to slave relation or schedule synchronization 1_B Master to slave relation or schedule synchronization changed, also set when TTOST.SPL is reset
SOG	3	rwh	**Start of Gap** 0_B No reference message seen with Next_is_Gap bit set 1_B Reference message with Next_is_Gap bit set becomes valid
RTMI	4	rwh	**Register Time Mark Interrupt** Set when time referenced by TTOCN.TMC (cycle, local, or global) equals TTTMK.TM, independent of the synchronization state. 0_B Time mark not reached 1_B Time mark reached
TTMI	5	rwh	**Trigger Time Mark Event Internal** Internal time mark events are configured by trigger memory element TMIN. Set when the trigger memory element becomes active, and the M_CAN is in synchronization state In_Gap or In_Schedule. 0_B Time mark not reached 1_B Time mark reached (Level 0: cycle time TTOCF.IRTO · 0x200)
SWE	6	rwh	**Stop Watch Polarity** 0_B No rising/falling edge at stop watch trigger detected 1_B Rising/falling edge at stop watch trigger detected

Field	Bits	Type	Description
GTW	7	rwh	**Global Time Wrap** 0_B No global time wrap occurred 1_B Global time wrap from 0xFFFF to 0x0000 occurred
GTD	8	rwh	**Global Time Discontinuity** 0_B No discontinuity of global time 1_B Discontinuity of global time
GTE	9	rwh	**Global Time Error** Synchronization deviation SD exceeds limit specified by TTOCF.LDSDL, TTCAN Level 0,2 only. 0_B Synchronization deviation within limit 1_B Synchronization deviation exceeded limit
TXU	10	rwh	**Tx Count Underflow** 0_B Number of Tx Trigger as expected 1_B Less Tx trigger than expected in one matrix cycle
TXO	11	rwh	**Tx Count Overflow** 0_B Number of Tx Trigger as expected 1_B More Tx trigger than expected in one matrix cycle
SE1	12	rwh	**Scheduling Error 1** 0_B No scheduling error 1 1_B Scheduling error 1 occurred
SE2	13	rwh	**Scheduling Error 2** 0_B No scheduling error 2 1_B Scheduling error 2 occurred
ELC	14	rwh	**Error Level Changed** Not set when error level changed during initialization. 0_B No change in error level 1_B Error level changed
IWT	15	rwh	**Initialization Watch Trigger** The initialization is restarted by resetting IWT. 0_B No missing reference message during system startup 1_B No system startup due to missing reference message
WT	16	rwh	**Watch Trigger** 0_B No missing reference message 1_B Missing reference message (Level 0: cycle time 0xFF00)
AW	17	rwh	**Application Watchdog** 0_B Application watchdog served in time 1_B Application watchdog not served in time

Field	Bits	Type	Description
CER	18	rwh	**Configuration Error** Trigger out of order. 0_B No error found in trigger list 1_B Error found in trigger list
0	31:19	r	**Reserved** Shall read 0, shall be written with 0.

TT Interrupt Enable 0

The settings in the TT Interrupt Enable register determine which status changes in the TT Interrupt Register will result in an interrupt.

R = Read, W = Write; -n = value after reset

TTIE0
TT Interrupt Enable 0 **(008324_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													CERE	AWE	WTE
r													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SBCE	0	rw	**Start of Basic Cycle Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
SMCE	1	rw	**Start of Matrix Cycle Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
CSME	2	rw	**Change of Synchronization Mode Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
SOGE	3	rw	**Start of Gap Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
RTMIE	4	rw	**Register Time Mark Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled

Field	Bits	Type	Description
TTMIE	5	rw	**Trigger Time Mark Event Internal Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
SWEE	6	rw	**Stop Watch Polarity Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
GTWE	7	rw	**Global Time Wrap Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
GTDE	8	rw	**Global Time Discontinuity Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
GTEE	9	rw	**Global Time Error Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
TXUE	10	rw	**Tx Count Underflow Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
TXOE	11	rw	**Tx Count Overflow Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
SE1E	12	rw	**Scheduling Error 1 Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
SE2E	13	rw	**Scheduling Error 2 Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
ELCE	14	rw	**Change Error Level Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
IWTE	15	rw	**Initialization Watch Trigger Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled

Field	Bits	Type	Description
WTE	16	rw	**Watch Trigger Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
AWE	17	rw	**Application Watchdog Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
CERE	18	rw	**Configuration Error Interrupt Enable** 0_B TT interrupt disabled 1_B TT interrupt enabled
0	31:19	r	**Reserved** Shall read 0, shall be written with 0.

TT Operation Status 0

R = Read, P = Protected write; -n = value after reset

The TTCAN function exists only on CAN0 node 0.

TTOST0

TT Operation Status 0 **($00832C_H$)** **Application Reset Value: $2000\ 0080_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPL	**WECS**	**AWE**	**WFE**	**GSI**	**TMP**			**GFI**	**WGTD**	**0**					
rh	rh	rh	rh	rh	rh			rh	rh	r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO								**QCS**	**QGTP**	**SYS**		**MS**		**EL**	
rh								rh	rh	rh		rh		rh	

Field	Bits	Type	Description
EL	1:0	rh	**Error Level** 00_B Severity 0 - No Error 01_B Severity 1 - Warning 10_B Severity 2 - Error 11_B Severity 3 - Severe Error
MS	3:2	rh	**Master State** 00_B Master_Off, no master properties relevant 01_B Operating as Time Slave 10_B Operating as Backup Time Master 11_B Operating as current Time Master

Field	Bits	Type	Description
SYS	5:4	rh	**Synchronization State** 00_B Out of Synchronization 01_B Synchronizing to TTCAN communication 10_B Schedule suspended by Gap (In_Gap) 11_B Synchronized to schedule (In_Schedule)
QGTP	6	rh	**Quality of Global Time Phase** Only relevant in TTCAN Level 0 and Level 2, otherwise fixed to '0'. 0_B Global time not valid 1_B Global time in phase with Time Master
QCS	7	rh	**Quality of Clock Speed** Only relevant in TTCAN Level 0 and Level 2, otherwise fixed to '1'. 0_B Local clock speed not synchronized to Time Master clock speed 1_B Synchronization Deviation ≤ SDL
RTO	15:8	rh	**Reference Trigger Offset** The Reference Trigger Offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of -127 is reached. In case the M_CAN becomes Time Master (MS[1:0] = "11"), the reset of RTO is delayed due to synchronization between Host and CAN clock domain. For time slaves the value configured by TTOCF.IRTO is read. Actual Reference Trigger offset value
WGTD	22	rh	**Wait for Global Time Discontinuity** 0_B No global time preset pending 1_B Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a reference message with Disc_Bit = '1' or after it received a reference message.
GFI	23	rh	**Gap Finished Indicator** Set when the CPU writes TTOCN.FGP, or by a time mark interrupt if TMG = '1', or via event trigger input if TTOCN.GCS = '1'. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message. 0_B Reset at the end of each reference message 1_B Gap finished by M_CAN
TMP	26:24	rh	**Time Master Priority** 000_B Priority of actual Time Master … 111_B Priority of actual Time Master
GSI	27	rh	**Gap Started Indicator** 0_B No Gap in schedule, reset by each reference message and for all time slaves 1_B Gap time after Basic Cycle has started
WFE	28	rh	**Wait for Event** 0_B No Gap announced, reset by a reference message with Next_is_Gap = '0' 1_B Reference message with Next_is_Gap = '1' received

Field	Bits	Type	Description
AWE	29	rh	**Application Watchdog Event** The application watchdog is served by reading TTOST. When the watchdog is not served in time, bit AWE is set, all TTCAN communication is stopped, and the M_CAN is set into Bus Monitoring Mode. 0_B Application Watchdog served in time 1_B Failed to serve Application Watchdog in time
WECS	30	rh	**Wait for External Clock Synchronization** 0_B No external clock synchronization pending 1_B Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.
SPL	31	rh	**Schedule Phase Lock** The bit is valid only when external synchronization is enabled (TTOCN.ESCN = '1'). In this case it signals that the difference between cycle time configured by TTGTP.CTP and the cycle time at the rising edge at the event trigger is less or equal 9 NTU. 0_B Phase outside range 1_B Phase inside range
0	21:16	r	**Reserved** **Shall read 0, shall be written with 0.**

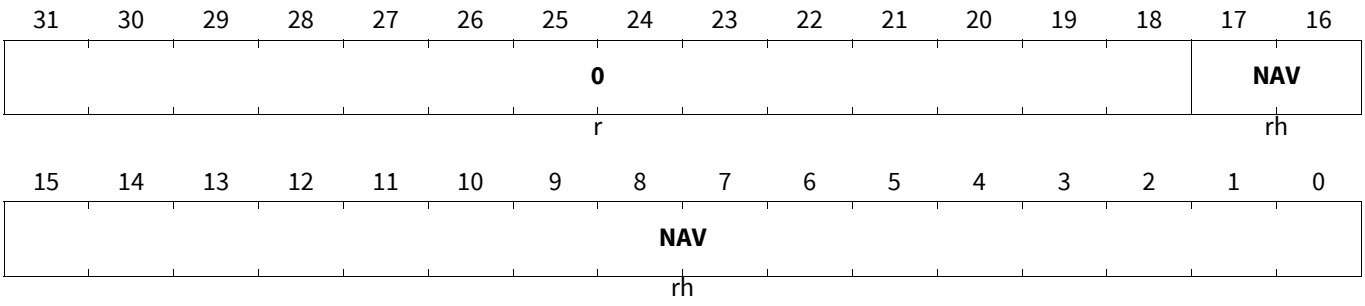
TUR Numerator Actual 0

The TTCAN function exists only on CAN0 node 0.

There is no drift compensation in TTCAN Level 1 (NAV = NC). In TTCAN Level 0 and Level 2, the drift between the node's local clock and the time master's local clock is calculated. The drift is compensated when the Synchronisation Deviation (difference between NC and the calculated NAV) is not more than $2^{(TTOCF.LDSDL + 5)}$. With TTOCF.LDSDL ≤ 7, this results in a maximum range for NAV of (NC - 0x1000) ≤ NAV ≤ (NC + 0x1000).

TURNA0

TUR Numerator Actual 0 **(008330_H)** **Application Reset Value: $0001\ 0000_H$**



Field	Bits	Type	Description
NAV	17:0	rh	**Numerator Actual Value** $0F000_H$ Actual numerator value … $20FFF_H$ Actual numerator value **others**, Illegal value

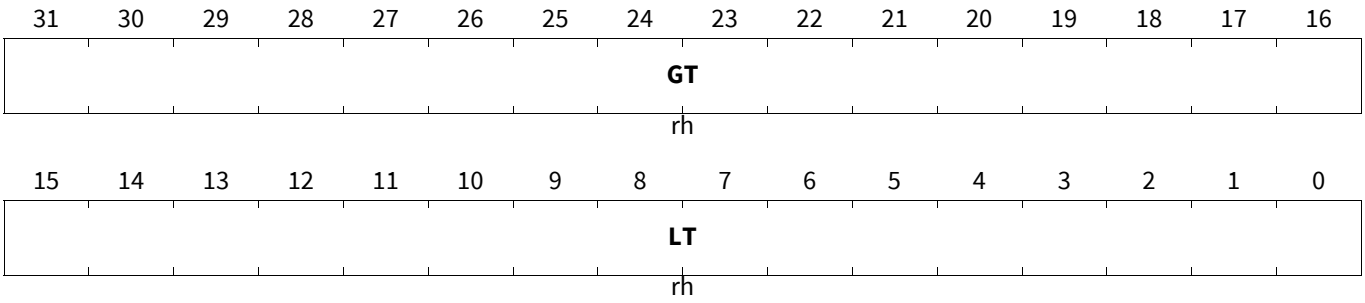
Field	Bits	Type	Description
0	31:18	r	**Reserved** Shall read 0, shall be written with 0.

TT Local & Global Time 0

The TTCAN function exists only on CAN0 node 0.

TTLGT0

TT Local & Global Time 0 **(008334$_H$)** **Application Reset Value: 0000 0000$_H$**



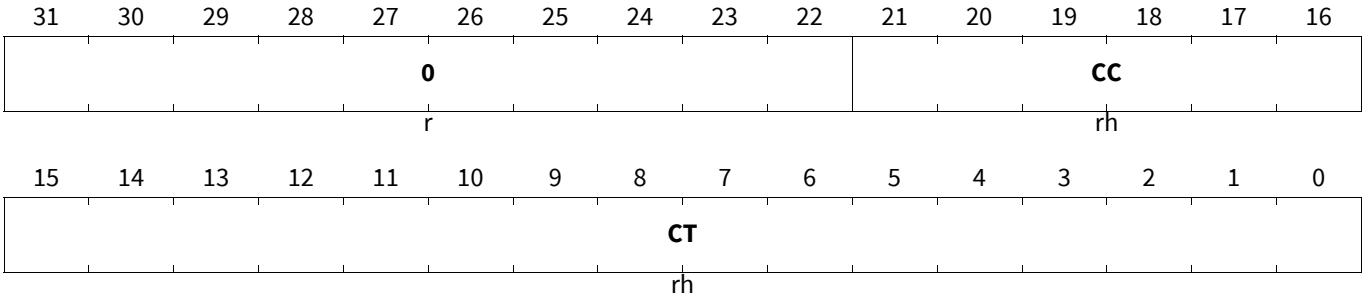
Field	Bits	Type	Description
LT	15:0	rh	**Local Time** Non-fractional part of local time, incremented once each local NTU. Local time value of TTCAN node
GT	31:16	rh	**Global Time** Non-fractional part of the sum of the node's local time and its local offset. Global time value of TTCAN network

TT Cycle Time & Count 0

The TTCAN function exists only on CAN0 node 0.

TTCTC0

TT Cycle Time & Count 0 **(008338$_H$)** **Application Reset Value: 003F 0000$_H$**



Field	Bits	Type	Description
CT	15:0	rh	**Cycle Time** Non-fractional part of the difference of the node's local time and Ref_Mark. Cycle time value of TTCAN Basic Cycle

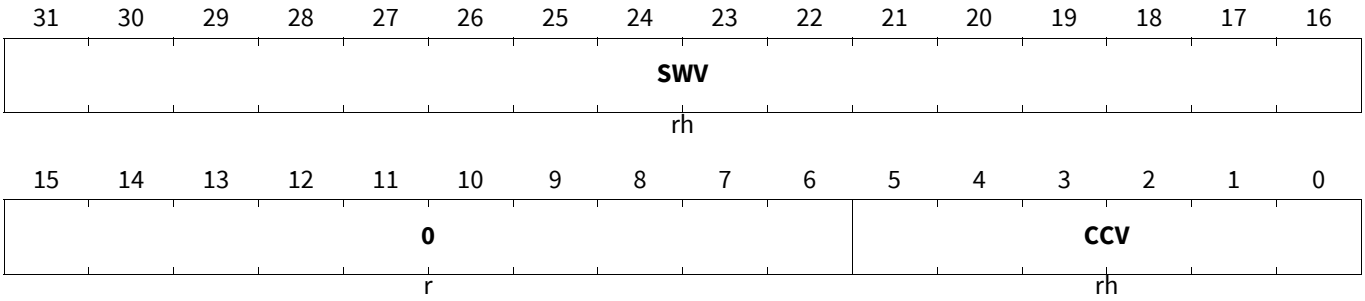
Field	Bits	Type	Description
CC	21:16	rh	**Cycle Count** Number of actual Basic Cycle in the System Matrix
0	31:22	r	**Reserved** Shall read 0, shall be written with 0.

TT Capture Time 0

The TTCAN function exists only on CAN0 node 0.

TTCPT0

TT Capture Time 0 **(00833C$_H$)** **Application Reset Value: 0000 0000$_H$**



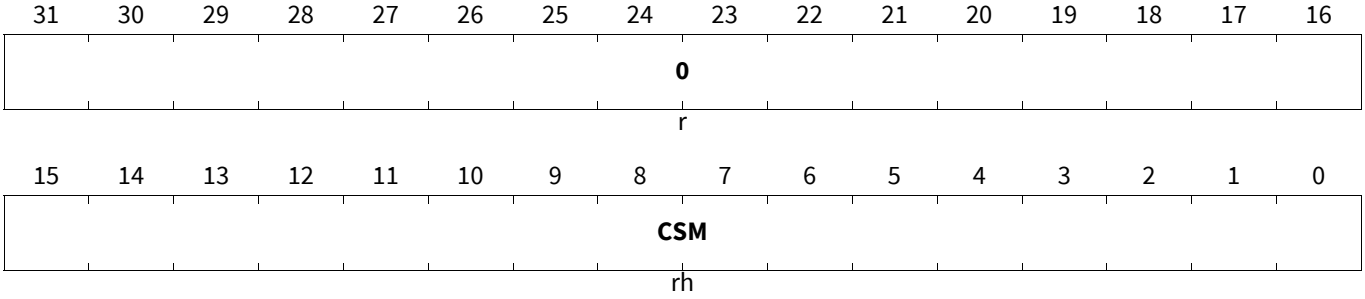
Field	Bits	Type	Description
CCV	5:0	rh	**Cycle Count Value** Cycle count value captured together with SWV. Captured cycle count value
SWV	31:16	rh	**Stop Watch Value** On a rising/falling edge (as configured via TTOCN.SWP) at the Stop Watch Trigger, when TTOCN.SWS is ≠ “00” and TTIR.SWE is ‘0’, the actual time value as selected by TTOCN.SWS (cycle, local, global) is copied to SWV and TTIR.SWE will be set to ‘1’. Capturing of the next stop watch value is enabled by resetting TTIR.SWE. Captured Stop Watch value
0	15:6	r	**Reserved** Shall read 0, shall be written with 0.

TT Cycle Sync Mark 0

The TTCAN function exists only on CAN0 node 0.

TTCSM0

TT Cycle Sync Mark 0 **(008340_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CSM	15:0	rh	**Cycle Sync Mark** The Cycle Sync Mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid. Captured cycle time
0	31:16	r	**Reserved** Shall read 0, shall be written with 0.

40.4.6 Message RAM

M_CAN

40.4.6.1 Message RAM Configuration

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via RXESCi.F0DS, RXESCi.F1DS, RXESCi.RBDS, and TXESCi.TBDS.

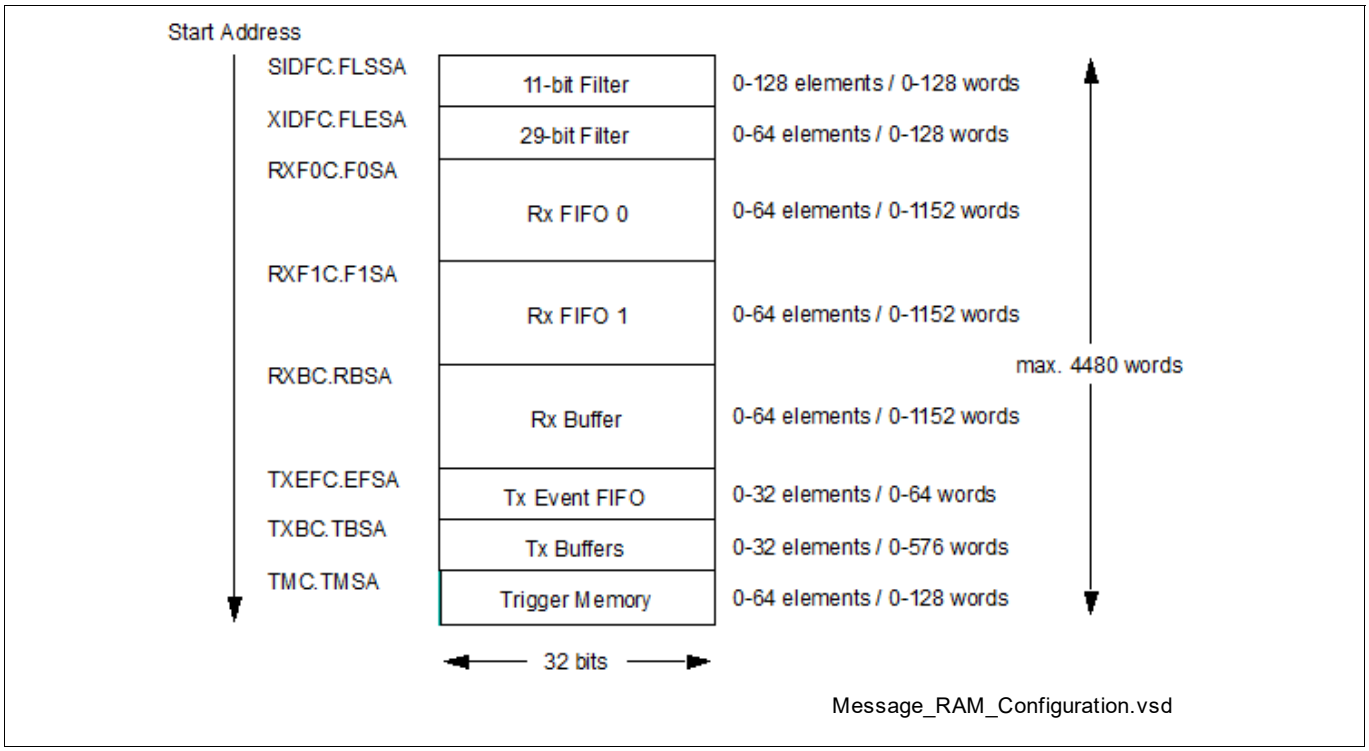


Figure 605 Message RAM Configuration

When the M_CAN addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

40.4.6.2 Rx Buffer and FIFO Element

Up to 64 Rx Buffers and two Rx FIFOs can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in **Table 382** below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register RXESCi.

Table 382 Message Layout - Rx Buffer and FIFO Element

	3 1	2 4	2 3	1 6	1 5	8	7	0
R0	ESI	XTD	RTR	ID[28:0]				
R1	ANMF	FIDX[6:0]		0	FDF	BRS	DLC[3:0]	RXTS[15:0]
R2	DB3[7:0]		DB2[7:0]		DB1[7:0]		DB0[7:0]	
R3	DB7[7:0]		DB6[7:0]		DB5[7:0]		DB4[7:0]	
…	…		…		…		…	
Rn	DBm[7:0]		DBm-1[7:0]		DBm-2[7:0]		DBm-3[7:0]	

Register 0

See Message layout, **Table 382**.

See Message layout.

RxMsgk_R0 (k=0-63)

Register 0 **($000000_H + k*48_H$)** **Application Reset Value: XXXX XXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESI	**XTD**	**RTR**	**ID**												
rwh	rwh	rwh	rwh												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID															
rwh															

Field	Bits	Type	Description
ID	28:0	rwh	**Identifier** Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].

Field	Bits	Type	Description
RTR	29	rwh	**Remote Transmission Request** Signals to the Host whether the received frame is a data frame or a remote frame. *Note: There are no remote frames in CAN FD format. In case a CAN FD frame was received (FDF = 1'), bit RTR reflects the state of the reserved bit r1.* 0_B Received frame is a data frame 1_B Received frame is a remote frame
XTD	30	rwh	**Extended Identifier** Signals to the Host whether the received frame has a standard or extended identifier. 0_B 11-bit standard identifier 1_B 29-bit extended identifier
ESI	31	rwh	**Error State Indicator** 0_B Transmitting node is error active 1_B Transmitting node is error passive

Register 1

See Message layout, **Table 382**.

See Message layout.

RxMsgk_R1 (k=0-63)

Register 1 **(000004_H+k*48_H)** **Application Reset Value: XXXX $XXXX_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ANMF	**FIDX**							**0**		**FDF**	**BRS**	**DLC**			
rwh	rwh							rwh		rwh	rwh	rwh			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXTS															
rwh															

Field	Bits	Type	Description
RXTS	15:0	rwh	**Rx Timestamp** Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC.TCP.

Field	Bits	Type	Description
DLC	19:16	rwh	**Data Length Code** 0_H CAN + CAN FD: received frame has 0 data bytes … 8_H CAN + CAN FD: received frame has 8 data bytes 9_H CAN FD: received frame has 12 (9*4-24) data bytes CAN: received frame has 8 data bytes … C_H CAN FD: received frame has 24 (12*4-24) data bytes CAN: received frame has 8 data bytes D_H CAN FD: received frame has 32 (13*16-176) data bytes CAN: received frame has 8 data bytes … F_H CAN FD: received frame has 64 (15*16-176) data bytes CAN: received frame has 8 data bytes
BRS	20	rwh	**Bit Rate Switch** 0_B Frame received without bit rate switching 1_B Frame received with bit rate switching
FDF	21	rwh	**Frame Data Format** 0_B Standard frame format 1_B CAN FD frame format (new DLC-coding and CRC)
FIDX	30:24	rwh	**Filter Index** 00_H Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC.LSS - 1 resp. XIDFC.LSE - 1. … $7F_H$ Index of matching Rx acceptance filter element (invalid if ANMF = '1'). Range is 0 to SIDFC.LSS - 1 resp. XIDFC.LSE - 1.
ANMF	31	rwh	**Accepted Non-matching Frame** Acceptance of non-matching frames may be enabled via GFC.ANFS and GFC.ANFE. 0_B Received frame matching filter index FIDX 1_B Received frame did not match any Rx filter element
0	23:22	rwh	**Reserved** Shall read 0, shall be written with 0.

Data Byte m

RxMsgk_DBm (k=0-63;m=0-63)

Data Byte m **(000008_H+k*48_H+m)** **Application Reset Value: XX_H**

7	6	5	4	3	2	1	0
DB							
rwh							

Field	Bits	Type	Description
DB	7:0	rwh	**Data Byte m**

Note: Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words (Rn = 3…17) are used for storage of a CAN message's data field.

40.4.6.3 Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TXBCi.TFQS and TXBCi.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register TXESCi.

Table 383 Message Layout - Tx Buffer Element

	31…24	23…16	15…8	7…0
T0	0 | XTD | RTR | ID[28:0]			
T1	MM[7:0]	EFC | 0 | FDF | BRS | DLC[3:0]	0	
T2	DB3[7:0]	DB2[7:0]	DB1[7:0]	DB0[7:0]
T3	DB7[7:0]	DB6[7:0]	DB5[7:0]	DB4[7:0]
…	…	…	…	…
Tn	DBm[7:0]	DBm-1[7:0]	DBm-2[7:0]	DBm-3[7:0]

Transmit Buffer 0

See Message layout.

TxMsgk_T0 (k=0-31)
Transmit Buffer 0 **($000000_H + k*48_H$)** **Application Reset Value: XXXX XXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESI	**XTD**	**RTR**	**ID**												
rw	rw	rw	rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID															
rw															

Field	Bits	Type	Description
ID	28:0	rw	**Identifier** Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Field	Bits	Type	Description
RTR	29	rw	**Remote Transmission Request** *Note: When RTR = 1, the M_CAN transmits a remote frame according to ISO11898-1, even if CCCR.FDOE enables the transmission in CAN FD format.* 0_B Transmit data frame 1_B Transmit remote frame
XTD	30	rw	**Extended Identifier** 0_B 11-bit standard identifier 1_B 29-bit extended identifier
ESI	31	rw	**Error State Indicator** *Note: The ESI bit of the transmit buffer is OR'ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive.* 0_B ESI bit in CAN FD format depends only on error passive flag 1_B ESI bit in CAN FD format transmitted recessive

Transmit Buffer 1

See Message layout.

TxMsgk_T1 (k=0-31)

Transmit Buffer 1 **(000004_H+k*48_H)** **Application Reset Value: XXXX $XXXX_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MM								EFC	0	FDF	BRS	DLC			
rw								rw	rw	rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
rw															

Field	Bits	Type	Description
DLC	19:16	rw	**Data Length Code** 0_H CAN + CAN FD: received frame has 0 data bytes … 8_H CAN + CAN FD: received frame has 8 data bytes 9_H CAN FD: received frame has 12 (9*4-24) data bytes CAN: received frame has 8 data bytes … C_H CAN FD: received frame has 24 (12*4-24) data bytes CAN: received frame has 8 data bytes D_H CAN FD: received frame has 32 (13*16-176) data bytes CAN: received frame has 8 data bytes … F_H CAN FD: received frame has 64 (15*16-176) data bytes CAN: received frame has 8 data bytes
BRS	20	rw	**Bit Rate Switching** *Note: Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled CCCR.FDOE = 1'. Bit BRS is only evaluated when in addition CCCR.BRSE = '1'.* 0_B CAN FD frames transmitted without bit rate switching 1_B CAN FD frames transmitted with bit rate switching
FDF	21	rw	**FD Format** 0_B Frame transmitted in Classical CAN format 1_B Frame transmitted in CAN FD format
EFC	23	rw	**Event FIFO Control** 0_B Don't store Tx events 1_B Store Tx events
MM	31:24	rw	**Message Marker** Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
0	15:0, 22	rw	**Reserved** Shall read 0, shall be written with 0.

Data Byte m

TxMsgk_DBm (k=0-31;m=0-63)

Data Byte m **(000008_H+k*48_H+m)** **Application Reset Value: XX_H**

7	6	5	4	3	2	1	0
DB							
rw							

Field	Bits	Type	Description
DB	7:0	rw	**Data Byte m**

Note: Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 3…17) are used for storage of a CAN message's data field.

40.4.6.4 Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the Host CPU gets this information in the order the messages were transmitted. Status information about the Tx Event FIFO can be obtained from register TXEFS.

Table 384 Message Layout - Tx Event FIFO Element

	31…24	23…16	15…8	7…0
E0	ESI, XTD, RTR, ID[28:0]			
E1	MM[7:0]	ET[1:0], FDF, BRS, DLC[3:0]	TXTS[15:0]	

Event 0

See Message layout.

TxEventk_E0 (k=0-31)

Event 0 **(000000_H + k*8)** **Application Reset Value: XXXX XXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ESI	XTD	RTR	ID												
rwh	rwh	rwh	rwh												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID															
rwh															

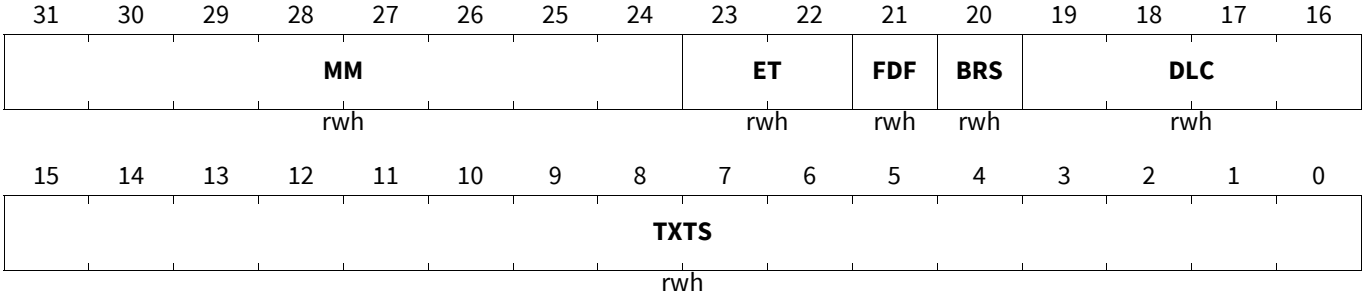
Field	Bits	Type	Description
ID	28:0	rwh	**Identifier** Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
RTR	29	rwh	**Remote Transmission Request** 0_B Data frame transmitted 1_B Remote frame transmitted
XTD	30	rwh	**Extended Identifier** 0_B 11-bit standard identifier 1_B 29-bit extended identifier
ESI	31	rwh	**Error State Indicator** 0_B Transmitting node is error active 1_B Transmitting node is error passive

Event 1

See Message layout.

TxEventk_E1 (k=0-31)

Event 1 **(000004_H+k*8)** **Application Reset Value: XXXX $XXXX_H$**



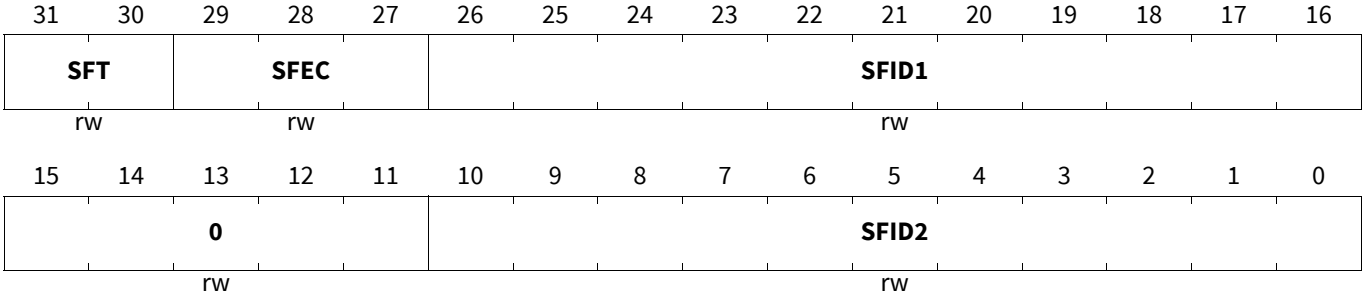
Field	Bits	Type	Description
TXTS	15:0	rwh	**Tx Timestamp** Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TSCC.TCP.
DLC	19:16	rwh	**Data Length Code** 0_H CAN + CAN FD: received frame has 0 data bytes … 8_H CAN + CAN FD: received frame has 8 data bytes 9_H CAN FD: received frame has 12 (9*4-24) data bytes CAN: received frame has 8 data bytes … C_H CAN FD: received frame has 24 (12*4-24) data bytes CAN: received frame has 8 data bytes D_H CAN FD: received frame has 32 (13*16-176) data bytes CAN: received frame has 8 data bytes … F_H CAN FD: received frame has 64 (15*16-176) data bytes CAN: received frame has 8 data bytes
BRS	20	rwh	**Bit Rate Switch** 0_B Frame transmitted without bit rate switching 1_B Frame transmitted with bit rate switching
FDF	21	rwh	**FD Format** 0_B Standard frame format 1_B CAN FD frame format (new DLC-coding and CRC)
ET	23:22	rwh	**Event Type** 00_B Reserved 01_B Tx event 10_B Transmission in spite of cancellation (always set for transmissions in DAR mode) 11_B Reserved
MM	31:24	rwh	**Message Marker** Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.

40.4.6.5 Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address SIDFCi.FLSSA plus the index of the filter element (0…127).

Standard Message 0

StdMsgk_S0 (k=0-127)
Standard Message 0 **(000000_H + k*4)** **Application Reset Value: XXXX $XXXX_H$**



Field	Bits	Type	Description
SFID2	10:0	rw	**Standard Filter ID 2** This bit field has a different meaning depending on the configuration of SFEC: 1) SFEC = “001”…”110” Second ID of standard ID filter element 2) SFEC = “111” Filter for Rx Buffers or for debug messages **SFID2[5:0]** defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message. **SFID2[8:6]** is used to control the filter event pins. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one host clock period in case the filter matches. **SFID2[10:9]** decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 000_H Store message into an Rx Buffer … $1FF_H$ Store message into an Rx Buffer 200_H Debug Message A … $3FF_H$ Debug Message A 400_H Debug Message B … $5FF_H$ Debug Message B 600_H Debug Message C … $7FF_H$ Debug Message C

Field	Bits	Type	Description
SFID1	26:16	rw	**Standard Filter ID 1** First ID of standard ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
SFEC	29:27	rw	**Standard Filter Element Configuration** All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101” or “110”, a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000_B Disable filter element 001_B Store in Rx FIFO 0 if filter matches 010_B Store in Rx FIFO 1 if filter matches 011_B Reject ID if filter matches 100_B Set priority if filter matches 101_B Set priority and store in FIFO 0 if filter matches 110_B Set priority and store in FIFO 1 if filter matches 111_B Store into Rx Buffer or as debug message, configuration of SFT[1:0] ignored
SFT	31:30	rw	**Standard Filter Type** *Note: With SFT = “11” the filter element is disabled and the acceptance filtering continues (same behaviour as with SFEC = “000”)* 00_B Range filter from SF1ID to SF2ID (SF2ID ≥ SF1ID) 01_B Dual ID filter for SF1ID or SF2ID 10_B Classic filter: SF1ID = filter, SF2ID = mask 11_B Filter element disabled
0	15:11	rw	**Reserved** Shall read 0, shall be written with 0.

40.4.6.6 Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address XIDFCi.FLESA plus two times the index of the filter element (0…63).

Table 385 Message Layout - Extended Message ID Filter Element

3 1	2 4	2 3	1 6	1 5	8	7	0

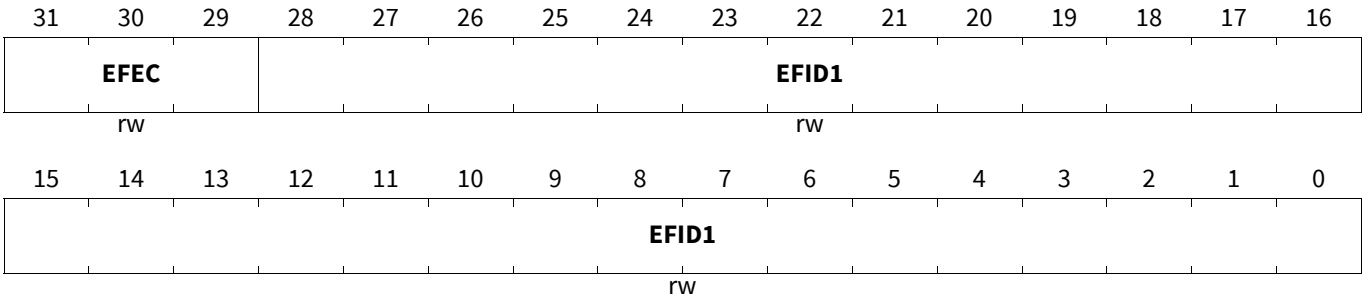
Table 385 Message Layout - Extended Message ID Filter Element (cont'd)

F0	EFEC[2:0]		EFID1[28:0]
F1	EFT[1:0]	0	EFID2[28:0]

Filter Element 0

See Message layout.

ExtMsgk_F0 (k=0-63)
Filter Element 0 **(000000_H + k*8)** **Application Reset Value: XXXX $XXXX_H$**

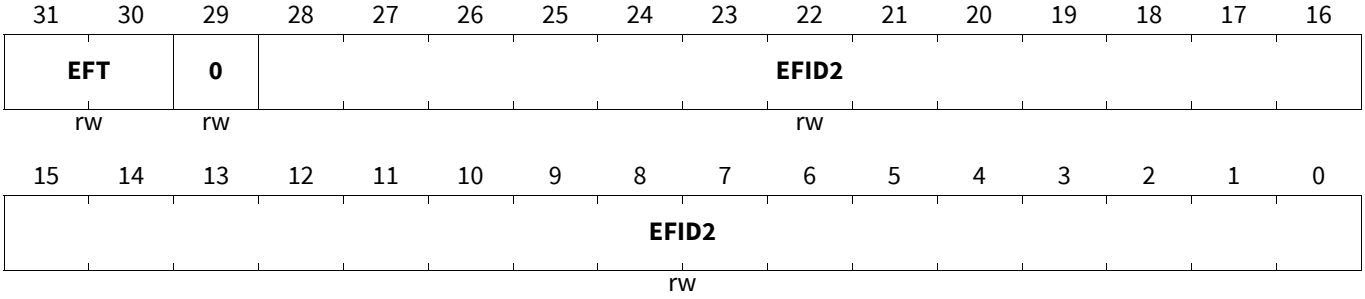


Field	Bits	Type	Description
EFID1	28:0	rw	**Extended Filter ID 1** First ID of extended ID filter element. When filtering for Rx Buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAMi masking mechanism is used.
EFEC	31:29	rw	**Extended Filter Element Configuration** All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = "100", "101" or "110", a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000_B Disable filter element 001_B Store in Rx FIFO 0 if filter matches 010_B Store in Rx FIFO 1 if filter matches 011_B Reject ID if filter matches 100_B Set priority if filter matches 101_B Set priority and store in FIFO 0 if filter matches 110_B Set priority and store in FIFO 1 if filter matches 111_B Store into Rx Buffer or as debug message, configuration of EFT[1:0] ignored

Filter Element 1

See Message layout.

ExtMsgk_F1 (k=0-63)
Filter Element 1 **(000004_H+k*8)** **Application Reset Value: XXXX $XXXX_H$**



Field	Bits	Type	Description
EFID2	28:0	rw	**Extended Filter ID 2** This bit field has a different meaning depending on the configuration of EFEC: 1) EFEC = “001”…”110” Second ID of extended ID filter element 2) EFEC = “111” Filter for Rx Buffers or for debug messages **EFID2[5:0]** defines the offset to the Rx Buffer Start Address RXBC.RBSA for storage of a matching message. **EFID2[8:6]** is used to control the filter event. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one host clock period in case the filter matches. **EFID2[10:9]** decides whether the received message is stored into an Rx Buffer or treated as message A, B, or C of the debug message sequence. 00000000_HStore message into an Rx Buffer … $000001FF_H$Store message into an Rx Buffer 00000200_HDebug Message A … $000003FF_H$Debug Message A 00000400_HDebug Message B … $000005FF_H$Debug Message B 00000600_HDebug Message C … $000007FF_H$Debug Message C
EFT	31:30	rw	**Extended Filter Type** 00_B Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) 01_B Dual ID filter for EF1ID or EF2ID 10_B Classic filter: EF1ID = filter, EF2ID = mask 11_B Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
0	29	rw	**Reserved** Shall read 0, shall be written with 0

40.4.6.7 Trigger Memory Element

Up to 64 trigger memory elements can be configured. When accessing a Trigger Memory element, its address is the Trigger Memory Start Address TTTMC.TMSA plus the index of the trigger memory element (0…63)

Table 386 Message Layout - Trigger Memory Element

	31	16	15	8	7	0
TM0	TM[15:0]		0	CC[6:0]	0	TMIN, TMEX, TYPE[3:0]
TM1	0	FTYPE, MNR[6:0]	0			MSC[2:0]

Trigger Memory Element 0

See Message layout, see Trigger Memory Element Overview.

TrigMsgk_TM0 (k=0-63)
Trigger Memory Element 0 **(000000_H + k*8)** **Application Reset Value: XXXX $XXXX_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TM															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CC							0		TMIN	TMEX	TYPE			
rw	rw							rw		rw	rw	rw			

Field	Bits	Type	Description
TYPE	3:0	rw	**Trigger Type** *Note: No ASC implemented. If and only if implemented: For ASC operation (ASC = "10", "11") only trigger types Rx_Trigger and Time_Base_Trigger should be used.* 0_H Tx_Ref_Trigger - valid when not in Gap 1_H Tx_Ref_Trigger_Gap - valid when in Gap 2_H Tx_Trigger_Single - starts a single transmission in an exclusive time window 3_H Tx_Trigger_Continous - starts continuous transmission in an exclusive time window 4_H Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window 5_H Tx_Trigger_Merged - starts a merged arbitration window 6_H Watch_Trigger - valid when not in Gap 7_H Watch_Trigger_Gap - valid when in Gap 8_H Rx_Trigger - check for reception 9_H Time_Base_Trigger - only control TMIN, TMEX, and ASC **Others**, End_of_List - illegal type, causes config error
TMEX	4	rw	**Time Mark Event External** 0_B No action 1_B When the time mark of the trigger memory element becomes active and TTOCN.TTMIE = '1'
TMIN	5	rw	**Time Mark Event Internal** 0_B No action 1_B TTIR.TTMI is set when trigger memory element becomes active

Field	Bits	Type	Description
CC	14:8	rw	**Cycle Code** Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List. 00_H valid for all cycles 01_H valid for all cycles 02_H valid every second cycle at cycle count mod2 = c 03_H valid every second cycle at cycle count mod2 = c 04_H valid every fourth cycle at cycle count mod4 = cc … 07_H valid every fourth cycle at cycle count mod4 = cc 08_H valid every eighth cycle at cycle count mod8 = ccc … $0F_H$ valid every eighth cycle at cycle count mod8 = ccc 10_H valid every sixteenth cycle at cycle count mod16 = cccc … $1F_H$ valid every sixteenth cycle at cycle count mod16 = cccc 20_H valid every thirty-second cycle at cycle count mod32 = ccccc … $3F_H$ valid every thirty-second cycle at cycle count mod32 = ccccc 40_H valid every sixty-fourth cycle at cycle count mod64 = cccccc … $7F_H$ valid every sixty-fourth cycle at cycle count mod64 = cccccc
TM	31:16	rw	**Time Mark** Cycle time for which the trigger becomes active.
0	7:6, 15	rw	**Reserved** Shall read 0, shall be written with 0.

See Message layout, see Trigger Memory Element Overview.

TrigMsgk_TM1 (k=0-63)

(000004_H+k*8) **PowerOn Reset Value: XXXX XXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								FTYPE	MMR						
rw								rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														MSC	
rw														rwh	

Field	Bits	Type	Description
MSC	2:0	rwh	**Message Status Count** Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1). *Note: The trigger memory elements have to be written when the M_CAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed. There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual Reference Trigger Offset (TTOST.RTO).* 000_B Actual status … 111_B Actual status
MMR	22:16	rw	**Message Number** Transmission: Trigger is valid for configured Tx Buffer number. Valid values are 0 to 31. Reception: Trigger is valid for standard / extended message ID filter element number. Valid values are 0 to 63 resp. 0 to 127.
FTYPE	23	rw	**Filter Type** 0_B 11-bit standard message ID 1_B 29-bit extended message ID
0	15:3, 31:24	rw	**Reserved** Shall read 0, shall be written with 0.

40.5 IO Interfaces

Table 387 List of CAN Interface Signals

Interface Signals	I/O	Description
INT(15:0)	out	**CAN Service Request**
TRIG(3:0)	in	**GTM timer output vector**
STM0_SR0_INT	in	**System Timer Service Request 0**
STM0_SR1_INT	in	**System Timer Service Request 1**
STM1_SR0_INT	in	**System Timer Service Request 0**
STM1_SR1_INT	in	**System Timer Service Request 1**
STM2_SR0_INT	in	**System Timer Service Request 0**
STM2_SR1_INT	in	**System Timer Service Request 1**
CANX0:RXDA	in	**CAN receive input node 0**
CANX0:RXDB		
CANX0:RXDC		
CANX0:RXDD		
CANX0:RXDE		
CANX0:RXDF		
CANX0:RXDG		
CANX0:RXDH		
CANX1:RXDA	in	**CAN receive input node 1**
CANX1:RXDB		
CANX1:RXDC		
CANX1:RXDD		
CANX1:RXDE		
CANX1:RXDF		
CANX1:RXDG		
CANX1:RXDH		
CANX2:RXDA	in	**CAN receive input node 2**
CANX2:RXDB		
CANX2:RXDC		
CANX2:RXDD		
CANX2:RXDE		
CANX2:RXDF		
CANX2:RXDG		
CANX2:RXDH		

Table 387 List of CAN Interface Signals (cont'd)

Interface Signals	I/O	Description
CANX3:RXDA	in	**CAN receive input node 3**
CANX3:RXDB		
CANX3:RXDC		
CANX3:RXDD		
CANX3:RXDE		
CANX3:RXDF		
CANX3:RXDG		
CANX3:RXDH		
CANX0:TXD	out	**CAN transmit output node 0**
CANX1:TXD	out	**CAN transmit output node 1**
CANX2:TXD	out	**CAN transmit output node 2**
CANX3:TXD	out	**CAN transmit output node 3**
ECTT1	in	**External CAN time trigger input**
ECTT2		
ECTT3		
ECTT4		
ECTT5		
ECTT6		
ECTT7		
ECTT8		
ECTT(15:9)		
TTCPT_TRIG(4:1)	in	**Capture time trigger input**
INT(15:0)	out	**CAN interrupt request**
DXSCLK	out	**DXS Clock, DAP module clock**
DSTDBG	in	**Destructive Debug entered**

Note: For the connectivity of the MCMCAN module to the STM module, please refer to ***"CAN Transmit Trigger Inputs" on Page 14****.*

40.6 Glossary - Terms and Abbreviations

This document uses the following terms and abbreviations.

- BRP Bit Rate Prescaler
- BSP Bit Stream Processor
- BTL Bit Timing Logic
- CAN Controller Area Network
- CAN FD Controller Area Network Flexible Data-rate
- CRC Cyclic Redundancy Check
- DLC Data Length Code
- ECC Error Correction Code
- ECU Electronic Control Unit
- EML Error Management Logic
- FSE Frame Synchronization Entity
- FSM Finite State Machine
- MRM Master Reference Mark
- MSC Message Status Count
- mtq minimum time quanta = CAN clock period (m_ttcan_clk)
- NTU Network Time Unit
- SSP Secondary Sample Point
- TDC Transmitter Delay Compensation
- tq time quantum
- TSEG1 Time Segment before Sample Point
- TSEG2 Time Segment after Sample Point
- TTCAN Time - Triggered CAN
- TUR Time Unit Ratio

40.7 Revision History

Table 388 Revision History

Reference	Change to Previous Version	Comment
V1.19.8		
Page 10, **Page 24**, **Page 172**	Note added for recommendation on CAN FD maximum bit rate, port configuration for CAN FD communication and Trigger Memory Element 0.	
V1.19.9		
-	No change in family specification.	
V1.19.10		
Page 25	Typo “layerm” corrected to “layer” in the note paragraph.	
Page 65	Access term “OEN” added to OCS register.	
Page 14, **Page 15**	STM Timer Trigger signals naming aligned as given in the Connectivity table in Appendix.	
Page 64	Updated figure (removed wrong ACCEN1).	
Page 177	Added note at the end of IO Interfaces table.	
V1.19.11		
Page 5	Wrong f_{MCANH} changed to f_{MCAN} at ‘f_{ASYN} is supplied from f_{MCAN}’. Additionally some typos fixed in this paragraph.	
Page 27	Wrong TX changed to RX at ‘TDCRi (i=0-3).TDCF AND RX is low’.	
Page 112	Corrected register IRi bit field 29 to ‘0’/’r’ (external/UM only).	
V1.19.12		
Page 2	Updated **Section 40.1**.	
V1.19.13		
Page 64	Updated information on bit implementation in A-step.	

41 FlexRay™ Protocol Controller (E-Ray)

The E-Ray IP-module performs communication according to the FlexRay™ [1)] protocol specification v2.1, developed for automotive applications. With maximum specified clock the bitrate can be programmed to values up to 10 Mbit/s. Additional bus driver (BD) hardware is required for connection to the physical layer.

41.1 Feature List

The E-Ray IP-module supports the following features:

- Conformance with FlexRay™ protocol specification v2.1
- Data rates of up to 10 Mbit/s on each channel
- Up to 128 Message Buffers configurable
- 8 Kbyte of Message RAM for storage of e.g. 128 Message Buffers with max. 48 byte data field or up to 30 Message Buffers with 254 byte Data Sections
- Configuration of Message Buffers with different payload lengths possible
- One configurable receive FIFO
- Each Message Buffer can be configured as receive buffer, as transmit buffer or as part of the receive FIFO
- Host access to Message Buffers via Input and Output Buffer.
  Input Buffer: Holds message to be transferred to the Message RAM
  Output Buffer: Holds message read from the Message RAM
- Filtering for slot counter, cycle counter, and channel
- Maskable module service requests
- Network Management supported
- Four service request lines
- Automatic delayed read access to Output Command Request Register (OBCR) if a data transfer from Message RAM to Output Shadow Buffer (initiated by a previous write access to the OBCR) is ongoing.
- Automatic delayed read access to Input Command Request Register (IBCR) if a data transfer from Input Shadow Buffer to Message RAM to (initiated by a previous write access to the IBCR) is ongoing.
- Four Input Buffer for building up transmission Frames in parallel.
- Flag indicating which Input Buffer is currently accessible by the host.

41.2 Overview

For communication on a FlexRay™ network, individual Message Buffers with up to 254 data byte are configurable. The message storage consists of a single-ported Message RAM that holds up to 128 Message Buffers. All functions concerning the handling of messages are implemented in the Message Handler. Those functions are the acceptance filtering, the transfer of messages between the two FlexRay™ Channel Protocol Controllers and the Message RAM, maintaining the transmission schedule as well as providing message status information.

The register set of the E-Ray IP-module can be accessed directly by an external Host via the module's Host interface. These registers are used to control/configure/monitor the FlexRay™ Channel Protocol Controllers, Message Handler, Global Time Unit, System Universal Control, Frame and Symbol Processing, Network Management, Service Request Control, and to access the Message RAM via Input / Output Buffer.

41.2.1 E-Ray Kernel Description

Figure below shows a global view of the E-Ray interface.

1) Infineon®, Infineon Technologies®, are trademarks of Infineon Technologies AG. FlexRay™ is a trademark of FlexRay Consortium.

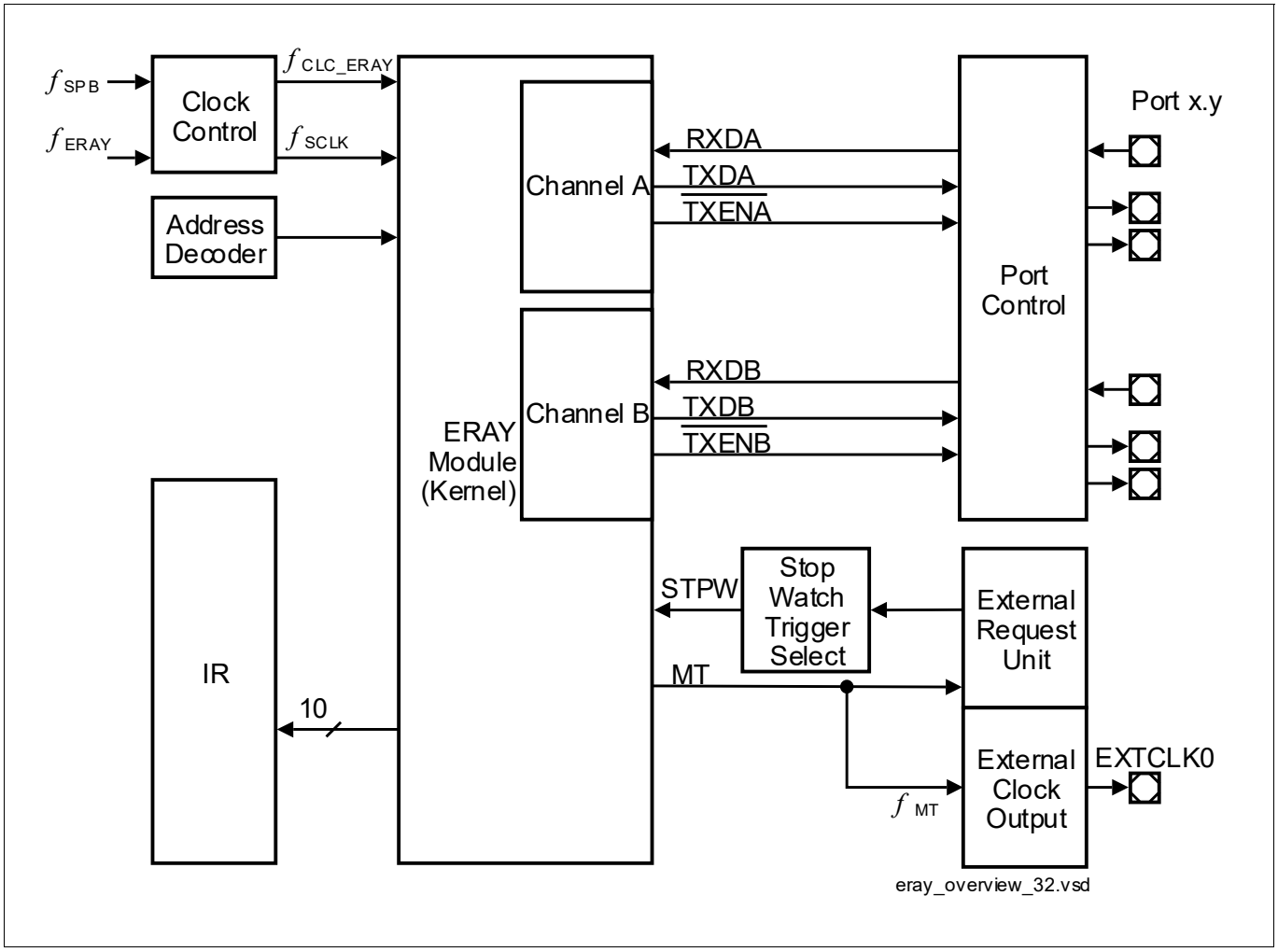


Figure 606 General Block Diagram of the E-Ray Interface

The E-Ray module communicates with the external world via three I/O lines each channel. The RXDAx and RXDBx lines are the receive data input signals, TXDA and TXDB lines are the transmit output signals, $\overline{\text{TXENA}}$ and $\overline{\text{TXENB}}$ the transmit enable signals.

Clock control, address decoding, and service request control are managed outside the E-Ray module kernel.

41.2.2 Block Diagram

The E-Ray is built up by the following main submodules:

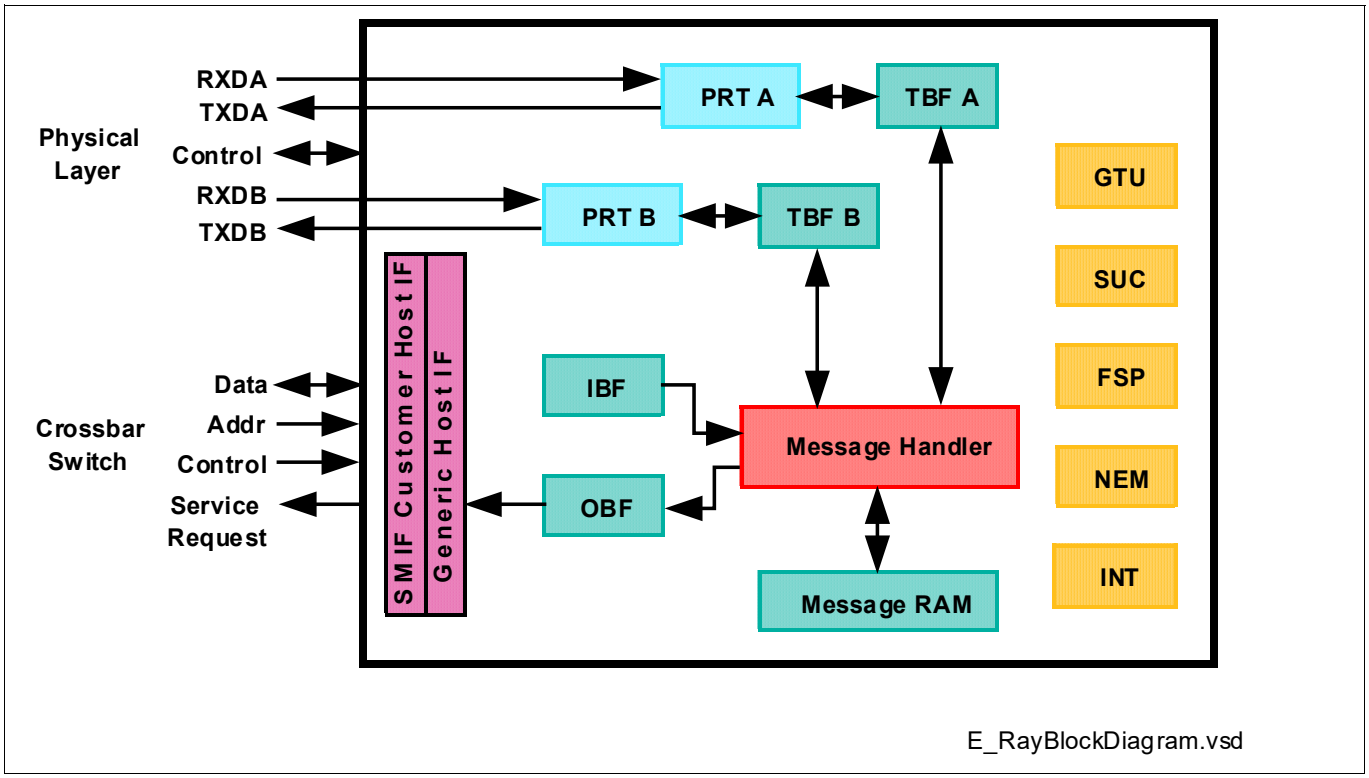


Figure 607 E-Ray Block Diagram

Customer Host Interface (CIF)

Connects the FPI Bus to the E-Ray IP-module via the Generic Host Interface.

Generic Host Interface (GIF)

The E-Ray IP-module is provided with an 8/16/32-bit Generic Host Interface prepared for the connection to a wide range of customer-specific Hosts. Configuration registers, status registers, and service request registers are attached to the respective blocks and can be accessed via the Generic Host Interface.

Input Buffer (IBF)

For write access to the Message Buffers configured in the Message RAM, the Host can write the Header and Data Section for a specific Message Buffer to the Input Buffer. The Message Handler then transfers the data from the Input Buffer to the selected Message Buffer in the Message RAM.

Because the Input Buffer (IBF) Scheme does only allow to write the entire Message Frame, not only parts of it, the number of IBF has been increased from originally 2 to 4. This enables to fill the buffer partly and at the end request transfer into Message RAM. Therefore 2 extra bits allow to switch between the two banks of IBF and one status bit signals the IBF currently active for Host writes.

Output Buffer (OBF)

For read access to a Message Buffer configured in the Message RAM the Message Handler transfers the selected Message Buffer to the Output Buffer. After the transfer has completed, the Host can read the Header and Data Section of the transferred Message Buffer from the Output Buffer.

Message Handler (MHD)

The E-Ray Message Handler controls data transfers between the following components:

- Input / Output Buffer and Message RAM
- Transient Buffer RAMs of the two FlexRay™ Protocol Controllers and Message RAM

Message RAM (MRAM)

The Message RAM consists of a single-ported RAM that stores up to 128 FlexRay™ Message Buffers together with the related configuration data (Header and Data Partition).

Transient Buffer RAM (TBF 1/2)

Stores the Data Section of two complete messages.

FlexRay™ Channel Protocol Controller (PRT A/B)

The FlexRay™ Channel Protocol Controllers consist of shift register and FlexRay™ protocol FSM. They are connected to the Transient Buffer RAMs for intermediate message storage and to the physical layer via bus driver BD.

They perform the following functionality:

- Control and check of bit timing
- Reception and transmission of FlexRay™ Frames and symbols
- Check of Header CRC
- Generation / check of Frame CRC
- Interfacing to bus driver

The FlexRay™ Channel Protocol Controllers have interfaces to:

- Physical Layer (bus driver)
- Transient Buffer RAM
- Message Handler
- Global Time Unit
- System Universal Control
- Frame and Symbol Processing
- Network Management
- Service Request Control

Global Time Unit (GTU)

The Global Time Unit performs the following functions:

- Generation of Microtick
- Generation of Macrotick
- Fault tolerant clock synchronization by FTM algorithm
  - Rate correction
  - Offset correction
- Cycle counter
- Timing control of static segment
- Timing control of dynamic segment (minislotting)
- Support of external clock correction

System Universal Control (SUC)

The System Universal Control controls the following functions:

- Configuration
- Wakeup
- Startup
- Normal Operation
- Passive Operation
- Monitor Mode

Frame and Symbol Processing (FSP)

The Frame and Symbol Processing controls the following functions:

- Checks the correct timing of Frames and symbols
- Tests the syntactical and semantical correctness of received Frames
- Sets the slot status flags

Network Management (NEM)

Handles of the Network Management vector

Service Request Control (INT)

The Service Request Controller performs the following functions:

- Provides error and status service request flags
- Enables and disables service request sources
- Assignment of service request sources to one of the two module service request lines
- Enables and disables module service request lines
- Manages the two service request timers
- Stop watch time capturing

41.3 Functional Description

This chapter describes the E-Ray implementation together with the related FlexRay™ protocol features. More information about the FlexRay™ protocol itself can be found in the FlexRay™ protocol specification v2.1.

Communication on FlexRay™ networks is based on Frames and symbols. The wakeup symbol (WUS) and the collision avoidance symbol (CAS) are transmitted outside the communication cycle to setup the time schedule. Frames and media access test symbols (MTS) are transmitted inside the communication cycle.

41.3.1 Definitions

FlexRay™ Frame: Header Segment + Payload Segment

Message Buffer: Header Section + Data Section

Message RAM: Header Partition + Data Partition

Data Frame: FlexRay™ Frame that is not a NULL Frame

41.3.2 Communication Cycle

A communication cycle in FlexRay™ consists of the following elements:

- Static Segment
- Dynamic Segment
- Symbol Window
- Network Idle Time (NIT)

Static segment, dynamic segment, and symbol window form the Network Communication Time (NCT). For each communication channel the slot counter starts at 1 and counts up until the end of the dynamic segment is reached. Both channels share the same arbitration grid which means that they use the same synchronized Macrotick.

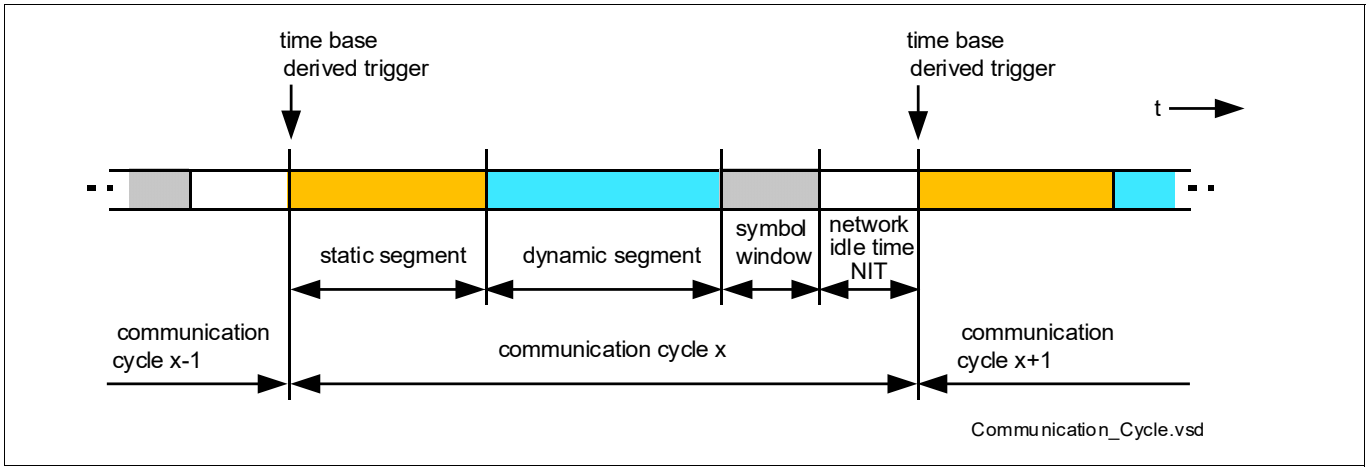


Figure 608 Structure of Communication Cycle

41.3.2.1 Static Segment

The Static Segment is characterized by the following features:

- Time slots of fixed length (optionally protected by bus guardian)
- Start of Frame transmission at action point of the respective static slot
- Payload length same for all Frames on both channel

Parameters: Number of Static Slots GTUC07.NSS, Static Slot Length GTUC07.SSL, Payload Length Static MHDC.SFDL, Action Point Offset GTUC09.APO.

41.3.2.2 Dynamic Segment

The Dynamic Segment is characterized by the following features:

- All controllers have bus access (no bus guardian protection possible)
- Variable payload length and duration of slots, different for both channels
- Start of transmission at minislot action point

Parameters: Number of Minislots GTUC08.NMS, Minislot Length GTUC08.MSL Minislot Action Point Offset GTUC09.MAPO, Start of Latest Transmit (last minislot) MHDC.SLT.

41.3.2.3 Symbol Window

During the symbol window only one media access test symbol (MTS) may be transmitted per channel. MTS symbols are send in “NORMAL_ACTIVE” state to test the bus guardian.

The symbol window is characterized by the following features:

- Send single symbol
- Transmission of the MTS symbol starts at the symbol windows action point

Parameters: Symbol Window Action Point Offset GTUC09.APO (same as for static slots), Network Idle Time Start GTUC04.NIT.

41.3.2.4 Network Idle Time (NIT)

During network idle time the Communication Controller has to perform the following tasks:

- Calculate clock correction terms (offset and rate)
- Distribute offset correction over multiple Macroticks
- Perform cluster cycle related tasks

Parameters: Network Idle Time Start GTUC04.NIT, Offset Correction Start GTUC04.OCS.

41.3.2.5 Configuration of Network Idle Time (NIT) Start and Offset Correction Start

The number of Macroticks per cycle (gMacroPerCycle) is assumed to be m. It is configured by programming GTUC02.MPC = m.

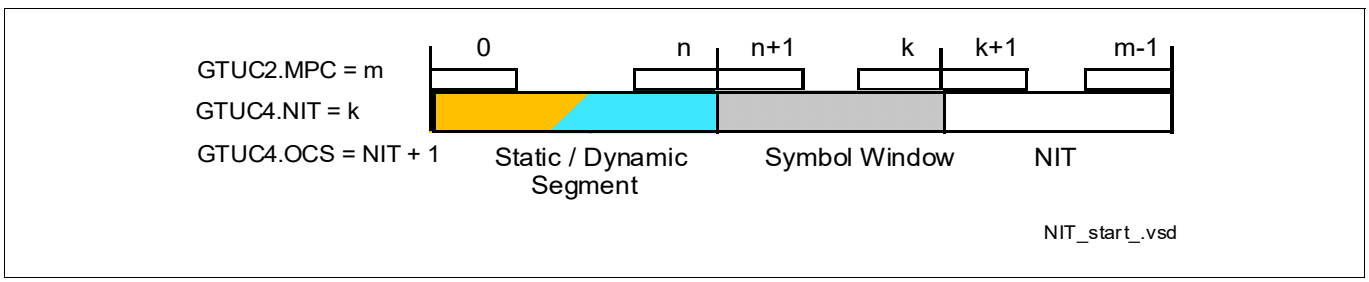


Figure 609 Configuration of network idle time (NIT) start and offset correction start

The static / dynamic segment starts with Macrotick 0 and ends with Macrotick n:

n = static segment length + dynamic segment offset + dynamic segment length - 1 Macrotick

n = gNumberOfStaticSlots • gdStaticSlot + dynamic segment offset
+ gNumberOfMinislots • gdMinislot - 1 Macroticks

The static segment length is configured by GTUC07.SSL and GTUC07.NSS.
The dynamic segment length is configured by GTUC08.MSL and GTUC08.NMS.

The dynamic segment offset is:

If gdActionPointOffset ≤ gdMinislotActionPointOffset:

dynamic segment offset = 0 MT

Else if gdActionPointOffset > gdMinislotActionPointOffset:

dynamic segment offset = gdActionPointOffset - gdMinislotActionPointOffset

The network idle time (NIT) starts with Macrotick k+1 and ends with the last Macrotick of cycle m-1. It has to be configured by setting GTUC04.NIT = k.

For the E-Ray the offset correction start is required to be

GTUC04.OCS ≥ GTUC04.NIT + 1 = k+1.

The length of symbol window results from the number of Macroticks between the end of the static / dynamic segment and the beginning of the NIT. It can be calculated by k - n.

41.3.3 Communication Modes

The FlexRay™ Protocol Specification v2.1 defines the Time-Triggered Distributed (TT-D) mode.

Time-triggered Distributed (TT-D)

In TT-D mode the following configurations are possible:

- **Pure static**: minimum 2 static slots + symbol window (optional)
- **Mixed static/dynamic**: minimum 2 static slots + dynamic segment + symbol window (optional)

A minimum of two coldstart nodes need to be configured for distributed time-triggered operation. Two fault-free coldstart nodes are necessary for the cluster startup. Each Startup Frame must be a SYNC Frame, therefore all coldstart nodes are sync nodes.

41.3.4 Clock Synchronization

In TT-D mode a distributed clock synchronization is used. Each node individually synchronizes itself to the cluster by observing the timing of received SYNC Frames from other nodes.

41.3.4.1 Global Time

Activities in a FlexRay™ node, including communication, are based on the concept of a global time, even though each individual node maintains its own view of it. It is the clock synchronization mechanism that differentiates the FlexRay™ cluster from other node collections with independent clock mechanisms. The global time is a vector of two values; the cycle (cycle counter) and the cycle time (Macrotick counter).

Cluster specific:

- Macrotick = basic unit of time measurement in a FlexRay™ network, a Macrotick consists of an integer number of Microticks
- Cycle length = duration of a communication cycle in units of Macroticks

41.3.4.2 Local Time

Internally, nodes time their behavior with Microtick resolution. Microticks are time units derived from the oscillator clock tick of the specific node. Therefore Microticks are controller-specific units. They may have different duration in different controllers. The precision of a node's local time difference measurements is a Microtick.

Node specific:

- Oscillator clock → prescaler → Microtick
- Microtick = basic unit of time measurement in a Communication Controller, clock correction is done in units of Microticks
- Cycle counter + Macrotick counter = nodes local view of the global time

41.3.4.3 Synchronization Process

Clock synchronization is performed by means of SYNC Frames. Only preconfigured nodes (sync nodes) are allowed to send SYNC Frames. In a two-channel cluster a sync node has to send its SYNC Frame on both channels.

For synchronization in FlexRay™ the following constraints have to be considered:

- Max. one SYNC Frame per node in one communication cycle
- Max. 15 SYNC Frames per cluster in one communication cycle
- Every node has to use all available SYNC Frames for clock synchronization
- Minimum of two sync nodes required for clock synchronization and startup

For clock synchronization the time difference between expected and observed arrival time of SYNC Frames received during the static segment, valid on both channels (two-channel cluster), is measured. The calculation of correction terms is done during network idle time (NIT) (offset: every cycle, rate: odd cycle) by using a FTA / FTM algorithm. For details see FlexRay™ protocol specification v2.1, chapter 8.

Offset (phase) Correction

- Only deviation values measured and stored in the current cycle used
- For a two channel node the smaller value will be taken
- Calculation during network idle time (NIT) of **every** communication cycle, value may be negative
- Offset correction value calculated in even cycles used for error checking only
- Checked against limit values (violation: “NORMAL_ACTIVE” → “NORMAL_PASSIVE” → “HALT”)
- Correction value is an integer number of Microticks
- Correction done in **odd** numbered cycles, distributed over the Macroticks beginning at offset correction start up to cycle end (end of network idle time (NIT)) to shift nodes next start of cycle (Macroticks lengthened / shortened)

Rate (frequency) Correction

- Pairs of deviation values measured and stored in even / odd cycle pair used
- For a two channel node the average of the differences from the two channels is used
- Calculated during network idle time (NIT) of **odd** numbered cycles, value may be negative
- Cluster drift damping is performed using global damping value
- Checked against limit values
- Correction value is a signed integer number of Microticks
- Distributed over Macroticks comprising the next **even / odd** cycle pair (Macroticks lengthened / shortened)

Synchronization Process

Clock synchronization is performed by means of SYNC Frames. Only preconfigured nodes (sync nodes) are allowed to send SYNC Frames. In a two-channel cluster a sync node has to send its SYNC Frame on both channels.

For synchronization in FlexRay™ the following constraints have to be considered:

- Max. one SYNC Frame per node in one communication cycle
- Max. 15 SYNC Frames per cluster in one communication cycle
- Every node has to use all available SYNC Frames for clock synchronization
- Minimum of two sync nodes required for clock synchronization and startup

For clock synchronization the time difference between expected and observed arrival time of SYNC Frames received during the static segment, valid on both channels (two-channel cluster), is measured. The calculation of correction terms is done during network idle time (NIT) (offset: every cycle, rate: odd cycle) by using a FTA / FTM algorithm. For details see FlexRay™ protocol specification v2.1, chapter 8.

SYNC Frame Transmission

SYNC Frame transmission is only possible from buffer 0 and 1. Message Buffer 1 may be used for SYNC Frame transmission in case that SYNC Frames should have different payloads on the two channels. In this case bit MRC.SPLM has to be programmed to 1.

Message Buffers used for SYNC Frame transmission have to be configured with the key slot ID and can be (re)configured in “DEFAULT_CONFIG” or “CONFIG” state only. For nodes transmitting SYNC Frames SUCC1.TXSY must be set to 1.

41.3.4.4 External Clock Synchronization

During normal operation, independent clusters can drift significantly. If synchronous operation across independent clusters is desired, external synchronization is necessary; even though the nodes within each cluster are synchronized. This can be accomplished with synchronous application of host-deduced rate and offset correction terms to the clusters.

- External offset / rate correction value is a signed integer
- External offset / rate correction value is added to calculated offset / rate correction value
- Aggregated offset / rate correction term (external + internal) is not checked against configured limits

41.3.5 Error Handling

The implemented error handling concept is intended to ensure that in case of a lower layer protocol error in a single node communication between non-affected nodes can be maintained. In some cases, higher layer program command activity is required for the Communication Controller to resume normal operation. A change of the error handling state will set bit EIR.PEMC in the Error Service Request Register and may trigger an service request to the Host if enabled. The actual error mode is signalled by CCEV.ERRM in the Communication Controller Error Vector register.

Table 389 Error Modes of the POC (Degradation Model)

Error Mode	Activity
ACTIVE (green)	**Full operation**, State: “NORMAL_ACTIVE” The Communication Controller is fully synchronized and supports the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers EIR and SIR.
PASSIVE (yellow)	**Reduced operation**, State: “NORMAL_PASSIVE”, Communication Controller self rescue allowed The Communication Controller stops transmitting Frames and symbols, but received Frames are still processed. Clock synchronization mechanisms are continued based on received Frames. No active contribution to the cluster wide clock synchronization. The host is informed of any error condition(s) or status change by interrupt (if enabled) or by reading the error and status interrupt flags from registers EIR and SIR.
COMM_HALT (red)	**Operation halted**, State: “HALT”, Communication Controller self rescue not allowed The Communication Controller stops Frame and symbol processing, clock synchronization processing, and the Macrotick generation. The host has still access to error and status information by reading the error and status interrupt flags from registers EIR and SIR. The bus drivers are disabled.

41.3.5.1 Clock Correction Failed Counter

When the Clock Correction Failed Counter reaches the maximum “without clock correction passive” limit defined by SUCC3.WCP, the POC transits from “NORMAL_ACTIVE” to “NORMAL_PASSIVE” state. When it reaches the “maximum without clock correction fatal” limit defined by SUCC3.WCF, it transits “NORMAL_ACTIVE” or “NORMAL_PASSIVE” to the “HALT” state. Both limits are defined in the SUC Configuration Register 3.

The Clock Correction Failed Counter CCEV.CCFC allows the Host to monitor the duration of the inability of a node to compute clock correction terms after the Communication Controller passed protocol startup phase. It will be incremented by one at the end of any **odd** numbered communication cycle where either the Missing Offset Correction signal SFS.MOCS nor the Missing Rate Correction signal SFS.MRCS flag is set. The two flags are located in the SYNC Frame Status register, while the Clock Correction Failed Counter is located in the Communication Controller Error Vector register.

The Clock Correction Failed Counter is reset to zero at the end of an **odd** communication cycle if neither the Missing Offset Correction signal SFS.MOCS nor the Missing Rate Correction signal SFS.MRCS flag is set.

The Clock Correction Failed Counter stops incrementing when the “maximum without clock correction fatal” value SUCC3.WCF as defined in the SUC Configuration Register 3 is reached (i.e. incrementing the counter at its maximum value will not cause it to “wraparound” back to zero). The Clock Correction Failed Counter is initialized to zero when the Communication Controller enters “READY” state or when “NORMAL_ACTIVE” state is entered.

41.3.5.2 Passive to Active Counter

The passive to active counter controls the transition of the POC from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. SUCC1.PTA in the SUC Configuration Register 1 defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the Communication Controller is allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. If SUCC1.PTA is reset to zero the Communication Controller is not allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state.

41.3.5.3 HALT Command

In case the Host wants to stop FlexRay™ communication of the local node it can bring the Communication Controller into “HALT” state by asserting the HALT command. This can be done by writing SUCC1.CMD = 0110_B in the SUC Configuration Register 1. When called in “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state the POC transits to “HALT” state at the end of the current cycle. When called in any other state SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED” and bit EIR.CNA in the Error Service Request Register is set to 1. If enabled an service request to the Host is generated.

41.3.5.4 FREEZE Command

In case the Host detects a severe error condition it can bring the Communication Controller into “HALT” state by asserting the FREEZE command. This can be done by writing SUCC1.CMD = 0111_B in the SUC Configuration Register 1. The FREEZE command triggers the entry of the “HALT” state immediately regardless of the actual POC state.

The POC state from which the transition to HALT state took place can be read from CCSV.PSL.

41.3.6 Communication Controller States

This chapter introduces the states of the Communication Controller.

41.3.6.1 Communication Controller State Diagram

State transitions are controlled by externals the application reset or RXDA/B, by the POC state machine, and by the CHI Command Vector SUCC1.CMD located in the SUC Configuration Register 1.

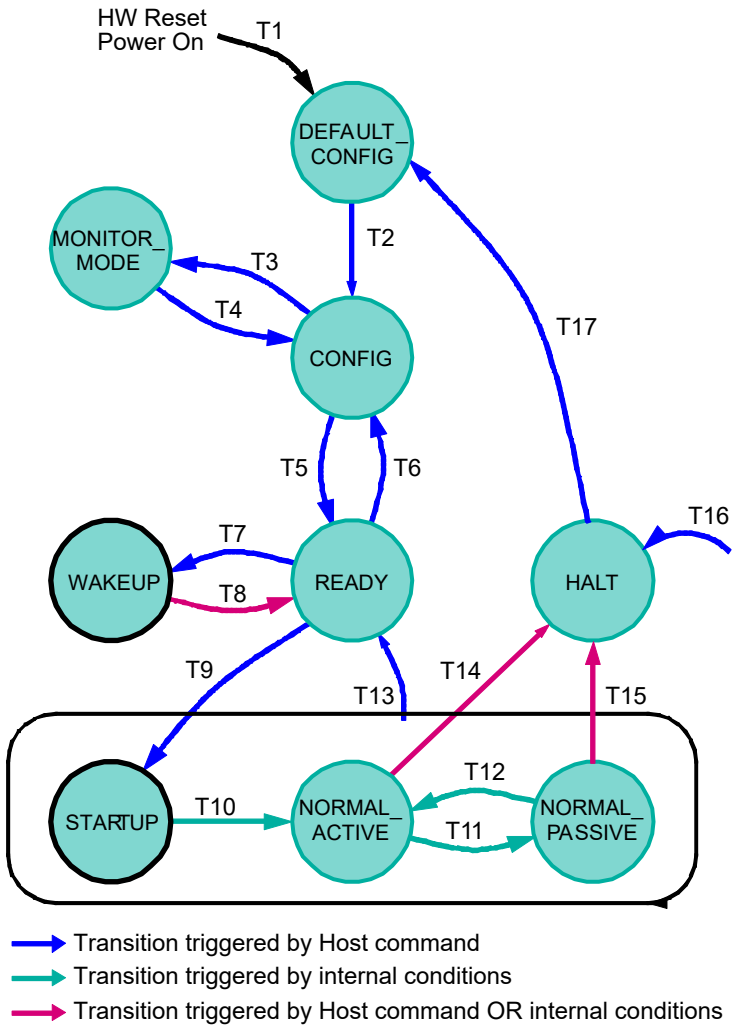


Figure 610 Overall State Diagram of E-Ray Communication Controller

The Communication Controller exits from all states to “HALT” state after application of the FREEZE command (SUCC1.CMD = 0111_B).

Table 390 State Transitions of E-Ray Overall State Machine

T#	Condition	From	To
1	application reset	HW Reset	DEFAULT_CONFIG
2	Command CONFIG, SUCC1.CMD = 0001_B	DEFAULT_CONFIG	CONFIG
3	Unlock sequence followed by command MONITOR_MODE, SUCC1.CMD = 1011_B	CONFIG	MONITOR_MODE
4	Command CONFIG, SUCC1.CMD = 0001_B	MONITOR_MODE	CONFIG
5	Unlock sequence followed by command READY, SUCC1.CMD = 0010_B	CONFIG	READY
6	Command CONFIG, SUCC1.CMD = 0001_B	READY	CONFIG
7	Command WAKEUP, SUCC1.CMD = 0011_B	READY	WAKEUP
8	Complete, non-aborted transmission of wakeup pattern OR received WUP OR received Frame Header OR command READY, SUCC1.CMD = 0010_B	WAKEUP	READY
9	Command RUN SUCC1.CMD = 0100_B	READY	STARTUP
10	Successful startup	STARTUP	NORMAL_ACTIVE
11	Clock Correction Failed counter reached Maximum Without Clock Correction Passive limit configured by WCP in SUC Configuration Register 3	NORMAL_ACTIVE	NORMAL_PASSIVE
12	Number of valid correction terms reached the Passive to Active limit configured by PTA in SUC Configuration Register 1	NORMAL_PASSIVE	NORMAL_ACTIVE
13	Command READY, SUCC1.CMD = 0010_B	STARTUP, NORMAL_ACTIVE, NORMAL_PASSIVE	READY
14	Clock Correction Failed counter reached Maximum Without Clock Correction Fatal limit configured by WCF in SUC Configuration Register 3 AND bit HCSE in the SUC Configuration Register 1 set to 1 OR command HALT, SUCC1.CMD = 0110_B	NORMAL_ACTIVE	HALT
15	Clock Correction Failed counter reached Maximum Without Clock Correction Fatal limit configured by WCF in SUC Configuration Register 3 AND bit HCSE in the SUC Configuration Register 1 set to 1 OR command HALT, SUCC1.CMD = 0110_B	NORMAL_PASSIVE	HALT

Table 390 State Transitions of E-Ray Overall State Machine (cont'd)

T#	Condition	From	To
16	Command FREEZE, SUCC1.CMD = 0111_B	All States	HALT
17	Command CONFIG, SUCC1.CMD = 0001_B	HALT	DEFAULT_CONFIG

41.3.6.2 DEFAULT_CONFIG State

In "DEFAULT_CONFIG" state, the Communication Controller is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state.

The Communication Controller enters this state

- When leaving application reset
- When exiting from "HALT" state

To leave "DEFAULT_CONFIG" state the Host has to write SUCC1.CMD = 0001_B in the SUC Configuration Register 1. The Communication Controller transits to "CONFIG" state.

41.3.6.2.1 CONFIG State

In "CONFIG" state, the Communication Controller is stopped. All configuration registers are accessible and the pins to the physical layer are in their inactive state. This state is used to initialize the Communication Controller configuration.

The Communication Controller enters this state

- When exiting from "DEFAULT_CONFIG" state
- When exiting from "MONITOR_MODE" or "READY" state

When the state has been entered via "HALT" and "DEFAULT_CONFIG" state, the Host can analyze status information and configuration. Before leaving "CONFIG" state the Host has to assure that the configuration is fault-free.

To leave "CONFIG" state, the Host has to perform the unlock sequence as described on "LCK". Directly after unlocking the "CONFIG" state the Host has to write SUCC1.CMD in the SUC Configuration Register 1 to enter the next state.

Internal counters and the Communication Controller status flags are reset when the Communication Controller leaves "CONFIG".

Note: The Message Buffer Status Registers (MHDS, TXRQ1 to TXRQ4, NDAT1 to NDAT4, MBSC1 to MBSC4) and status data stored in the Message RAM and are not affected by the transition of the POC from "CONFIG" to "READY" state.

When the Communication Controller is in "CONFIG" state it is also possible to bring the Communication Controller into a power saving mode by halting the module clocks (f_{SCLK}, f_{CLC_ERAY}). To do this the Host has to assure that all Message RAM transfers have finished before turning off the clocks.

41.3.6.3 MONITOR_MODE

After unlocking "CONFIG" state and writing SUCC1.CMD = 1011_B the Communication Controller enters "MONITOR_MODE". In this mode the Communication Controller is able to receive FlexRay™ Frames and to detect wakeup pattern. The temporal integrity of received Frames is not checked, and therefore cycle counter filtering is not supported. It is not possible to distinguish between static and dynamic frames, because limited functions

in Monitor Mode (FRF.RSS will be ignored, filtering not functional).This mode can be used for debugging purposes in case e.g. that startup of a FlexRay™ network fails. After writing SUCC1.CMD = 0001_B the Communication Controller transits back to “CONFIG” state.

In MONITOR_MODE the pick first valid mechanism is disabled. This means that a receive Message Buffer may only be configured to receive on one channel. Received Frames are stored into Message Buffers according to Frame ID and receive channel. NULL Frames are handled like Data Frames. After Frame reception only status bits MBS.VFRA, MBS, MBS.MLST, MBS.RCIS, MBS.SFIS, MBS.SYNS, MBS.NFIS, MBS.PPIS, MBS.RESS have valid value.

In “MONITOR_MODE” the Communication Controller is not able to distinguish between CAS and MTS symbols. In case one of these symbols is received on one or both of the two channels, the flags SIR.MTSA resp. SIR.MTSB are set. SIR.CAS has no function in “MONITOR_MODE”.

41.3.6.4 READY State

After unlocking “CONFIG” state and writing SUCC1.CMD = 0010_B the Communication Controller enters “READY” state. From this state the Communication Controller can transit to WAKEUP state and perform a cluster wakeup or to “STARTUP” state to perform a coldstart or to integrate into a running communication.

The Communication Controller enters this state

- When exiting from “CONFIG”, “WAKEUP”, “STARTUP”, “NORMAL_ACTIVE”, or “NORMAL_PASSIVE” state by writing SUCC1.CMD = 0010_B (READY command).

The Communication Controller exits from this state

- To “CONFIG” state by writing SUCC1.CMD = 0001_B (CONFIG command)
- To “WAKEUP” state by writing SUCC1.CMD = 0011_B (WAKEUP command)
- To “STARTUP” state by writing SUCC1.CMD = 0100_B (RUN command)

Internal counters and the Communication Controller status flags are reset when the Communication Controller enters “STARTUP” state.

Note: Status bits MHDS, registers TXRQ1 to TXRQ4, and status data stored in the Message RAM are not affected by the transition of the POC from “READY” to “STARTUP” state.

41.3.6.5 WAKEUP State

The description below is intended to help configuring wakeup for the E-Ray IP-module. A detailed description of the wakeup procedure together with the respective SDL diagrams can be found in the FlexRay™ protocol specification v2.1, section 7.1.

The Communication Controller enters this state

- When exiting from “READY” state by writing SUCC1.CMD = 0011_B (WAKEUP command).

The Communication Controller exits from this state to “READY” state

- After complete non-aborted transmission of wakeup pattern
- After WUP reception
- After detecting a WUP collision
- After reception of a Frame Header
- By writing SUCC1.CMD = 0010_B (READY command)

The cluster wakeup must precede the communication startup in order to ensure that all mechanisms defined for the startup work properly. The minimum requirement for a cluster wakeup is that all bus drivers are supplied with power. A bus driver has the ability to wake up the other components of its node when it receives a wakeup pattern on its channel. At least one node in the cluster needs an **external** wakeup source.

The Host completely controls the wakeup procedure. It is informed about the state of the cluster by the bus driver and the Communication Controller and configures bus guardian (if available) and Communication Controller to perform the cluster wakeup. The Communication Controller provides to the Host the ability to transmit a special wakeup pattern on each of its available channels separately. The Communication Controller needs to recognize the wakeup pattern only during “WAKEUP” state.

Wakeup may be performed on only one channel at a time. The Host has to configure the wakeup channel while the Communication Controller is in “CONFIG” state by writing bit SUCC1.WUCS in the SUC Configuration Register 1. The Communication Controller ensures that ongoing communication on this channel is not disturbed. The Communication Controller cannot guarantee that all nodes connected to the configured channel awake upon the transmission of the wakeup pattern, since these nodes cannot give feedback until the startup phase. The wakeup procedure enables single-channel devices in a two-channel system to trigger the wakeup, by only transmitting the wakeup pattern on the single channel to which they are connected. Any coldstart node that deems a system startup necessary will then wake the remaining channel before initiating communication startup.

The wakeup procedure tolerates any number of nodes simultaneously trying to wakeup a single channel and resolves this situation such that only one node transmits the pattern. Additionally the wakeup pattern is collision resilient, so even in the presence of a fault causing two nodes to simultaneously transmit a wakeup pattern, the resulting collided signal can still wake the other nodes.

After wakeup the Communication Controller returns to “READY” state and signals the change of the wakeup status to the Host by setting bit SIR.WST in the Status Service Request Register. The wakeup status vector can be read from the Communication Controller Status Vector register CCSV.WSV. If a valid wakeup pattern was received also either flag SIR.WUPA or flag SIR.WUPB in the Status Service Request Register is set.

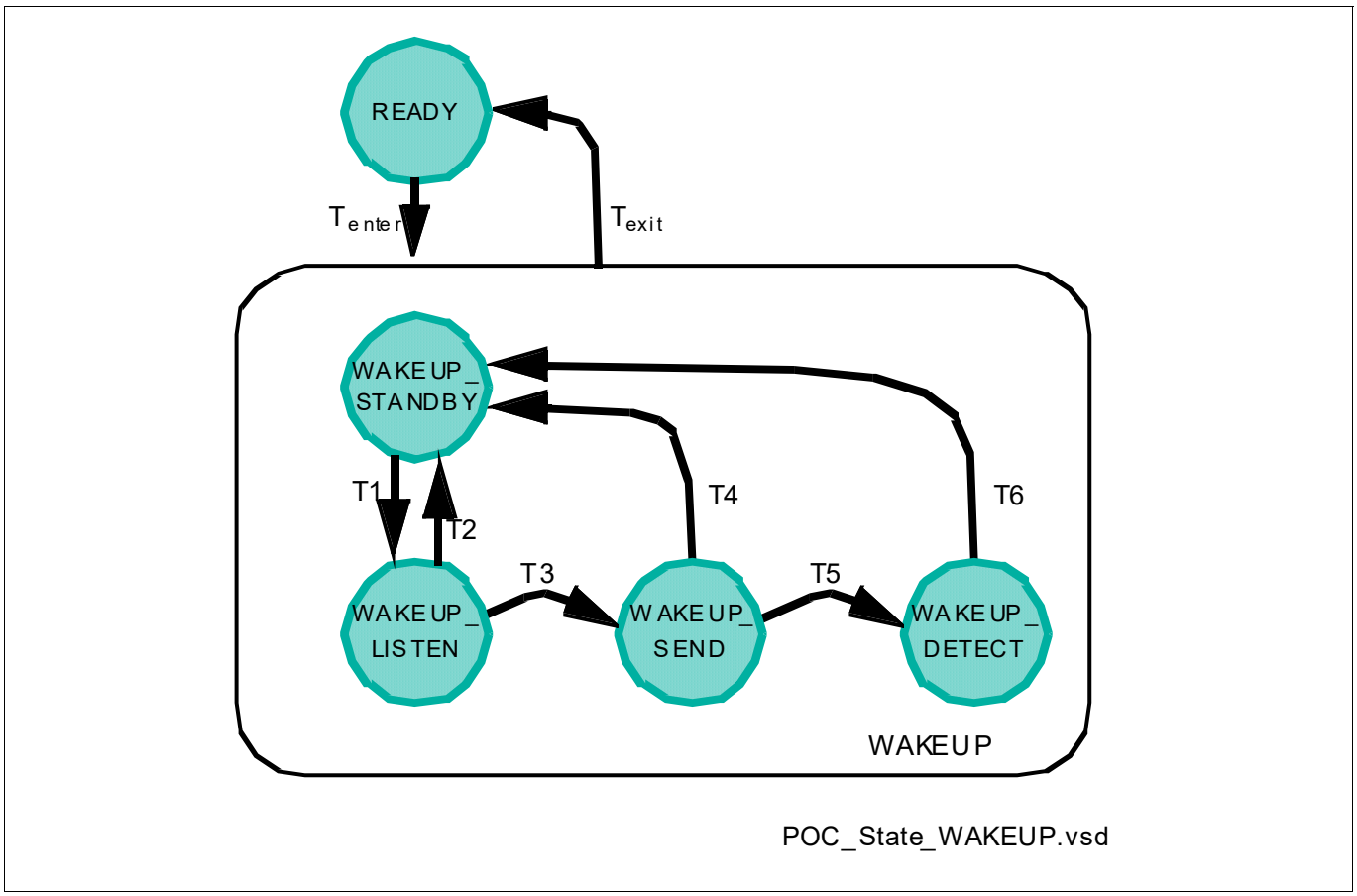


Figure 611 Structure of POC State WAKEUP

Table 391 State Transitions WAKEUP

T#	Condition	From	To
enter	Host commands change to “WAKEUP” state by writing SUCC1.CMD = 0011_B (WAKEUP command)	READY	WAKEUP
1	CHI command WAKEUP triggers wakeup FSM to transit to “WAKEUP_LISTEN” state	WAKEUP_ STANDBY	WAKEUP_LISTEN
2	Received WUP on wakeup channel selected by flag SUCC1.WUCS in the SUC Configuration Register 1 OR Frame Header on either available channel	WAKEUP_ LISTEN	WAKEUP_STANDBY
3	Timer event	WAKEUP_ LISTEN	WAKEUP_SEND
4	Complete, non-aborted transmission of wakeup pattern	WAKEUP_ SEND	WAKEUP_STANDBY
5	Collision detected	WAKEUP_ SEND	WAKEUP_DETECT
6	Wakeup timer expired OR WUP detected on wakeup channel selected by flag SUCC1.WUCS in the SUC Configuration Register 1 OR Frame Header received on either available channel	WAKEUP_ DETECT	WAKEUP_STANDBY
exit	Wakeup completed (after T2 or T4 or T6) OR Host commands change to “READY” state by writing SUCC1.CMD = 0010_B (READY command). This command also resets the wakeup FSM to “WAKEUP_STANDBY” state	WAKEUP	READY

The “WAKEUP_LISTEN” state is controlled by the wakeup timer and the wakeup noise timer. The two timers are controlled by the parameters listen time-out SUCC2.LT and listen time-out noise SUCC2.LTN. Both values can be configured in the SUC Configuration Register 2. listen time-out enables a fast cluster wakeup in case of a noise free environment, while listen time-out noise enables wakeup under more difficult conditions regarding noise interference.

In “WAKEUP_SEND” state the Communication Controller transmits the wakeup pattern on the configured channel and checks for collisions. After return from wakeup the Host has to bring the Communication Controller into “STARTUP” state by CHI command RUN.

In “WAKEUP_DETECT” state the Communication Controller attempts to identify the reason for the wakeup collision detected in “WAKEUP_SEND” state. The monitoring is bounded by the expiration of listen time-out as configured by SUCC2.LT in the SUC Configuration Register 2. Either the detection of a wakeup pattern indicating a wakeup attempt by another node or the reception of a Frame Header indication existing communication, causes the direct transition to “READY” state. Otherwise WAKEUP_DETECT is left after expiration of listen time-out; in this case the reason for wakeup collision is unknown.

The Host has to be aware of possible failures of the wakeup and act accordingly. It is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal time it takes another coldstart node to become awake and to be configured.

The FlexRay™ Protocol Specification v2.1 recommends that two different Communication Controllers shall awake the two channels.

Host activities

The host must coordinate the wakeup of the two channels and must decide whether, or not, to wake a specific channel. The sending of the wakeup pattern is initiated by the Host and generated by the Communication Controller. The wakeup pattern is detected by the remote BDs and signalled to their local Hosts.

Wakeup procedure controlled by Host (single-channel wakeup):

- Configure the Communication Controller in “CONFIG” state
  - Select wakeup channel by programming bit SUCC1.WUCS
- Check local BDs whether a WUP was received
- Activate BD of selected wakeup channel
- Command Communication Controller to start wakeup on the configured channel by writing SUCC1.CMD = 0011_B
  - Communication Controller enters “WAKEUP
  - Communication Controller returns to “READY” state and signals status of wakeup attempt to Host
- Wait predefined time to allow the other nodes to wakeup and configure themselves
- Coldstart node: wait for WUP on the other channel
  - In a dual channel cluster wait for WUP on the other channel
  - Reset coldstart inhibit flag CCSV.CSI by writing SUCC1.CMD = 1001_B (ALLOW_COLDSTART command))
- Reset Coldstart Inhibit flag CCSV.CSI in the CCSV register by writing SUCC1.CMD = 1001_B (ALLOW_COLDSTART command), coldstart node only
- Command Communication Controller to enter startup by writing SUCC1.CMD = 0100_B (RUN command)

Wakeup procedure triggered by BD:

- Wakeup recognized by BD
- BD triggers power-up of Host (if required)
- BD signals wakeup event to Host
- Host configures its local Communication Controller
- If necessary Host commands wakeup of second channel and waits predefined time to allow the other nodes to wakeup and configure themselves
- Host commands Communication Controller to enter “STARTUP” state by writing SUCC1.CMD = 0100_B (RUN command)

Wakeup pattern (WUP)

The wakeup pattern is composed of at least two wakeup symbols (WUS). Wakeup symbol and wakeup pattern are configured by the PRT Configuration Registers PRTC1 and PRTC2.

- Single channel wakeup, wakeup symbol may not be sent on both channels at the same time
- Wakeup symbol collision resilient for up to two sending nodes (two overlapping wakeup symbols still recognizable)
- Wakeup symbol must be configured identical in all nodes of a cluster
- Wakeup symbol transmit low time configured by PRTC2.TXL
- Wakeup symbol idle time used to listen for activity on the bus, configured by PRTC2.TXI
- A wakeup pattern composed of at least two Tx-wakeup symbols needed for wakeup
- Number of repetitions configurable by PRTC1.RWP (2 to 63 repetitions)
- Wakeup symbol receive window length configured by PRTC1.RXW
- Wakeup symbol receive low time configured by PRTC2.RXL
- Wakeup symbol receive idle time configured by PRTC2.RXI

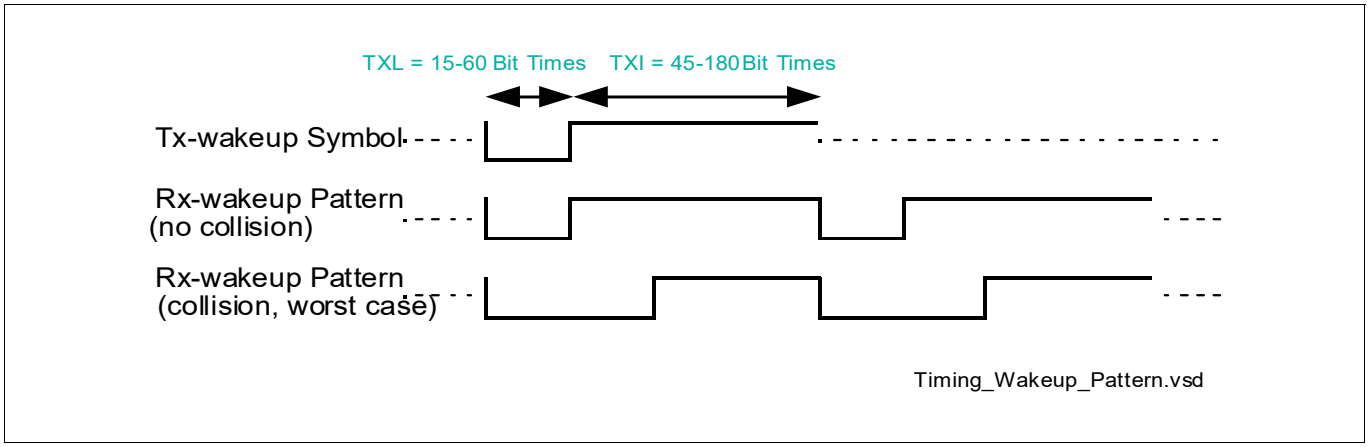


Figure 612 Timing of Wakeup Pattern

41.3.6.6 STARTUP State

The description below is intended to help configuring startup for the E-Ray IP-module. A detailed description of the startup procedure together with the respective SDL diagrams can be found in the FlexRay™ protocol specification v2.1, section 7.2.

Any node entering “STARTUP” state that has coldstart capability should assure that both channels attached have been awakened before initiating coldstart.

It cannot be assumed that all nodes and stars need the same amount of time to become completely awake and to be configured. Since at least two nodes are necessary to start up the cluster communication, it is advisable to delay any potential startup attempt of the node having instigated the wakeup by the minimal amount of time it takes another coldstart node to become awake, to be configured and to enter startup. It may require several hundred milliseconds (depending on the hardware used) before all nodes and stars are completely awakened and configured.

Startup is performed on all channels synchronously. During startup, a node only transmits startup Frames.

A fault-tolerant, distributed startup strategy is specified for initial synchronization of all nodes. In general, a node may enter “NORMAL_ACTIVE” state via (see **Figure 613**):

- Coldstart path initiating the schedule synchronization (leading coldstart node)
- Coldstart path joining other coldstart nodes (following coldstart node)
- Integration path integrating into an existing communication schedule (all other nodes)

A coldstart attempt begins with the transmission of a collision avoidance symbol (CAS). Only a coldstart node that had transmitted the CAS transmits Frames in the first four cycles after the CAS, it is then joined firstly by the other coldstart nodes and afterwards by all other nodes.

A coldstart node has the Transmit Startup Frame in Key Slot bit SUCC1.TXST and Transmit SYNC Frame in Key Slot bit SUCC1.TXSY in the SUC Configuration Register 1 set to 1. The Message Buffer 0 holds the key slot ID which defines the slot number where the Startup Frame is send. In the Frame Header of the Startup Frame the Startup Frame indicator bit is set.

In clusters consisting of three or more nodes, at least three nodes shall be configured to be coldstart nodes. In clusters consisting of two nodes, both nodes must be coldstart nodes. At least two fault-free coldstart nodes are necessary for the cluster to startup.

Each Startup Frame must also be a SYNC Frame; therefore each coldstart node will also be a sync node. The number of coldstart attempts is configured by SUCC1.CSA in the SUC Configuration Register 1.

A non-coldstart node requires at least two startup Frames from distinct nodes for integration. It may start integration before the coldstart nodes have finished their startup. It will not finish its startup until at least two coldstart nodes have finished their startup.

Both non-coldstart nodes and coldstart nodes start passive integration via the integration path as soon as they receive SYNC Frames from which to derive the TDMA schedule information. During integration the node has to adapt its own clock to the global clock (rate and offset) and has to make its cycle time consistent with the global schedule observable at the network. Afterwards, these settings are checked for consistency with all available network nodes. The node can only leave the integration phase and actively participate in communication when these checks are passed.

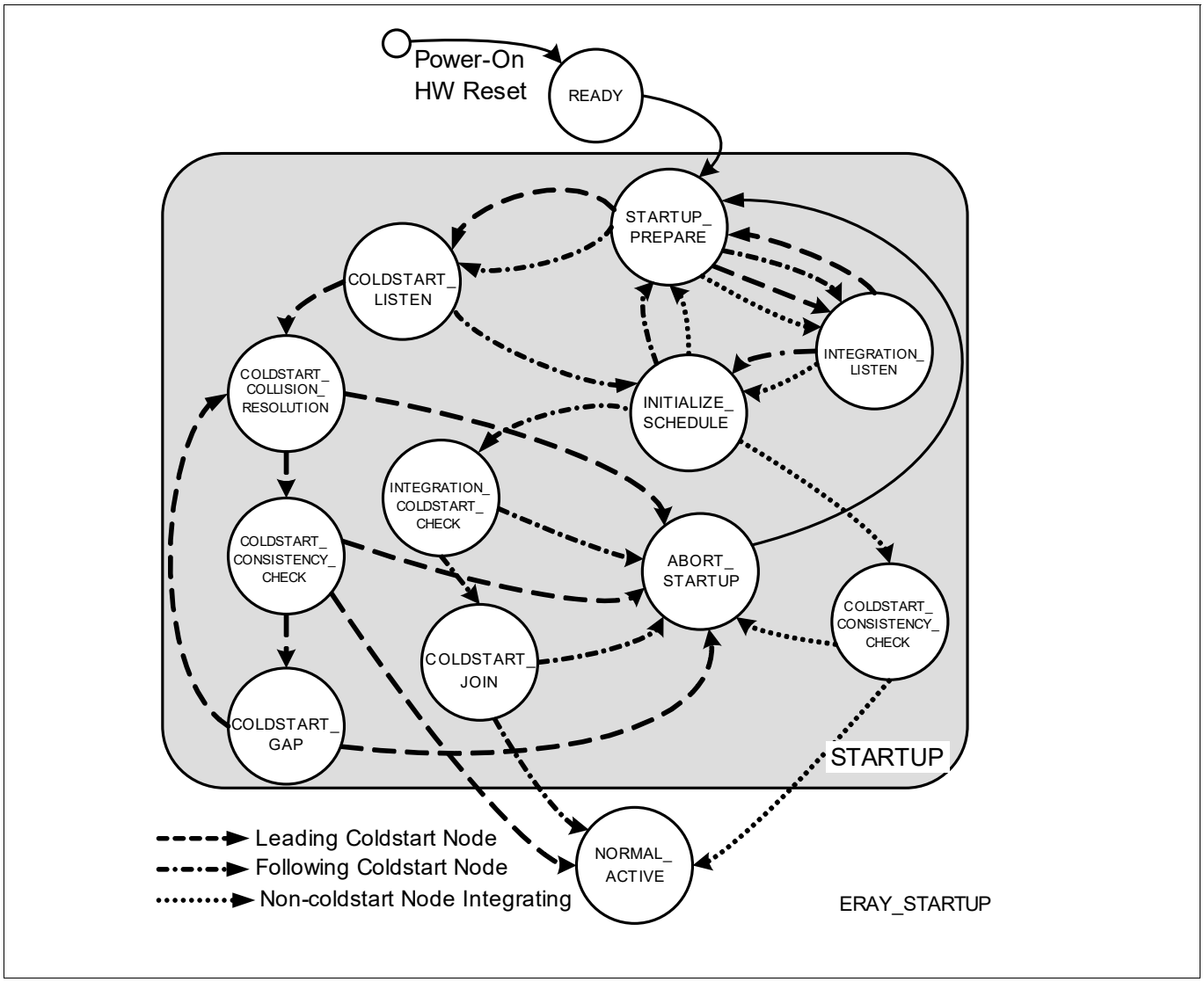


Figure 613 State Diagram Time-Triggered Startup

Coldstart Inhibit Mode

In coldstart inhibit mode the node is prevented from initializing the TDMA communication schedule. If bit CCSV.CSI in the Communication Controller Status Vector register is set, the node is not allowed to initialize the cluster communication, i.e. entering the coldstart path is prohibited. The node is allowed to integrate to a running cluster or to transmit startup Frames after another coldstart node started the initialization of the cluster communication.

The coldstart inhibit bit CCSV.CSI is set whenever the POC enters “READY” state. The bit has to be cleared under control of the Host by CHI command ALLOW_COLDSTART (SUCC1.CMD = 1001_B)

41.3.6.7 Startup Time-outs

The Communication Controller supplies two different Microtick timers supporting two time-out values, startup time-out and startup noise time-out. The two timers are reset when the Communication Controller enters the “COLDSTART_LISTEN” state. The expiration of either of these timers causes the node to leave the initial sensing phase (“COLDSTART_LISTEN” state) with the intention of starting up communication.

Note: The startup and startup noise timers are identical with the wakeup and wakeup noise timers and use the same configuration values SUCC2.LT and SUCC2.LTN from the SUC Configuration Register 2.

Startup Time-out

The startup time-out limits the listen time used by a node to determine if there is already communication between other nodes or at least one coldstart node actively requesting the integration of others.

The startup timer is configured by programming SUCC2.LT (pdListenTimeout) in the SUC Configuration Register 2.

The startup timer is restarted upon:

- Entering the “COLDSTART_LISTEN” state
- Both channels reaching idle state while in “COLDSTART_LISTEN” state

The startup timer is stopped:

- If communication channel activity is detected on one of the configured channels while the node is in the “COLDSTART_LISTEN” state
- When the “COLDSTART_LISTEN” state is left

Once the startup time-out expires, neither an overflow nor a cyclic restart of the timer is performed. The timer status is kept for further processing by the startup state machine.

Startup Noise Time-out

At the same time the startup timer is started for the first time (transition from “STARTUP_PREPARE” state to “COLDSTART_LISTEN” state), the startup noise timer is started. This additional time-out is used to improve reliability of the startup procedure in the presence of noise.

The startup noise timer is configured by programming SUCC2.LTN (gListenNoise - 1) in the SUC Configuration Register 2 (see “**SUCC2**).

The startup noise time-out is:
pdListenTimeout • ´gListenNoise = SUCC2.LT • (SUCC2.LTN + 1)

The startup noise timer is restarted upon:

- Entering the “COLDSTART_LISTEN” state
- Reception of correctly decoded Headers or CAS symbols while the node is in “COLDSTART_LISTEN” state

The startup noise timer is stopped when the “COLDSTART_LISTEN” state is left.

Once the startup noise time-out expires, neither an overflow nor a cyclic restart of the timer is performed. The status is kept for further processing by the startup state machine. Since the startup noise timer won’t be restarted when random channel activity is sensed, this time-out defines the fall-back solution that guarantees that a node will try to start up the communication cluster even in the presence of noise.

41.3.6.8 Path of leading Coldstart Node (initiating coldstart)

When a coldstart node enters “COLDSTART_LISTEN”, it listens to its attached channels.

If no communication is detected, the node enters the “COLDSTART_COLLISION_ RESOLUTION” state and commences a coldstart attempt. The initial transmission of a CAS symbol is succeeded by the first regular cycle. This cycle has the number zero.

From cycle zero on, the node transmits its startup Frame. Since each coldstart node is allowed to perform a coldstart attempt, it may occur that several nodes simultaneously transmit the CAS symbol and enter the coldstart path. This situation is resolved during the first four cycles after CAS transmission.

As soon as a node that initiates a coldstart attempt receives a CAS symbol or a Frame Header during these four cycles, it re-enters the “COLDSTART_LISTEN” state. Thereby, only one node remains in this path. In cycle four, other coldstart nodes begin to transmit their startup Frames.

After four cycles in “COLDSTART_COLLISION_RESOLUTION” state, the node that initiated the coldstart enters the “COLDSTART_CONSISTENCY_CHECK” state. It collects all startup Frames from cycle four and five and performs the clock correction. If the clock correction does not deliver any errors and it has received at least one valid Startup Frame pair, the node leaves “COLDSTART_CONSISTENCY_CHECK” and enters “NORMAL_ACTIVE” state.

The number of coldstart attempts that a node is allowed to perform is configured by SUCC1.CSA in the SUC Configuration Register 1. The number of remaining coldstarts attempts CCSV.RCA can be read from Communication Controller Status Vector register. The number of remaining attempts is reduced by one for each attempted coldstart. A node may enter the “COLDSTART_LISTEN” state only if this value is larger than one and it may enter the “COLDSTART_COLLISION_RESOLUTION” state only if this value is larger than zero. If the number of coldstart attempts is one, coldstart is inhibited but integration is still possible.

Path of following Coldstart Node (responding to leading Coldstart Node)

When a coldstart node enters the “COLDSTART_LISTEN” state, it tries to receive a valid pair of startup Frames to derive its schedule and clock correction from the leading coldstart node.

As soon as a valid Startup Frame has been received the “INITIALIZE_SCHEDULE” state is entered. If the clock synchronization can successfully receive a matching second valid Startup Frame and can derive a schedule from this startup Frames, the “INTEGRATION_COLDSTART_CHECK” state is entered.

In “INTEGRATION_COLDSTART_CHECK” state it is assured that the clock correction can be performed correctly and that the coldstart node from which this node has initialized its schedule is still available. The node collects all SYNC Frames and performs clock correction in the following double-cycle. If clock correction does not signal any errors and if the node continues to receive sufficient Frames from the same node it has integrated on, the “COLDSTART_JOIN” state is entered.

In “COLDSTART_JOIN” state integrating coldstart nodes begin to transmit their own startup Frames. Thereby the node that initiated the coldstart and the nodes joining it can check if their schedules agree to each other. If for the following three cycles the clock correction does not signal errors and at least one other coldstart node is visible, the node leaves “COLDSTART_JOIN” state and enters “NORMAL_ACTIVE” state. Thereby it leaves “STARTUP” at least one cycle after the node that initiated the coldstart.

Path of Non-coldstart Node

When a non-coldstart node enters the INTEGRATION_LISTEN state, it listens to its attached channels and tries to receive FlexRay™ Frames.

As soon as a valid Startup Frame has been received the “INITIALIZE_SCHEDULE” state is entered. If the clock synchronization can successfully receive a matching second valid Startup Frame and derive a schedule from this, the INTEGRATION_CONSISTENCY_ CHECK state is entered.

In “INTEGRATION_CONSISTENCY_CHECK” state it is verified that the clock correction can be performed correctly and that enough coldstart nodes (at least 2) send startup Frames that agree to the nodes own schedule. Clock correction is activated, and if any errors are signalled, the integration attempt is aborted.

During the first even cycle in this state, either two valid startup Frames or the Startup Frame of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

During the first double-cycle in this state, either two valid Startup Frame pairs or the Startup Frame pair of the node that this node has integrated on must be received; otherwise the node aborts the integration attempt.

If after the first double-cycle less than two valid startup Frames are received within an even cycle, or less than two valid Startup Frame pairs are received within a double-cycle, the startup attempt is aborted.

Nodes in this state need to see two valid Startup Frame pairs for two consecutive double-cycles each to be allowed to leave STARTUP and enter NORMAL_ACTIVE. Consequently, they leave startup at least one double-cycle after the node that initiated the coldstart and only at the end of a cycle with an odd cycle number.

41.3.6.9 NORMAL_ACTIVE State

As soon as the node that transmitted the first CAS symbol (resolving the potential access conflict and entering “STARTUP” via coldstart path) and one additional node have entered the “NORMAL_ACTIVE” state, the startup phase for the cluster has finished. In the “NORMAL_ACTIVE” state, all configured messages are scheduled for transmission. This includes all Data Frames as well as the SYNC Frames. Rate and offset measurement is started in all even cycles (even/odd cycle pairs required).

In “NORMAL_ACTIVE” state the Communication Controller supports regular communication functions

- The Communication Controller performs transmissions and reception on the FlexRay™ bus as configured
- Clock synchronization is running
- The Host interface is operational

The Communication Controller exits from that state to

- “HALT” state by writing SUCC1.CMD = 0110_B (HALT command, at the end of the current cycle)
- “HALT” state by writing SUCC1.CMD = 0111_B (FREEZE command, immediately)
- “HALT” state due to change of the error state from “ACTIVE” to “COMM_HALT”
- “NORMAL_PASSIVE” state due to change of the error state from “ACTIVE” to “PASSIVE”
- “READY” state by writing SUCC1.CMD = 0010_B (READY command)

41.3.6.10 NORMAL_PASSIVE State

“NORMAL_PASSIVE” state is entered from “NORMAL_ACTIVE” state when the error state changes from ACTIVE (green) to PASSIVE (yellow).

In “NORMAL_PASSIVE” state, the node is able to receive all Frames (node is fully synchronized and performs clock synchronization). In comparison to the “NORMAL_ACTIVE” state the node does not actively participate in communication, i.e. neither symbols nor Frames are transmitted.

In “NORMAL_PASSIVE” state

- The Communication Controller performs reception on the FlexRay™ bus
- The Communication Controller does not transmit any Frames or symbols on the FlexRay™ bus
- Clock synchronization is running
- The Host interface is operational

The Communication Controller exits from this state to

- “HALT” state by writing SUCC1.CMD = 0110_B (HALT command, at the end of the current cycle)
- “HALT” state by writing SUCC1.CMD = 0111_B (FREEZE command, immediately)

- “HALT” state due to change of the error state from “PASSIVE” to “COMM_HALT”
- “NORMAL_ACTIVE” state due to change of the error state from “PASSIVE” to “ACTIVE”. The transition takes place when CCEV.PTAC from the Communication Controller Error Vector register equals SUCC1.PTA - 1.
- “READY” state by writing SUCC1.CMD = 0010_B (READY command)

41.3.6.11 HALT State

In this state all communication (reception and transmission) is stopped.

The Communication Controller enters this state

- By writing SUCC1.CMD = 0110_B (HALT command) while the Communication Controller is in “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state
- By writing SUCC1.CMD = 0111_B (FREEZE command) from all states
- When exiting from “NORMAL_ACTIVE” state because the clock correction failed counter reached the “maximum without clock correction fatal” limit
- When exiting from “NORMAL_PASSIVE” state because the clock correction failed counter reached the “maximum without clock correction fatal” limit

The Communication Controller exits from this state to “DEFAULT_CONFIG” state

- By writing SUCC1.CMD = 0001_B (CONFIG command)

When the Communication Controller enters “HALT” state, all configuration and status data is maintained for analyzing purposes.

When the Host writes SUCC1.CMD = 0110_B (HALT command), the Communication Controller sets bit CCSV.HRQ in the Communication Controller Status Vector register and enters “HALT” state after the current communication cycle has finished.

When the Host writes SUCC1.CMD = 0111_B (FREEZE command), the Communication Controller enters “HALT” state immediately and sets the CCSV.FSI bit in the Communication Controller Status Vector register.

The POC state from which the transition to HALT state took place can be read from CCSV.PSL.

41.3.7 Network Management

The accrued Network Management (NM) vector is located in the Network Management Register 1 to Network Management Register 3 (NMVx (x = 1-3)). The Communication Controller performs a logical OR operation over all Network Management (NM) vectors out of all received valid Network Management (NM) Frames with the Payload Preamble Indicator (PPI) bit set. Only a static Frame may be configured to hold Network Management (NM) information. The Communication Controller updates the Network Management (NM) vector at the end of each cycle.

The length of the Network Management (NM) vector can be configured from 0 to 12 byte by NML in the NEM Configuration Register. The Network Management (NM) vector length must be configured identically in all nodes of a cluster.

To configure a transmit buffer to send FlexRay™ Frames with the PPI bit set, the PPIT bit in the Header Section of the respective transmit buffer has to be set via WRHS1.PPIT. In addition the Host has to write the Network Management (NM) information to the Data Section of the respective transmit buffer.

The evaluation of the Network Management (NM) vector has to be done by the application running on the Host.

Note: In case a Message Buffer is configured for transmission / reception of Network Management Frames, the payload length configured in Header 2 of that Message Buffer should be equal or greater than the length of the NM Vector configured by NEMC.NML.
When the Communication Controller transits to “HALT” state, the cycle count is not incremented and therefore the NM Vector is not updated. In this case NMV1 to NMV3 holds the value from the cycle before.

41.3.8 Filtering and Masking

Filtering is done by checking specific fields in a received Frame against the corresponding configuration constants of the valid Message Buffers and the actual slot and cycle counter values (acceptance filtering), or by

comparing the configuration constants of the valid Message Buffers against the actual slot and cycle counter values (transmit filtering). A Message Buffer is only updated / transmitted if the required matches occur.

Filtering is done on the following fields:

- Channel ID
- Frame ID
- Cycle Counter

The following filter combinations for acceptance / transmit filtering are allowed:

- Frame ID + Channel ID
- Frame ID + Channel ID + Cycle Counter

In order to store a received message in a Message Buffer all configured filters must match.

Note: For the FIFO the acceptance filter is configured by the FIFO Rejection Filter and the FIFO Rejection Filter Mask.

A message will be transmitted in the time slot corresponding to the configured Frame ID on the configured channel(s). If cycle counter filtering is enabled the configured cycle filter value must also match.

41.3.8.1 Frame ID Filtering

Every transmit and receive buffer contains a Frame ID stored in the Header Section. This Frame ID is used differently for receive and transmit buffers.

Receive Buffers

A received message is stored in the first receive buffer where the received Frame ID matches the configured Frame ID, provided channel ID and cycle counter criteria are also met.

Transmit Buffers

For transmit buffers the configured Frame ID is used to determine the appropriate slot for message transmission. The Frame will be transmitted in the time slot corresponding to the configured Frame ID, provided channel ID and cycle counter criteria are also met.

41.3.8.2 Channel ID Filtering

There is a 2-bit channel filtering field (CHA, CHB) located in the Header Section of each Message Buffer in the Message RAM. It serves as a filter for receive buffers, and as a control field for transmit buffers (see Table below).

Table 392 Channel Filtering Configuration

CHA	CHB	**Transmit Buffer**transmit Frame	**Receive Buffer**store valid receive Frame
1	1	on both channels (static segment only)	received on channel A or B (store first semantically valid Frame, static segment only)
1	0	on channel A	received on channel A
0	1	on channel B	received on channel B
0	0	no transmission	ignore Frame

Note: If a Message Buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no Frames are transmitted resp. received Frames are ignored (same function as CHA = CHB = 0)

Receive Buffers

Valid received Frames are stored if they are received on the channels specified in the channel filtering field. Only in static segment a receive buffer may be setup for reception on both channels (CHA and CHB set). Other filtering criteria must also be met.

If a valid Header Segment was stored, the respective MBC flag in the Message Buffer Status Changed register is set. If a valid Payload Segment was stored, the respective NDn (n = 0-31) to NDn (n = 96-127) flag in the New Data NDAT1 to NDAT4 register is set. In both cases, if bit RDHS1.MBI in the Header Section of the respective Message Buffer is set, the RXI flag in the Status Service Request Register is set to 1. If enabled an service request is generated.

Transmit Buffers

The content of the buffer is transmitted only on the channels specified in the channel filtering field when the Frame ID filtering and cycle counter filtering criteria are also met. Only in static segment a transmit buffer may be setup for transmission on both channels (CHA and CHB set). After transmission has completed, and if bit WRHS1.MBI in the Header Section of the respective Message Buffer is set, the TXI flag in the Status Service Request Register is set to 1. If enabled an service request is generated.

41.3.8.3 Cycle Counter Filtering

Cycle counter filtering is based on the notion of a cycle set. For filtering purposes, a match is detected if any one of the elements of the cycle set is matched. The cycle set is defined by the cycle code field in the Header Section of each Message Buffer.

If Message Buffer 0 is configured to hold the startup / SYNC Frame or the single slot Frame by bits TXST, TXSY, and TSM in the SUC Configuration Register 1, cycle counter filtering for Message Buffer 0 should be disabled.

*Note: Sharing of a static time slot via cycle counter filtering between different nodes of a FlexRay™ network is **not** allowed.*

The set of cycle numbers belonging to a cycle set is determined as described in Table below.

Table 393 Definition of Cycle Set

Cycle Code	Matching Cycle Counter Values		
$000000x_B$	all Cycles		
$000001c_B$	every second Cycle	at (Cycle Count)mod2	= c
$00001cc_B$	every fourth Cycle	at (Cycle Count)mod4	= cc
$0001ccc_B$	every eighth Cycle	at (Cycle Count)mod8	= ccc
$001cccc_B$	every sixteenth Cycle	at (Cycle Count)mod16	= cccc
$01ccccc_B$	every thirty-second Cycle	at (Cycle Count)mod32	= ccccc
$1cccccc_B$	every sixty-fourth Cycle	at (Cycle Count)mod64	= cccccc

Table below gives some examples for valid cycle sets to be used for cycle counter filtering:

Table 394 Examples for Valid Cycle Sets

Cycle Code	Matching Cycle Counter Values
0000011_B	1-3-5-7- ….-63 ↵
0000100_B	0-4-8-12- ….-60 ↵
0001110_B	6-14-22-30- ….-62 ↵
0011000_B	8-24-40-56 ↵
0100011_B	3-35 ↵
1001001_B	9 ↵

Receive Buffers

The received message is stored only if the received cycle counter matches an element of the receive buffer's cycle set. Channel ID and Frame ID criteria must also be met.

Transmit Buffers

The content of the buffer is transmitted on the configured channels when an element of the cycle set matches the current cycle counter value and the Frame ID matches the slot counter value.

41.3.8.4 FIFO Filtering

For FIFO filtering there is one rejection filter and one rejection filter mask available. The FIFO rejection filter consists of 20 bits for **Channel** (2 bits), **Frame ID** (11 bits), and **Cycle Code** (7 bits). Rejection filter and rejection filter mask can be configured in "DEFAULT_CONFIG" or "CONFIG" state only. The filter configuration in the Header Sections of Message Buffers belonging to the FIFO is ignored.

A valid received Frame is stored in the FIFO if channel ID, Frame ID, and cycle counter are not rejected by the configured rejection filter and rejection filter mask, and if there is no matching dedicated receive buffer.

41.3.9 Transmit Process

The transmit process is described in the following sections.

41.3.9.1 Static Segment

For the static segment, if there are several messages pending for transmission, the message with the Frame ID corresponding to the next sending slot is selected for transmission.

The Data Section of transmit buffers assigned to the static segment can be updated until the end of the preceding time slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

41.3.9.2 Dynamic Segment

In the dynamic segment, if several messages are pending, the message with the highest priority (lowest Frame ID) is selected next. Only Frame ID's which are higher than the largest static Frame ID are allowed for the dynamic segment.

In the dynamic segment different slot counter sequences are possible (concurrent sending of different Frame ID's on both channels). Therefore pending messages are selected according to their Frame ID and their channel configuration bit.

The Data Section of transmit buffers assigned to the dynamic segment can be updated until the end of the preceding slot. This means that a transfer from the Input Buffer has to be started by writing to the Input Buffer Command Request register latest at this time.

The start of latest transmit configured by SLT in the MHD Configuration Register 1 defines the maximum minislot value allowed before inhibiting new Frame transmission in the dynamic segment of the current cycle.

41.3.9.3 Transmit Buffers

A portion of the E-Ray Message Buffers can be configured as transmit buffers by programming bit CFG in the Header Section of the respective Message Buffer to 1. This can be done via the Write Header Section 1 register.

There exist the following possibilities to assign a transmit buffer to the Communication Controller channels:

- Static segment: channel A **or** channel B, channel A **and** channel B
- Dynamic segment: channel A **or** channel B

Message Buffer 0 is dedicated to hold the startup Frame, the SYNC Frame, or the designated single slot Frame as configured by TXST, TXSY, and TSM in the SUC Configuration Register 1. In this case it can be reconfigured in “DEFAULT_CONFIG” or “CONFIG” state only. This ensures that any node transmits at most one startup / SYNC Frame per communication cycle. Transmission of startup / SYNC Frames from other Message Buffers is not possible.

All other Message Buffers configured for transmission in static or dynamic segment are reconfigurable during runtime. Due to the organization of the Data Partition in the Message RAM (reference by data pointer), reconfiguration of the configured payload length and the data pointer in the Header Section of a Message Buffer may lead to erroneous configurations. If a Message Buffer is reconfigured during runtime it may happen that this Message Buffer is not send out in the respective communication cycle.

The Communication Controller does not have the capability to calculate the Header CRC. The Host is supposed to provide the Header CRCs for all transmit buffers. If Network Management is required the Host has to set the PPIT bit in the Header Section of the respective Message Buffer to 1 and write the Network Management information to the Data Section of the Message Buffer.

The payload length field configures the data payload length in 2-byte words. If the configured payload length of a static transmit buffer is shorter than the payload length configured for the static segment by SFDL in the Message Handler Configuration Register 1, the Communication Controller generates padding byte to ensure that Frames have proper physical length. The padding pattern is logical zero.

Each transmit buffer provides a transmission mode flag TXM that allows the Host to configure the transmission mode for the transmit buffer in the static segment. If this bit is set, the transmitter operates in the single-shot mode. If this bit is cleared, the transmitter operates in the continuous mode. In dynamic segment the transmitter always works in single-shot mode.

If a Message Buffer is configured in the continuous mode, the Communication Controller does not reset the transmission request flag TXR after successful transmission. In this case a Frame is sent out each time the Frame ID and cycle counter filter match. The TXR flag can be reset by the Host by writing the respective Message Buffer number to the Input Buffer Command Request register while bit STXRH in the Input Buffer Command Mask register is reset to 0.

If two or more transmit buffers are configured with the same Frame ID **and** cycle counter filter value, the transmit buffer with the lowest Message Buffer number will be transmitted in the respective slot.

41.3.9.4 Frame Transmission

To prepare a transmit buffer for transmission the following steps are required:

- Configure the Message Buffer as transmit buffer by writing bit CFG = 1 in the Write Header Section 1 register
- Write transmit message (Header and Data Section) to the Input Buffer.
- To transfer a transmit message from Input Buffer to the Message RAM proceed as described on “Data Transfer from Input Buffer to Message RAM”.
- If configured in the Input Buffer Command Mask register the Transmission Request flag for the respective Message Buffer will be set as soon as the transfer has completed, and the Message Buffer is ready for transmission.
- Check whether the Message Buffer has been transmitted by checking the TXR bits (TXR = 0) in the Transmission Request 1 to 4 registers (single-shot mode only).

In single-shot mode the Communication Controller resets the TXR flag after transmission has been completed. Now the Host may update the transmit buffer with the next message. The Communication Controller does not

transmit the message before the Host has indicated that the update is completed by setting the Transmission Request flag TXR again. The Host can check the actual state of the TXR flags of all Message Buffers by reading the Transmission Request registers. After successful transmission, if bit WRHS1.MBI in the Header Section of the respective Message Buffer is set, the transmit service request flag in the Status Service Request Register is set (TXI = 1). If enabled an service request is generated.

41.3.9.5 NULL Frame Transmission

If in static segment the Host does not set the transmission request flag before transmit time, and if there is no other transmit buffer with matching filter criteria (matching Frame ID and cycle counter filter), the Communication Controller transmits a NULL Frame with the NULL Frame indication bit reset to 0 and the payload data reset to zero.

In the following cases the Communication Controller transmits a NULL Frame with the NULL Frame indication bit reset to 0, and the rest of the Frame Header and the Frame length unchanged (payload data is reset to zero):

- All transmit buffers configured for the slot have cycle counter filters that do not match the current cycle
- There are matching Frame ID's and cycle counter filters, but none of these transmit buffers has the transmission request flag TXR set

NULL Frames are not transmitted in the dynamic segment.

41.3.10 Receive Process

The receive process is described in the following sections.

41.3.10.1 Frame Reception

To prepare or change a Message Buffer for reception the following steps are required:

- Configure the Message Buffer as receive buffer by writing bit CFG = 0 in the Write Header Section 1 register
- Configure the receive buffer by writing the configuration data (Header Section) to the Input Buffer
- Transfer the configuration from Input Buffer to the Message RAM by writing the number of the target Message Buffer to the Input Buffer Command Request register.

Once these steps are performed, the Message Buffer functions as an active receive buffer and participates in the internal acceptance filtering process, which takes place every time the Communication Controller receives a message. The first matching receive buffer is updated from the received message. If the Message Buffer holds an unprocessed Data Section (ND = 1) it is overwritten with the new message and the MLST bit in the respective Message Buffer Status register is set.

If the payload length of a received Frame PLC is longer than the value programmed by PLC in the Header Section of the respective Message Buffer, the data field stored in the Message Buffer is truncated to that length.

If no Frame, a NULL Frame, or a corrupted Frame is received in a slot, the Data Section of the Message Buffer configured for this slot is not updated. In this case only the flags in the Message Buffer Status register are updated to signal the cause of the problem. In addition the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set.

When the Data Section of a receive buffer has been updated from a received Frame, the respective New Data NDn (n = 0-31) to NDn (n = 96-127) flag in the New Data NDAT1 to NDAT4 registers is set. When the Message Handler has updated the Message Buffer status, the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set. If bit RDHS1.MBI in the Header Section of the respective Message Buffer is set, the receive service request flag in the Status Service Request Register is set (RXI = 1). If enabled an service request is generated.

To read a receive buffer from the Message RAM via the Output Buffer proceed as described on “Data Transfer from Message RAM to Output Buffer”.

Note: The ND and MBC flags are automatically cleared by the Message Handler when the received message has been transferred to the Output Buffer.

41.3.10.2 NULL Frame reception

The Payload Segment of a received NULL Frame is **not** copied into the matching receive buffer. If a NULL Frame has been received, the Header Section of the matching Message Buffer is updated from the received NULL Frame. The NULL Frame indication bit in the Header Section 3 of the respective Message Buffer is reset (NFI = 0) and the respective MBC flag in the Message Buffer Status Changed 1,2,3,4 registers is set.

In case that bit ND and / or MBC were already set before this event because the Host did not read the last received message, bit MLST in the Message Buffer Status register of the respective Message Buffer is also set.

41.3.11 FIFO Function

A group of the Message Buffers can be configured as a cyclic First-In-First-Out (FIFO). The group of Message Buffers belonging to the FIFO is contiguous in the register map starting with the Message Buffer referenced by FFB and ending with the Message Buffer referenced by LCB in the Message RAM Configuration register. Up to 128 Message Buffers can be assigned to the FIFO.

41.3.11.1 Description

Every valid incoming message not matching with any dedicated receive buffer but passing the programmable FIFO filter is stored into the FIFO. In this case Frame ID, payload length, receive cycle count, and the status bits of the addressed FIFO Message Buffer are overwritten with Frame ID, payload length, receive cycle count, and the status from the received message and can be read by the Host for message identification. Bit RFNE in the Status Service Request Register shows that the FIFO is not empty, bit RFCL in the Status Service Request Register is set when the last available Message Buffer belonging to the FIFO is written, bit RFO in the Error Service Request Register shows that a FIFO overrun has been detected. If enabled, service requests are generated.

There are two index registers associated with the FIFO. The PUT Index Register (PIDX) is an index to the next available location in the FIFO. When a new message has been received it is written into the Message Buffer addressed by the PIDX register. The PIDX register is then incremented and addresses the next available Message Buffer. If the PIDX register is incremented past the highest numbered Message Buffer of the FIFO, the PIDX register is loaded with the number of the first (lowest numbered) Message Buffer in the FIFO chain. The GET Index Register (GIDX) is used to address the next Message Buffer of the FIFO to be read. The GIDX register is incremented after transfer of the contents of a Message Buffer belonging to the FIFO to the Output Buffer. The PUT Index Register and the GET Index Register are not accessible by the Host.

The FIFO is completely filled when the PUT index (PIDX) reaches the value of the GET index (GIDX). When the next message is written to the FIFO before the oldest message has been read, both PUT index and GET index are incremented and the new message overwrites the oldest message in the FIFO. This will set FIFO overrun flag RFO in the Error Service Request Register.

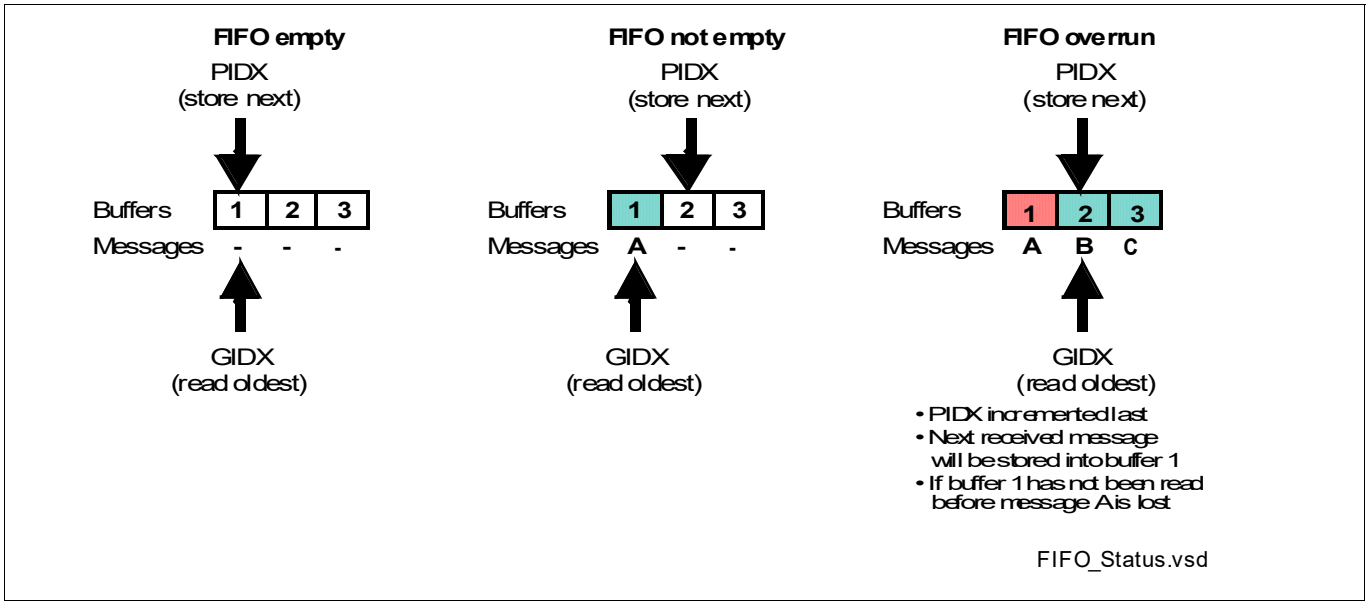


Figure 614 FIFO Status: Empty, Not Empty, Overrun

A FIFO non empty status is detected when the PUT index (PIDX) differs from the GET index (GIDX). In this case flag RFNE is set. This indicates that there is at least one received message in the FIFO. The FIFO empty, FIFO not empty, and the FIFO overrun states are explained in **Figure 614** for a three Message Buffer FIFO.

There is a programmable FIFO rejection filter for the FIFO. The FIFO Rejection Filter register (FRF) defines a filter pattern for messages to be rejected. The FIFO rejection filter consists of channel filter, Frame ID filter, and cycle counter filter. If bit RSS is set to 1 (default), all messages received in the static segment are rejected by the FIFO. If bit RNF is set to 1 (default), received NULL Frames are not stored in the FIFO.

The FIFO Rejection Filter Mask register (FRFM) specifies which bits of the Frame ID filter in the FIFO Rejection Filter register are marked “don’t care” for rejection filtering.

41.3.11.2 Configuration of the FIFO

For all Message Buffers belonging to the FIFO the data pointer to the first 32-bit word of the Data Section of the respective Message Buffer in the Message RAM has to be configured via the Write Header Section 3 register. All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask and needs not be configured in the Header Sections of the Message Buffers belonging to the FIFO.

When programming the data pointers for the Message Buffers belonging to the FIFO, the payload length of all Message Buffers should be programmed to the same value.

Note: It is recommended to program the MBI bits of the Message Buffers belonging to the FIFO to 0 via WRHS1.MBI to avoid generation of RX interrupts.
If the payload length of a received Frame is longer than the value programmed by WRHS2.PLC in the Header Section of the respective Message Buffer, the data field stored in a Message Buffer of the FIFO is truncated to that length.

41.3.11.3 Access to the FIFO

To read from the FIFO the Host has to trigger a transfer from the Message RAM to the Output Buffer by writing the number of the first Message Buffer of the FIFO (referenced by FFB) to the Output Buffer Command Request register. The Message Handler then transfers the Message Buffer addressed by the GET Index Register (GIDX) to the Output Buffer. After this transfer the GET Index Register (GIDX) is incremented.

41.3.12 Message Handling

The Message Handler controls data transfers between the Input / Output Buffer and the Message RAM and between the Message RAM and the two Transient Buffer RAMs. All accesses to the internal RAM's are 32 bit accesses.

Access to the Message Buffers stored in the Message RAM is done under control of the Message Handler state machine. This avoids conflicts between accesses of the two protocol controllers and the Host to the Message RAM.

Frame IDs of Message Buffers assigned to the static segment have to be in the range from 1 to NSS as configured in the GTU Configuration Register 7. Frame IDs of Message Buffers assigned to the dynamic segment have to be in the range from NSS + 1 to 2047.

Received messages with no matching dedicated receive buffer (static or dynamic segment) are stored in the receive FIFO (if configured) if they pass the FIFO rejection filter.

41.3.12.1 Host access to Message RAM

The message transfer between Input Buffer and Message RAM as well as between Message RAM and Output Buffer is triggered by the Host by writing the number of the target / source Message Buffer to be accessed to the Input or Output Buffer Command Request register.

The Input / Output Buffer Command Mask registers can be used to write / read Header and Data Section of the selected Message Buffer separately. If bit STXRS in the Input Buffer Command Mask register is set (STXRS = 1), the transmission request flag TXR of the selected Message Buffer is automatically set after the Message Buffer has been updated.

If bit STXRS in the Input Buffer Command Mask register is reset (STXRS = 0), the transmission request flag TXR of the selected Message Buffer is reset. This can be used to stop transmission from Message Buffers operated in continuous mode.

Input Buffer (IBF) and the Output Buffer (OBF) are build up as a double buffer structure. One half of this double buffer structure is accessible by the Host (IBF Host / OBF Host), while the other half (IBF Shadow / OBF Shadow) is accessed by the Message Handler for data transfers between IBF / OBF and Message RAM.

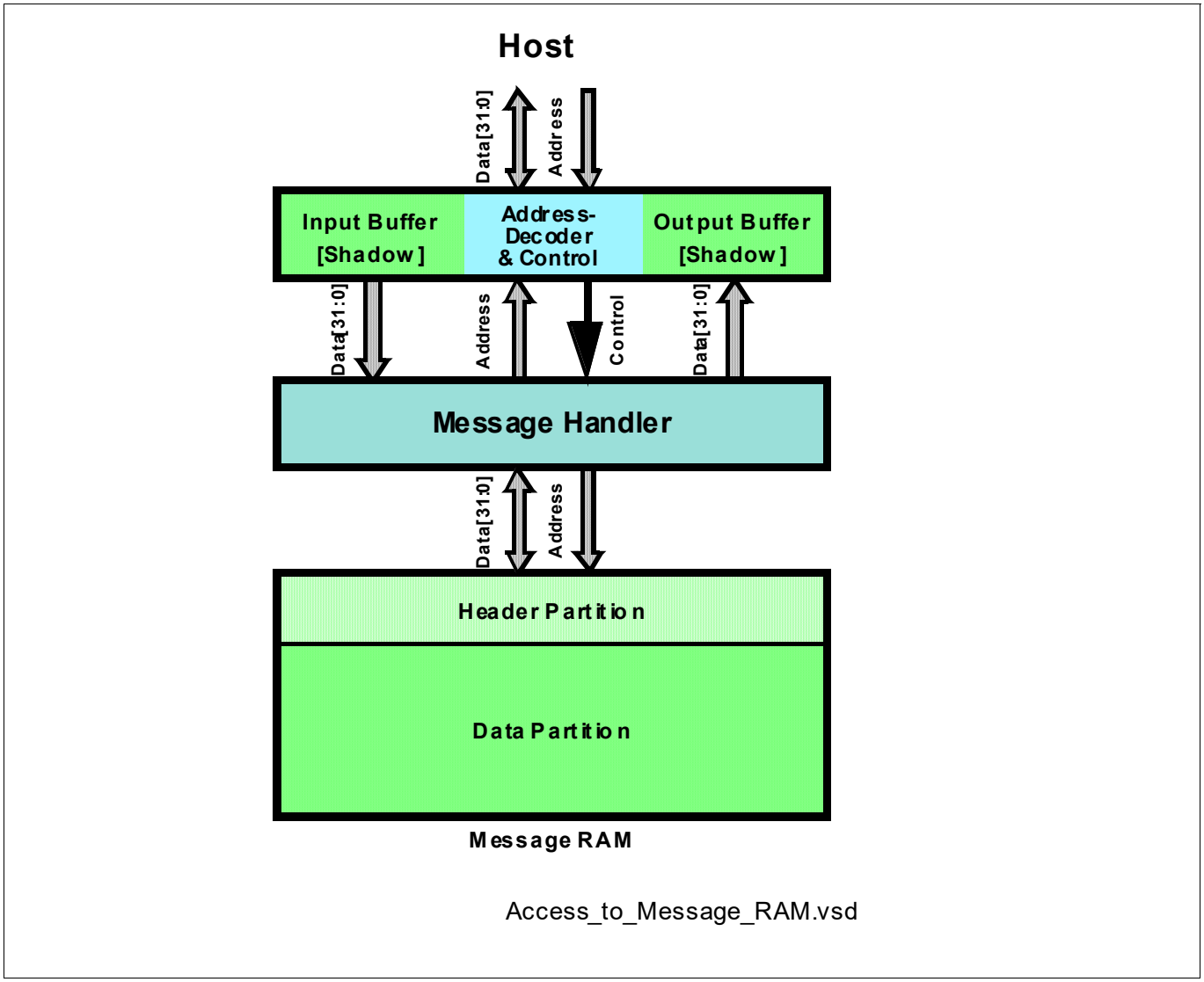


Figure 615 Host Access to Message RAM

Data Transfer from Input Buffer to Message RAM

To configure / update a Message Buffer in the Message RAM, the Host has to write the data to WRDSnn (nn = 01-64) and the Header to WRHS1, WRHS2, WRHS3. Two sets of WRDSnn (nn = 01-64) are available in parallel and selected by CUST1.IBF1PAG and CUST1.IBF2PAG. CUST1.IBFS shows which Input Buffer is currently used as Input Shadow Buffer and which as Input Host Buffer. WRHS1, WRHS2, and WRHS3 does only exist once. The specific action is selected by configuring the Input Buffer Command Mask IBCM.

When the Host writes the number of the target Message Buffer in the Message RAM to IBRH in the Input Buffer Command Request register IBCR, IBF Host and IBF Shadow are swapped (see **Figure 615**).

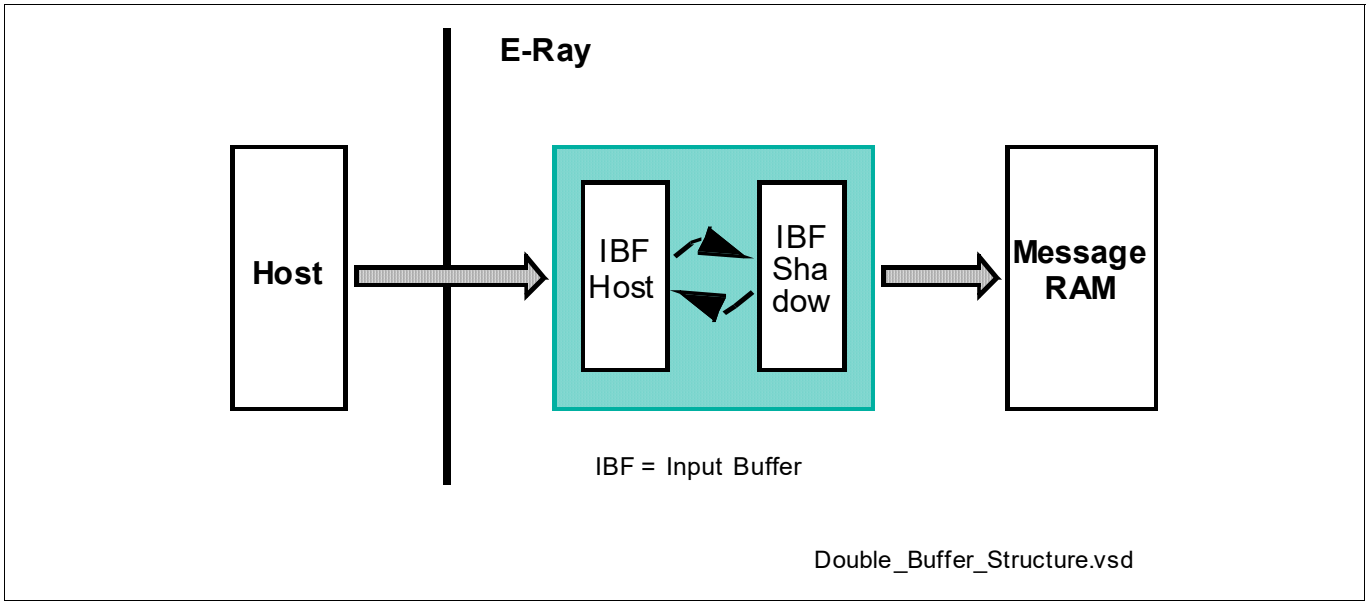


Figure 616 Swapping of IBCM and IBCR Bit

In addition the bits in the Input Buffer Command Mask and Input Buffer Command Request registers are also swapped to keep them attached to the respective IBF section (see Figure below).

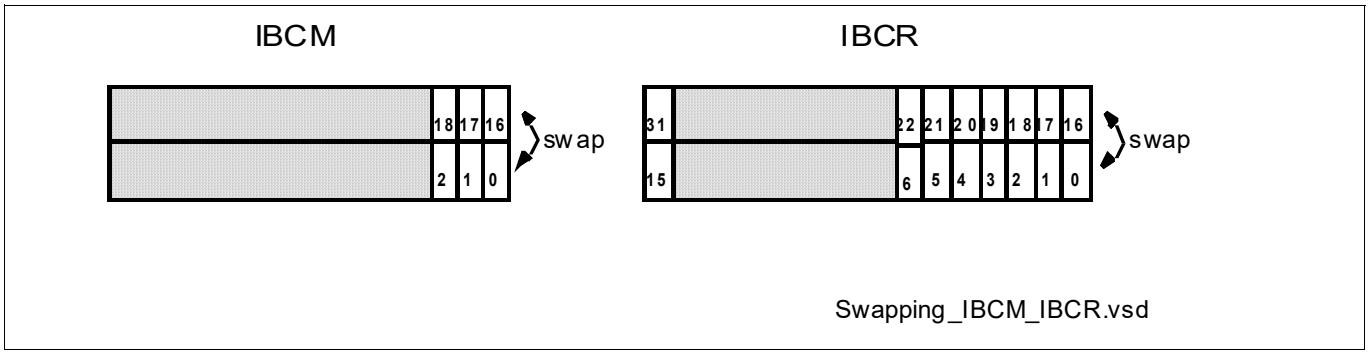


Figure 617 Swapping of IBCM and IBCR Bit

With this write operation the IBSYS bit in the Input Buffer Command Request register is set to 1. The Message Handler then starts to transfer the contents of IBF Shadow to the Message Buffer in the Message RAM selected by IBRS.

While the Message Handler transfers the data from IBF Shadow to the target Message Buffer in the Message RAM, the Host may write the next message to IBF Host. After the transfer between IBF Shadow and the Message RAM has completed, the IBSYS bit is set back to 0 and the next transfer to the Message RAM may be started by the Host by writing the respective target Message Buffer number to IBRH in the Input Buffer Command Request register.

If a write access to IBRH occurs while IBSYS is 1, IBSYH is set to 1. After completion of the ongoing data transfer from IBF Shadow to the Message RAM, IBF Host and IBF Shadow are swapped, IBSYH is reset to 0, IBSYS remains set to 1, and the next transfer to the Message RAM is started. In addition the Message Buffer numbers stored under IBRH and IBRS and the Command Mask flags are also swapped.

Table 395 Assignment of Input Buffer Command Mask Bit

Pos.	Access	Bit	Function
18	rh	STXRS	Set Transmission Request Shadow
17	rh	LDSS	Load Data Section Shadow
16	rh	LHSS	Load Header Section Shadow
2	rwh	STXRH	Set Transmission Request Host
1	rwh	LDSH	Load Data Section Host
0	rwh	LHSH	Load Header Section Host

Table 396 Assignment of Input Buffer Command Request Bit

Pos.	Access	Bit	Function
31	rh	IBSYS	**IBF Busy Shadow,** signals ongoing transfer from IBF Shadow to Message RAM
22…16	rh	IBRS	**IBF Request Shadow,** number of Message Buffer currently / last updated
15	rh	IBSYH	**IBF Busy Host,** transfer request pending for Message Buffer referenced by IBRH
6-0	rwh	IBRH	**IBF Request Host,** number of Message Buffer to be updated next

Data Transfer from Message RAM to Output Buffer

To read a Message Buffer from the Message RAM, the Host has to write to Command Request register OBCR to trigger the data transfer as configured in Output Buffer Command Mask OBCM register. After the transfer has completed, the Host can read the transferred data from RDDSnn (nn = 01-64), RDHS1, RDHS2, RDHS3, and MBS.

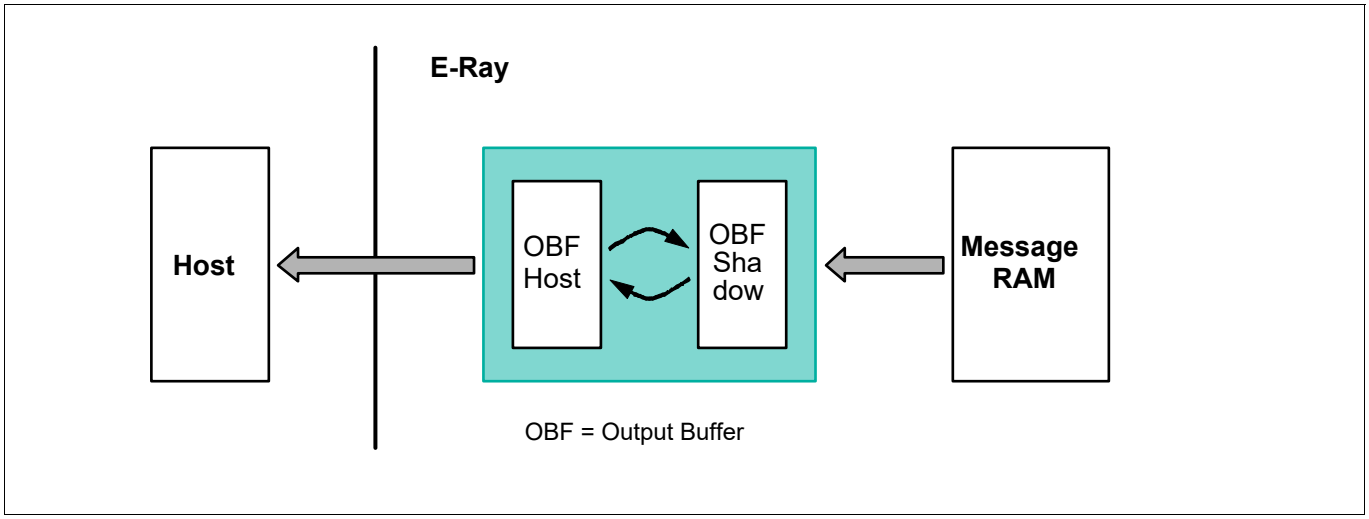


Figure 618 Double Buffer Structure Output Buffer

OBF Host and OBF Shadow as well as bits OBCM.RHSS, OBCM.RDSS, OBCM.RHSH, OBCM.RDSH and bits OBCR.OBRS, OBCR.OBRH are swapped under control of bits OBCR.VIEW and OBCR.REQ.

Writing bit OBCR.REQ to 1 copies bits OBCM.RHSS, OBCM.RDSS and bits OBCR.OBRS to an internal storage (see **Figure 619**).

After setting OBCR.REQ to 1, OBCR.OBSYS is set to 1, and the transfer of the Message Buffer selected by OBCR.OBRS from the Message RAM to OBF Shadow is started. After the transfer between the Message RAM and OBF Shadow has completed, the OBCR.OBSYS bit is set back to 0. Bits OBCR.REQ and OBCR.VIEW can only be set to 1 while OBCR.OBSYS is 0.

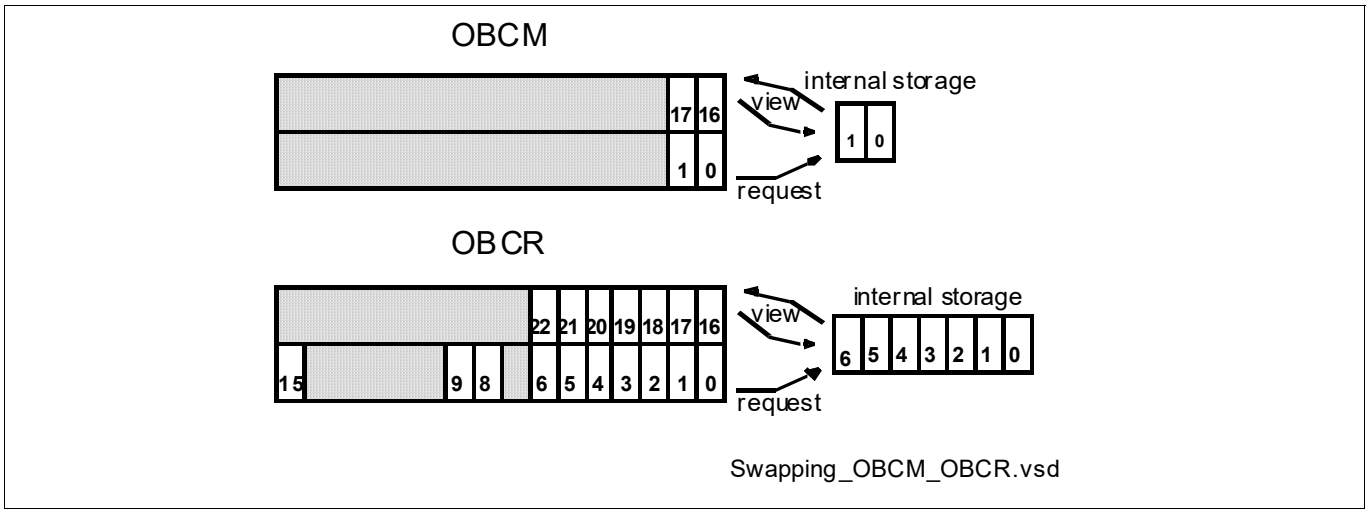


Figure 619 Swapping of OBCM and OBCR Bit

OBF Host and OBF Shadow are swapped by setting bit OBCR.VIEW to 1 while bit OBCR.OBSYS is 0 (see **Figure 619**).

In addition bits OBCR.OBRH and bits OBCM.RHSH, OBCM.RDSH are swapped with the registers internal storage thus assuring that the Message Buffer number stored in OBCR.OBRH and the mask configuration stored in OBCM.RHSH, OBCM.RDSH matches the transferred data stored in OBF Host (see **Figure 619**).

Now the Host can read the transferred Message Buffer from OBF Host while the Message Handler may transfer the next message from the Message RAM to OBF Shadow.

Table 397 Assignment of Output Buffer Command Mask Bit

Pos.	Access	Bit	Function
17	rh	RDSH	Data Section available for Host access
16	rh	RHSH	Header Section available for Host access
1	rwh	RDSS	Read Data Section Shadow
0	rwh	RHSS	Read Header Section Shadow

Table 398 Assignment of Output Buffer Command Request Bit

Pos.	Access	Bit	Function
22…16	rh	OBRH	**OBF Request Host** number of Message Buffer available for Host access
15	rh	OBSYS	**OBF Busy Shadow** signals ongoing transfer from Message RAM to OBF Shadow
9	rw	REQ	**Request Transfer from Message RAM to OBF Shadow**
8	rw	VIEW	**View OBF Shadow, swap OBF Shadow, and OBF Host**
6…0	rw	OBRS	**OBF Request Shadow** number of Message Buffer for next request

41.3.12.2 Data Transfers between IBF / OBF and Message RAM

This document uses the following terms and abbreviations:

Table 399 Terms and Abbreviations

Term	Meaning
MHD	Message Handler
IBF	Input Buffer 1 or 2 RAM
OBF	Output Buffer 1 or 2 RAM
MBF	Message Buffer RAM
TBF	Transient Buffer RAM Channel A (TBF1) or Channel B (TBF2)
IBF⇒ MBF	Transfer from IBF to MBF
MBF ⇒ OBF	Transfer from MBF to OBF
MBF⇒ TBF	Transfer from MBF to TBF
TBF⇒ MBF	Transfer from TBF to MBF
SS	Slot Status
SS⇒ MBF	Transfer SS to MBF

Message Handler functionality

The MHD controls the access to the MBF. It manages data-transfer between MBF and IBF, OBF, TBF1, TBF2. The data-path are shown in Figure below.

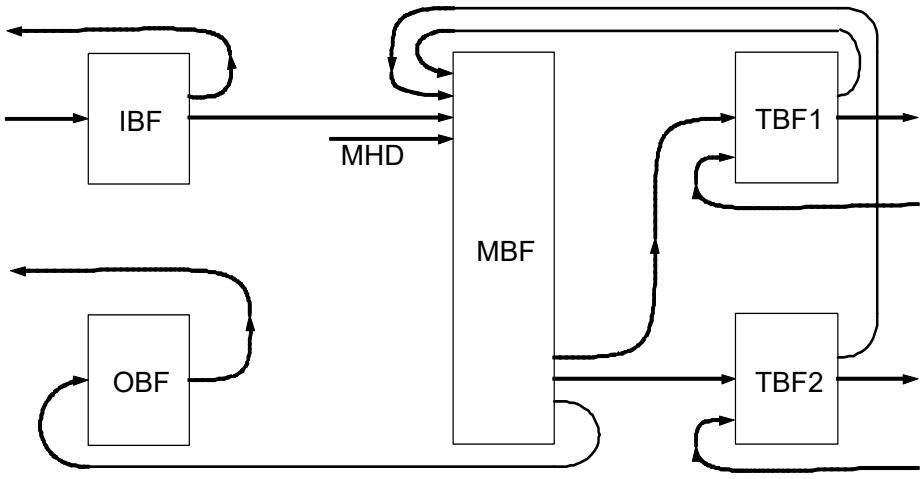


Figure 620 Interconnection of RAMs

Furthermore a search-algorithm allows to find the next valid message object in the MBF for transmission or reception.

Each transfer consists of a setup-time, four time steps to transfer the Header-section and a payload-length-dependent number of time steps to transfer the data-section. The internal data-busses have a width of 32 bits. Thereby it is possible to transfer two 2-byte words in one time step. If the payload consists of an odd number of

2-byte words the last time step of the data-section contains only 16 bit of valid data. If the Payload-Length (PL) is e.g. 7, the data-section consists of 4 time steps.

The maximum length for the data-section is 64 time steps, the minimum length is zero time steps.

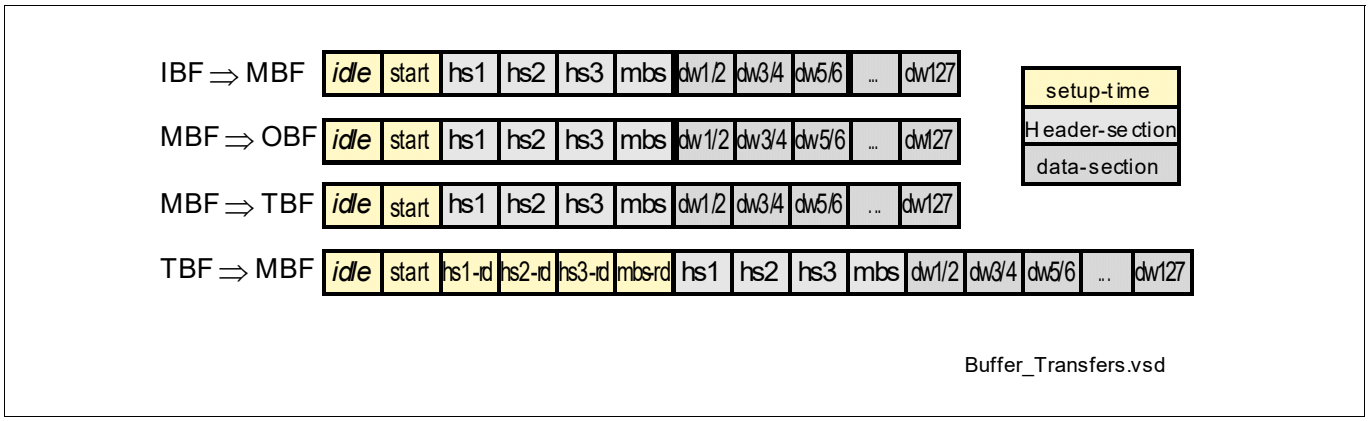


Figure 621 Different Possible Buffer Transfers

The update of the Slot-Status consists of a setup-time and one time-step to write the new Slot-Status.

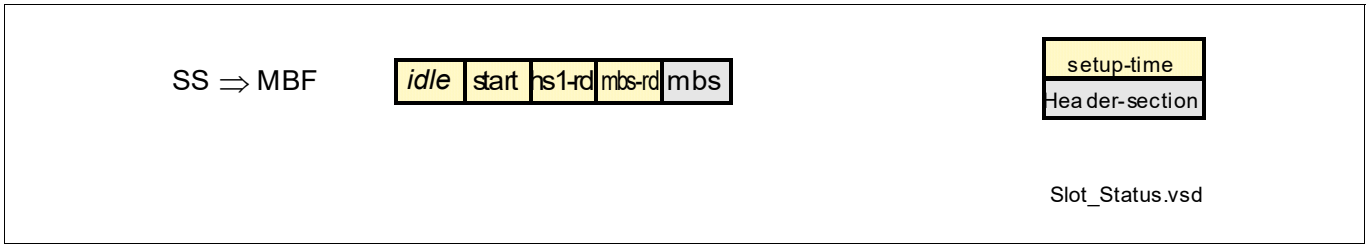


Figure 622 Update of Slot Status

The length of a time step depends on the number of concurrent tasks.

The following concurrent tasks are executed under control of the Message Handler:

- Data transfer between IBF or OBF and MBF
- Data transfer between TBF1 and MBF, search next TX / RX Message Buffer CHA
- Data transfer between TBF2 and MBF, search next TX / RX Message Buffer CHB

Thereby the time step length can vary between one and three f_{CLC_ERAY}periods.

Under certain conditions it is possible that a transfer is stopped or interrupted for a number of time steps until it is continued.

When a IB FÞ MBF is started short after a TBF Þ MBF or SS Þ MBF the transfer from IBF has to wait until the setup-time of the internal transfer has finished (see Figure below)

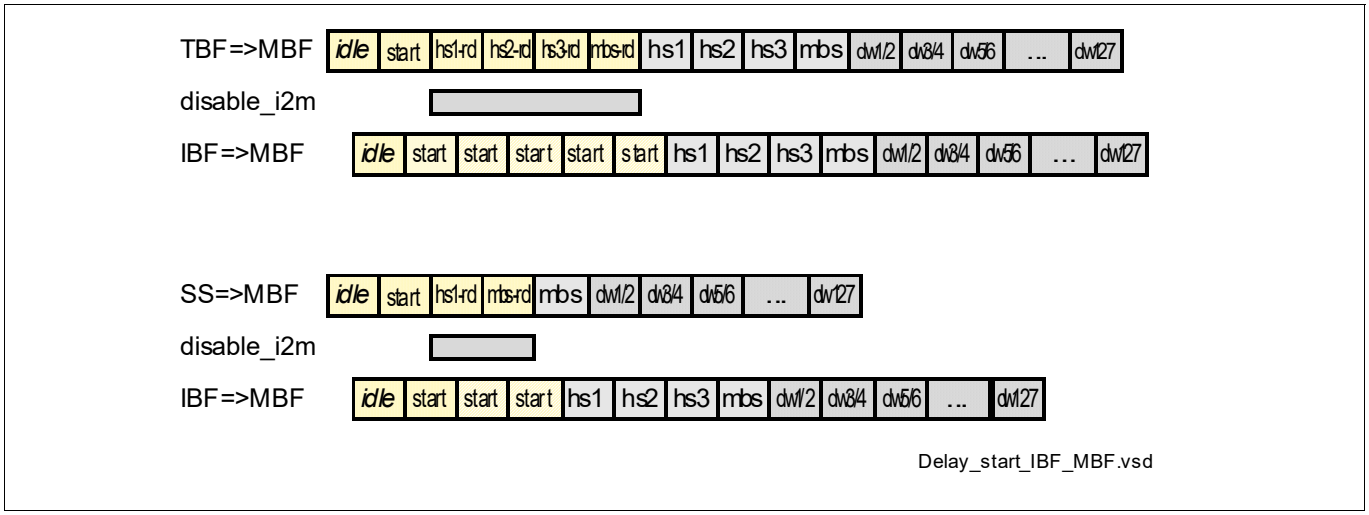


Figure 623 Delay start of IBF => MBF

The internal signal “disable_i2m” is always active when the TBF ⇒ MBF is in state “hs1-rd”, “hs2-rd”, “hs3-rd” or “mbs-rd” and when the SS⇒ MBF is in state “hs1-rd” or “mbs-rd”.

The IBF ⇒ MBF is hold in state “start” until the internal signal “disable_i2m” gets inactive.

These additional time-steps are independent of any address-counter-values. This means, the IBF ⇒ MBF has to wait even if it writes to another buffer than the internal transfer.

Multiple requests of transfers between IBF/OBF and Message RAM

The time required to transfer the contents of a Message Buffer between IBF / OBF and Message RAM depends on the number of 4-byte words to be transferred, the number of concurrent tasks to be managed by the Message Handler, and in special cases the type and address range of the internal transfer. The number of 4-byte words varies from 4 (Header Section only) to 68 (Header + maximum Data Section) plus a short setup time to start the first transfer, while the number of concurrent task varies from one to three. The 4 Header words have to be included in calculation even if only the Data Section is requested for transfer.

The following concurrent tasks are executed under control of the Message Handler:

- Data transfer between IBF or OBF and MBF
- Data transfer between TBF1 and MBF, search next TX / RX Message Buffer CHA
- Data transfer between TBF2 and MBF, search next TX / RX Message Buffer CHB

Transfers between IBF and MBF respectively MBF and OBF can only be handled one after another. In case that e.g. a IBF⇒ MBF has been started shortly before a MBF ⇒ OBF is requested, the MBF ⇒ OBF has to wait until the IBF ⇒ MBF has completed.

In case that e.g. a second IBF⇒ MBF is requested, a MBF⇒ OBF is requested and a IBF⇒ MBF is ongoing, the MBF⇒ OBF has to wait until the first IBF⇒ IBF has completed. The second IBF=MBF has to wait until the MBF⇒ OBF has completed (see figure below) independent whether MBF⇒ OBF or second IBF⇒ MBF is requested first.

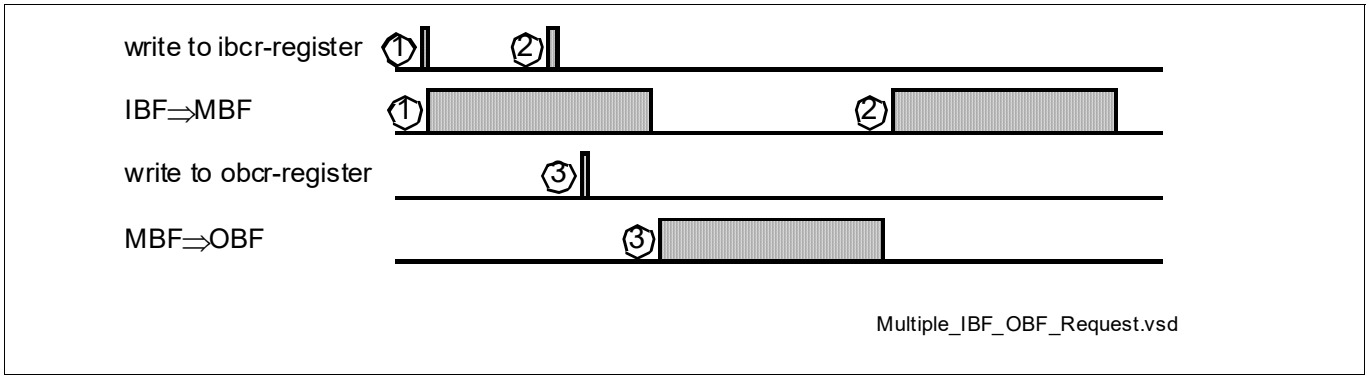


Figure 624 Multiple IBF/OBF Request

Worst case for single request

When a message with a large payload length is received the TBF⇒ MBF is started at the begin of the next slot (n+1). If the next slot is a dynamic slot without transmission/reception (minislot), it may happen that the TBF⇒ MBF has not finished until begin of the next but one slot (n+2). In this case the TBF⇒ MBF will be service requested (break) to start a transmission in the next but one slot (MBF⇒ TBF) and/or to update the slot status (SS⇒ MBF) for the RX-buffer corresponding with next slot (n+1). After this interruption the TBF⇒ MBF is continued.

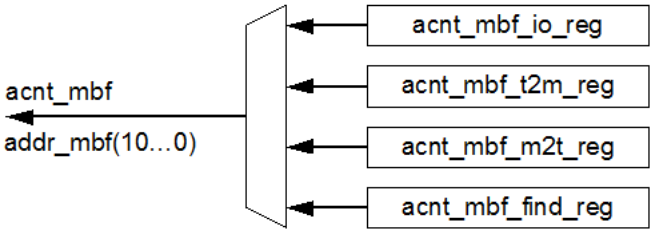


Figure 625 Address Counter Scheme of Message RAM (simplified)

For the transfers IBF⇒ MBF / MBF⇒ OBF, TBF⇒ MBF and MBF⇒ TBF separate address-counter are implemented (see **Figure 625**).

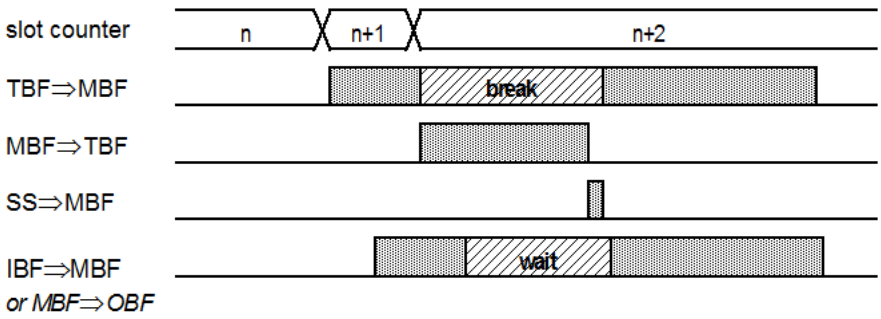


Figure 626 interruption of TBF to MBF

If the address-counter for IBF⇒ MBF / MBF⇒ OBF (acnt_mbf_io_reg) reaches the address of the interrupted TBF=>MBF (acnt_mbf_t2m_reg) the IBF⇒ MBF / MBF⇒ OBF has to wait until the TBF⇒ MBF is continued (see **Figure 626**).

The relative time is measured in f_{CLC_ERAY}cycles. Absolute time depends on the actual f_{CLC_ERAY}cycle period.

tbf_to_mbf_break time$_{max}$ =	(setup time + mbf_to_tbf time$_{max}$) + (setup time + ss_to_mbf)
cycles$_{req}$ =	(number of concurrent tasks) x ((setup time + (number of 4-byte words)$_{req}$) + tbf_to_mbf_break time)
setup time =	2 f_{CLC_ERAY}cycles

Worst case for one IBF⇒ MBF or MBF⇒ OBF:

Max. break time: tbf_to_mbf_break time$_{max}$ =	(2+68) + (4+1) = 75
Max. number of f_{CLC_ERAY}cycles: cycles$_{req}$ =	3 x (6 + 68 + 75) = 435

Worst case for multiple transfers

If a second IBF⇒ MBF and a MBF⇒ OBF (see **Figure 624**) is requested directly after the first IBF⇒ MBF has started following worst case timing could appear:

$cycles_{trans}$ =	(remaining cycles of transfer running) + (cycles of second requested transfer) + (cycles of third requested transfer)
$cycles_{trans}$ =	$cycles_{rem}$ + $cycles_{req\_2}$+ $cycles_{req\_3}$
Max. number of f_{CLC_ERAY}cycles: $cycles_{trans}$ =	447 + 435 + 447 = 1329

41.3.12.3 Minimum f_{CLC_ERAY}

To calculate the minimum $f_{CLC_ERAY}$the worst case scenario has to be considered.

The worst case scenario depends on the following parameters

- maximum payload length
- minimum minislot length
- number of configured Message Buffers (excluding FIFO)
- used channels (single/dual channel)

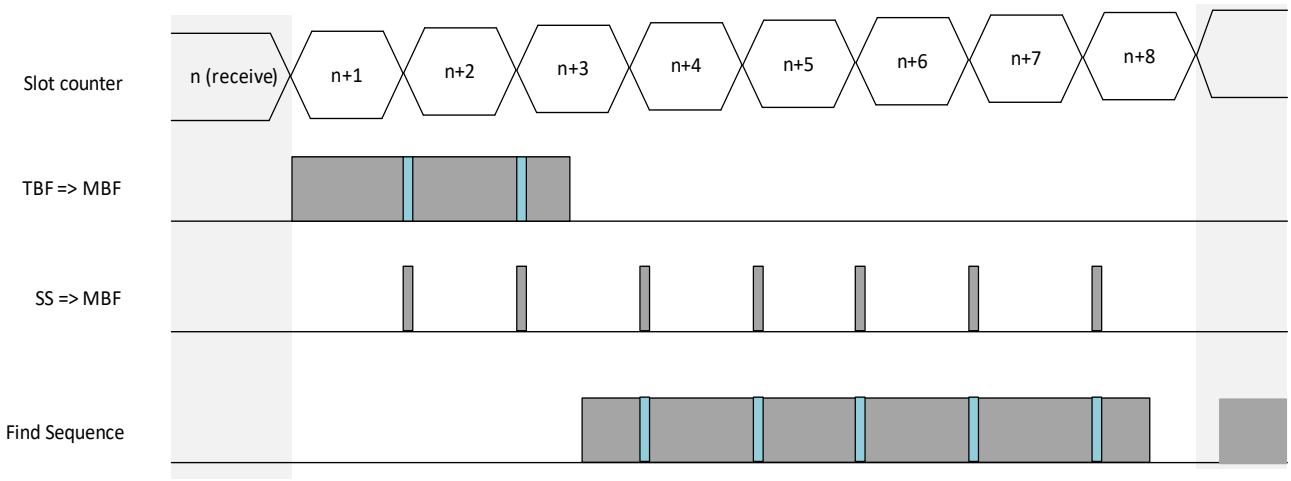


Figure 627 worst case scenario

Worst case scenario:

- reception of message with a maximum payload length in Slot n (n is 7,15,23,31,39,…)
- slot n+1 to n+7 are empty dynamic slots (minislot) and configured as receive buffer
- the find-sequence (usually started in slot 8,16,24,32,40,…) has to scan the maximum number of configured buffers
- the number of concurrent tasks has its maximum value of three

The find-sequence is executed each 8 Slots (slot 8,16,24,32,40,…). It has to be finished until the next find-sequence is requested.

The length of a TBF⇒ MBF varies from 4 (Header Section only) to 68 (Header + maximum Data Section) time step plus a setup time of 6 time steps.

$f_{CLC\_ERAY}cycles_{t2m}$ = number of concurrent tasks) x (setup $time_{t2m}$ + (number of 4-byte words)$_{t2m}$)

A SS⇒ MBF has a fixed length of 1 time steps plus a setup time of 4 time steps.

f_{CLC_ERAY} $cycles_{ss2m}$ = (number of concurrent tasks) x 5

The find sequence has a maximum length of 128 (maximum number of buffers) time steps plus a setup time of 2 time steps.

f_{CLC_ERAY} $cycles_{find}$ = (number of concurrent tasks) x (setup $time_{find}$ + (number of configured buffers))

A minislot has a length of 2 to 63 Macrotick (gdMinislot). The minimum nominal Macrotick period (cdMinMTNom) is 1 µs. A sequence of 8 minislots has a length of

$time_{8minislots}$ = 8 x gdMinislot x cdMinMTNom

The maximum period $T_{CLC\_ERAY} = 1/f_{CLC\_ERAY}$ can be calculated as followed:

$time_{8minislots} \geq$ (f_{CLC_ERAY} period in µs) x ((f_{CLC_ERAY} $cycles_{t2m}$) + 7 x (f_{CLC_ERAY} $cycles_{ss2m}$) + ($f_{CLC\_ERAY}cycles_{find}$))

f_{CLC_ERAY}period in ms ≤

$$\frac{time_{8minislots}}{(cycles_{t2m}) + 7 \times (cycles_{ss2m}) + (cycles_{find})}$$

minimum $time_{8minislots}$ = 8 x 2 x 1 µs = 16 µs

maximum f_{CLC_ERAY} $cycles_{t2m}$ = 3 x (6 + 68) = 222

maximum f_{CLC_ERAY} $cycles_{ss2m}$ = 3 * 5 = 15

maximum f_{CLC_ERAY} $cycles_{find}$ = 3 * (2 + 128) = 390

f_{CLC_ERAY} period in ms ≤

$$\frac{16\mu s}{222+7\times15+390} \approx 22.315 ns$$

The minimum f_{CLC_ERAY} frequency for this worst case scenario is 44.8125 MHz.

A too low f_{CLC_ERAY} frequency can cause a malfunction of the E-Ray.

The E-Ray can detect several malfunctions and reports this by setting the corresponding flag in the Message Handler Constraints Flags (MHDF) register.

41.3.12.3.1 Minimum f_{CLC_ERAY} for various maximum payload length

Table below summarizes the minimum required f_{CLC_ERAY} frequency for various maximum payload length assuming:

- a minimum minislot length of 2 µs.
- a maximum of 128 configured Message Buffers.

- dual channels in use.

Table 400 Minimum f_{CLC_ERAY} for different maximum payload length

Maximum payload length of 32 bit words	4	8	16	32	64
minimum f_{CLC_ERAY}	32,82 MHz	33,57 MHz	35,07 MHz	38,07 MHz	44,1 MHz

41.3.12.3.2 Minimum f_{CLC_ERAY} for various minimum minislot length

Table below summarizes the minimum required f_{CLC_ERAY} frequency for various minimum minislot length assuming:

- a maximum payload length of 254 bytes / 64 four-byte-words.
- a maximum 128 configured Message Buffers.
- dual channels in use.

Table 401 Minimum f_{CLC_ERAY} for different minimum minislot length

gdMinislot at dMinMTNom = 1 µs	2 µs	3 µs	4 µs	7 µs	8 µs
minimum f_{CLC_ERAY}	44,82 MHz	29,88 MHz	22,412 MHz	12,8 MHz	9,96 MHz

41.3.12.3.3 Minimum f_{CLC_ERAY} for various amount of configured Message Buffers

Table below summarizes the minimum required f_{CLC_ERAY} frequency for various amount of configured Message Buffers assuming:

- a maximum payload length of 254 bytes / 64 four-byte-words.
- a minimum minislot length of 2 µs.
- dual channels in use.

Table 402 Minimum f_{CLC_ERAY} for different amount of configured Message Buffers

Configured maximum amount of Message Buffers	128	64	32
minimum f_{CLC_ERAY}	44,82 MHz	32,82 MHz	26,82 MHz

41.3.12.3.4 Minimum f_{CLC_ERAY} for a typical configuration

´The minimum required f_{CLC_ERAY} frequency for various assuming the following typical E-Ray configuration:

- a maximum payload length of 32 bytes / 8 four-byte-words.
- a minimum minislot length of 7 µs.
- a maximum 128 configured Message Buffers.
- dual channels in use

The minimum f_{CLC_ERAY} frequency for this typical example would be 10 MHz.

41.3.12.4 FlexRay™ Protocol Controller access to Message RAM

The two Transient Buffer RAMs (TBF 1, TBF 2) are used to buffer the data for transfer between the two FlexRay™ Protocol Controllers and the Message RAM.

Each Transient Buffer RAM is build up as a double buffer, able to store two complete FlexRay™ messages. There is always one buffer assigned to the corresponding Protocol Controller while the other one is accessible by the Message Handler.

If e.g. the Message Handler writes the next message to be send to Transient Buffer Tx, the FlexRay™ Channel Protocol Controller can access Transient Buffer Rx to store the message it is actually receiving. During transmission of the message stored in Transient Buffer Tx, the Message Handler transfers the last received message stored in Transient Buffer Rx to the Message RAM (if it passes acceptance filtering) and updates the respective Message Buffer.

Data transfers between the Transient Buffer RAMs and the shift registers of the FlexRay™ Channel Protocol Controllers are done in words of 32 bit. This enables the use of a 32 bit shift register independent of the length of the FlexRay™ messages.

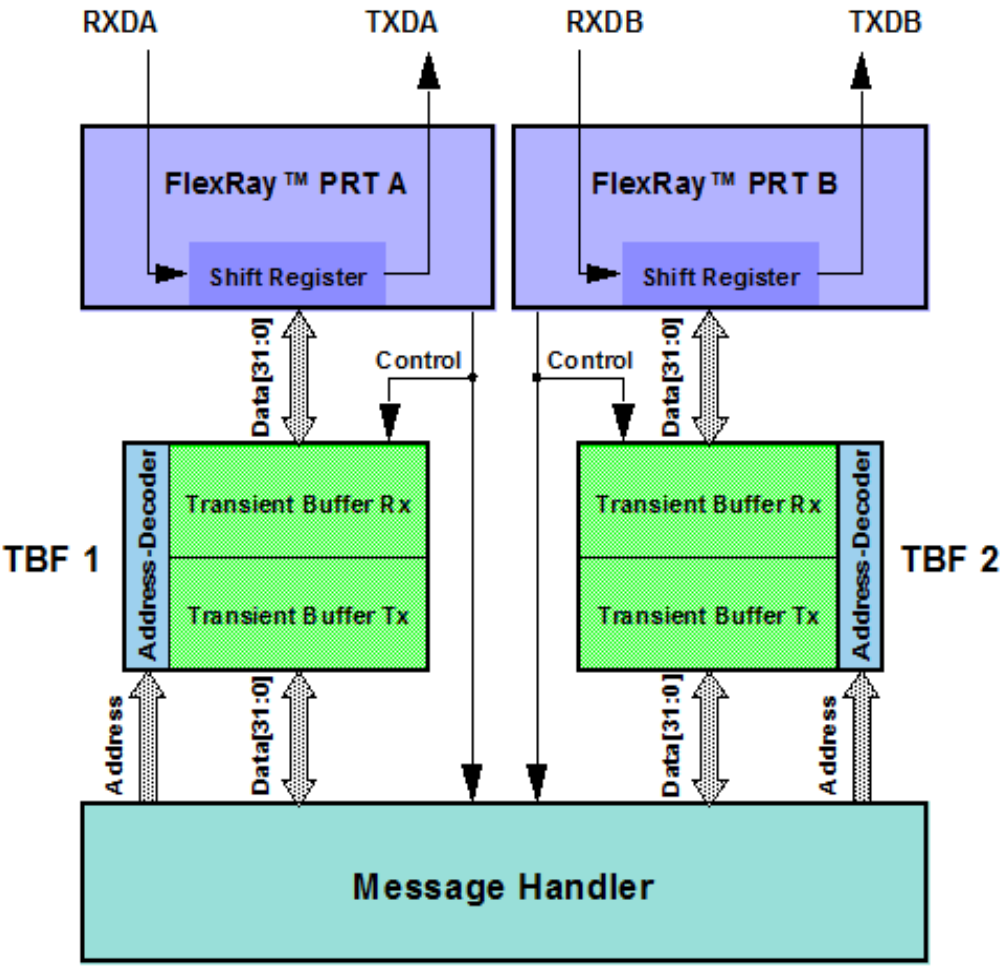


Figure 628 Access to Transient Buffer RAMs

41.3.13 Message RAM

To avoid conflicts between Host access to the Message RAM and FlexRay™ message reception / transmission, the Host cannot directly access the Message Buffers in the Message RAM. These accesses are handled via the Input and Output Buffers. The Message RAM is able to store up to 128 Message Buffers depending on the configured payload length.

The Message RAM is organized in 2048 32-bit words. To achieve the required flexibility with respect to different numbers of data byte per FlexRay™ Frame (0 to 254), the Message RAM has a structure as shown in **Figure 629**.

The Data Partition is allowed to start at Message RAM word number: (MRC.LCB + 1) • 4

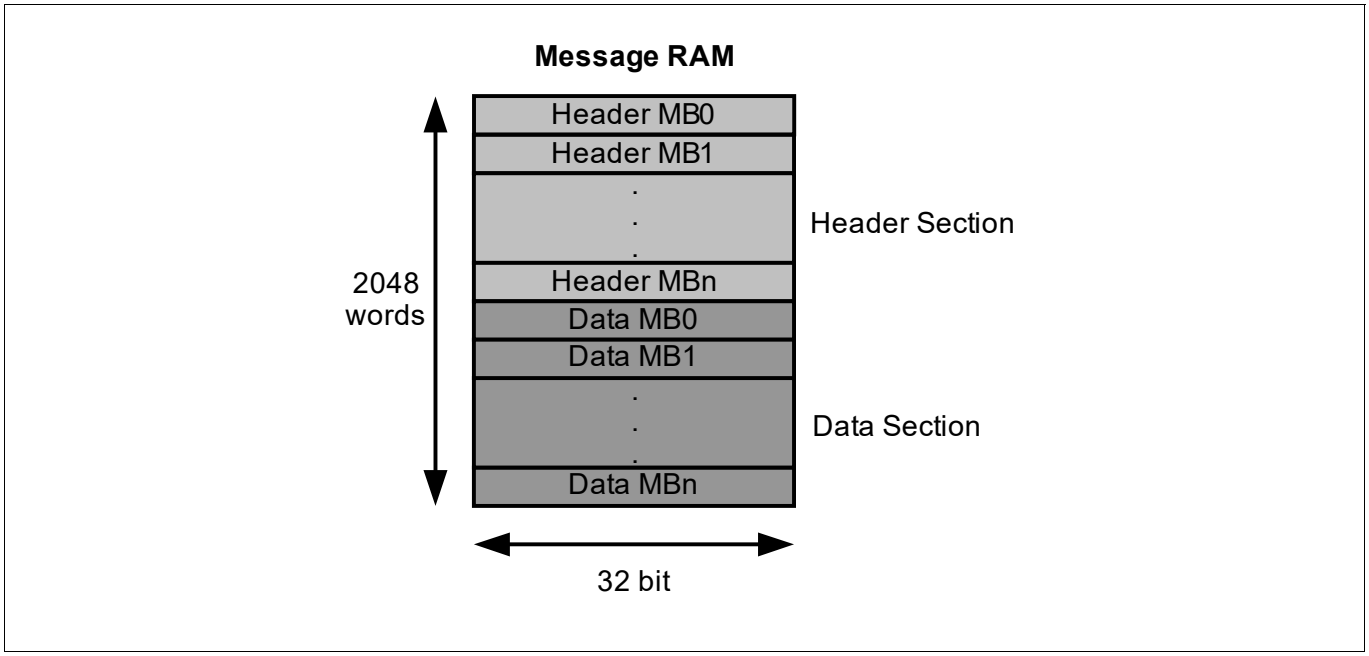


Figure 629 Structure of Message RAM

Header Partition

Stores Header Segments of FlexRay™ Frames:

- Supports a maximum of 128 Message Buffers
- Each Message Buffer has a Header of four 32 bit words
- Header 3 of each Message Buffer holds the11 bit pointer to the respective Data Section in the Data Partition

Data Partition

Flexible storage of Data Sections with different length. Some maximum values are:

- 30 Message Buffers with 254 byte Data Section each
- Or 56 Message Buffers with 128 byte Data Section each
- Or 128 Message Buffers with 48 byte Data Section each

Restriction: Header Partition + Data Partition may not occupy more than 2048 32-bit words.

41.3.13.1 Header Partition

The Header of each Message Buffer occupies four 32-bit words in the Header Partition of the Message RAM. The Header of Message Buffer 0 starts with the first word in the Message RAM.

For transmit buffers the Header CRC has to be calculated by the Host.

Payload Length Received PLR, Receive Cycle Count RCC, Received on Channel Indication RCI, Startup Frame Indication bit SFI, Sync bit SYN, NULL Frame Indication bit NFI, Payload Preamble Indication bit PPI, and Reserved bit RES are only updated from received valid Frames (including valid NULL Frames).

Header word 4 of each configured Message Buffer holds the respective Message Buffer Status MBS information.

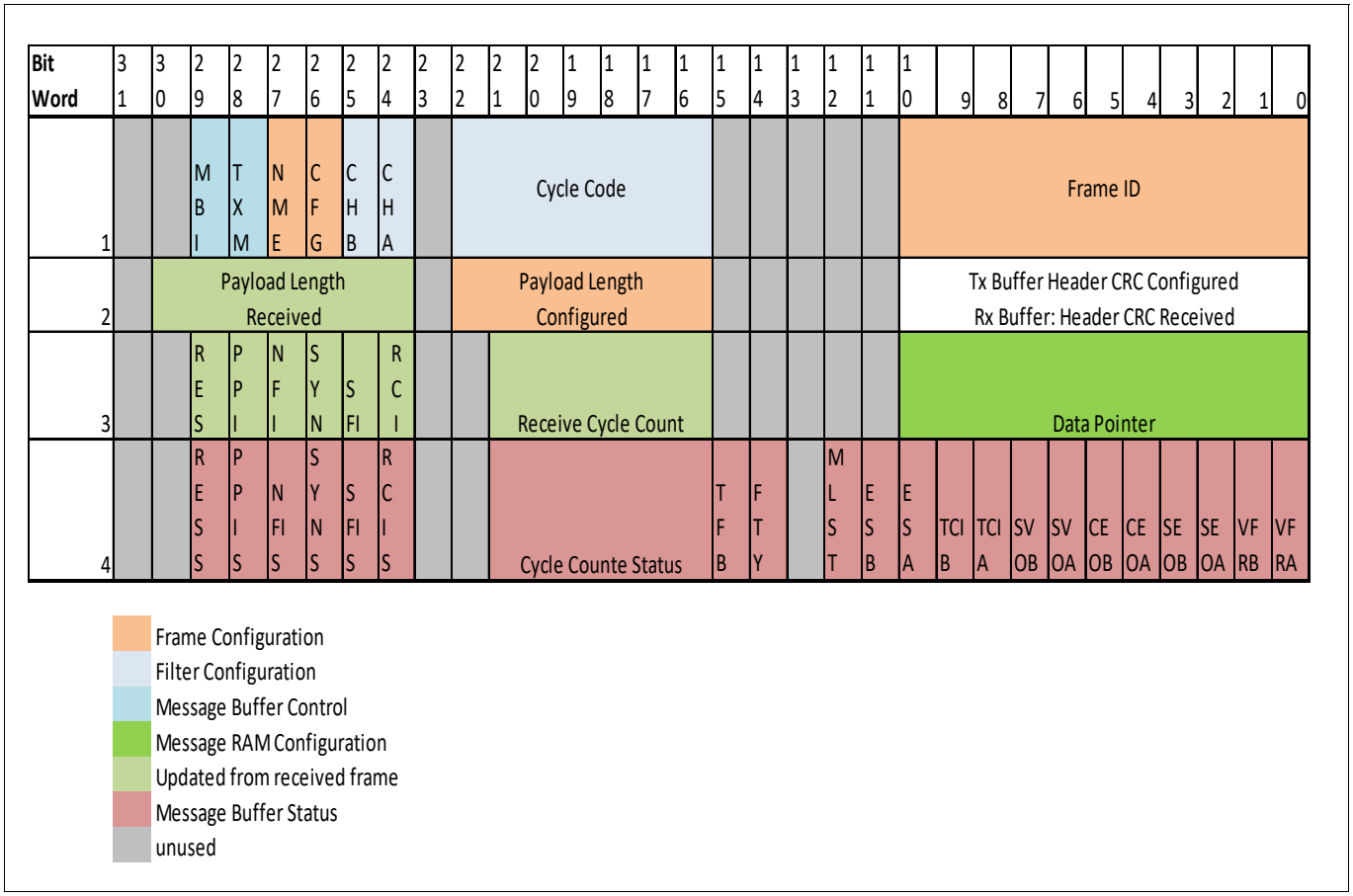


Figure 630 Header Section of a Message Buffer in the Message RAM

Header 1 (word 0)

Write access via WRHS1, read access via RDHS1:

- Frame ID: Slot counter filtering configuration
- Cycle Code: Cycle counter filtering configuration
- CHA, CHB: Channel filtering configuration
- CFG: Message Buffer configuration: receive / transmit
- PPIT: Payload Preamble Indicator Transmit
- XMI: Transmit mode configuration: single-shot / continuous
- MBI: Message Buffer receive / transmit service request enable

Header 2 (word 1)

Write access via WRHS2, read access via RDHS2:

- Header CRC
  - Transmit Buffer: Configured by the Host (calculated from Frame Header Segment)
  - Receive Buffer: Updated from received Frame
- Payload Length Configured
  - Length of Data Section (2-byte words) as configured by the Host
- Payload Length Received
  - Length of Payload Segment (2-byte words) stored from received Frame

Header 3

Write access via WRHS3, read access via RDHS3:

- Data Pointer
  - Pointer to the beginning of the corresponding Data Section in the Data Partition

Read access via RDHS3, valid for receive buffers only, updated from received Frames:

- Receive Cycle Count: Cycle count from received Frame
- RCI: Received on Channel Indicator
- SFI: Startup Frame Indicator
- SYN: SYNC Frame Indicator
- NFI: NULL Frame Indicator
- PPI: Payload Preamble Indicator
- RES: Reserved bit

Message Buffer Status MBS (word 3)

Read access via MBS, updated by the Communication Controller at the end of the configured slot.

- VFRA: Valid Frame Received on channel A
- VFRB: Valid Frame Received on channel B
- SEOA: Syntax Error Observed on channel A
- SEOB: Syntax Error Observed on channel B
- CEOA: Content Error Observed on channel A
- CEOB: Content Error Observed on channel B
- SVOA: Slot boundary Violation Observed on channel A
- SVOB: Slot boundary Violation Observed on channel B
- TCIA: Transmission Conflict Indication channel A
- TCIB: Transmission Conflict Indication channel B
- ESA: Empty Slot Channel A
- ESB: Empty Slot Channel B
- MLST: Message LoST
- FTA: Frame Transmitted on Channel A
- FTA: Frame Transmitted on Channel B
- Cycle Count Status: Actual cycle count when status was updated
- RCIS: Received on CHannel Indicator Status
- SFIS: Startup Frame Indicator Status
- SYNS: SYNC Frame Indicator Status
- NFIS: NULL Frame Indicator Status
- PPIS: Payload Preamble Indicator Status
- RESS: Reserved Bit Status

41.3.13.2 Data Partition

The Data Partition of the Message RAM stores the Data Sections of the Message Buffers configured for reception / transmission as defined in the Header Partition. The number of data bytes for each Message Buffer can vary from 0 to 254. To optimize the data transfer between the shift registers of the two FlexRay™ Protocol Controllers and the Message RAM as well as between the Host interface and the Message RAM, the physical width of the Message RAM is set to 4 bytes.

The Data Partition starts after the last word of the Header Partition. When configuring the Message Buffers in the Message RAM the programmer has to assure that the data pointers point to addresses within the Data Partition. **Table 403** below shows an example how the Data Sections of the configured Message Buffers can be stored in the Data Partition of the Message RAM.

The beginning and the end of a Message Buffer's Data Section is determined by the data pointer and the payload length configured in the Message Buffer's Header Section, respectively. This enables a flexible usage of the available RAM space for storage of Message Buffers with different data length.

If the size of the Data Section is an odd number of 2-byte words, the remaining 16 bits in the last 32-bit word are unused (see Table below)

Table 403 Example for Structure of the Data Section in the Message RAM

Bit Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
…	unused								unused								unused								unused							
…	unused								unused								unused								unused							
…	**MB1** Data3								**MB1** Data2								**MB1** Data1								**MB1** Data0							
…	…								…								…								…							
…	…								…								…								…							
…	**MB1** Data(m)								**MB1** Data(m-1)								**MB1** Data(m-2)								**MB1** Data(m-3)							
…	…								…								…								…							
…	…								…								…								…							
…	…								…								…								…							
…	**MBn** Data3								**MBn** Data2								**MBn** Data1								**MBn** Data0							
…	…								…								…								…							
…	**MBn** Data(k)								**MBn** Data(k-1)								**MBn** Data(k-2)								**MBn** Data(k-3)							
2046	**MB80** Data3								**MB80** Data2								**MB80** Data1								**MB80** Data0							
2047	unused								unused								**MB80** Data5								**MB80** Data4							

41.3.13.3 ECC Check

There is an ECC checking mechanism implemented in the E-Ray module to assure the integrity of the data stored in the seven RAM blocks of the module. The RAM blocks have an ECC generator / checker attached as shown in **Figure 631**. When data is written to a RAM block, the local ECC generator generates the ECC data. The ECC data is stored together with the respective data word. The ECC data is checked each time a data word is read from any of the RAM blocks.

If an ECC error is detected, the respective error flag is set. The ECC error flags MHDS.EIBF, MHDS.EOBF, MHDS.EMR, MHDS.ETBF1, MHDS.ETBF2, and the faulty Message Buffer indicators MHDS.FMBD, MHDS.MFMB, MHDS.FMB are located in the Message Handler Status register. These error flags control the error interrupt flag EIR.EERR.

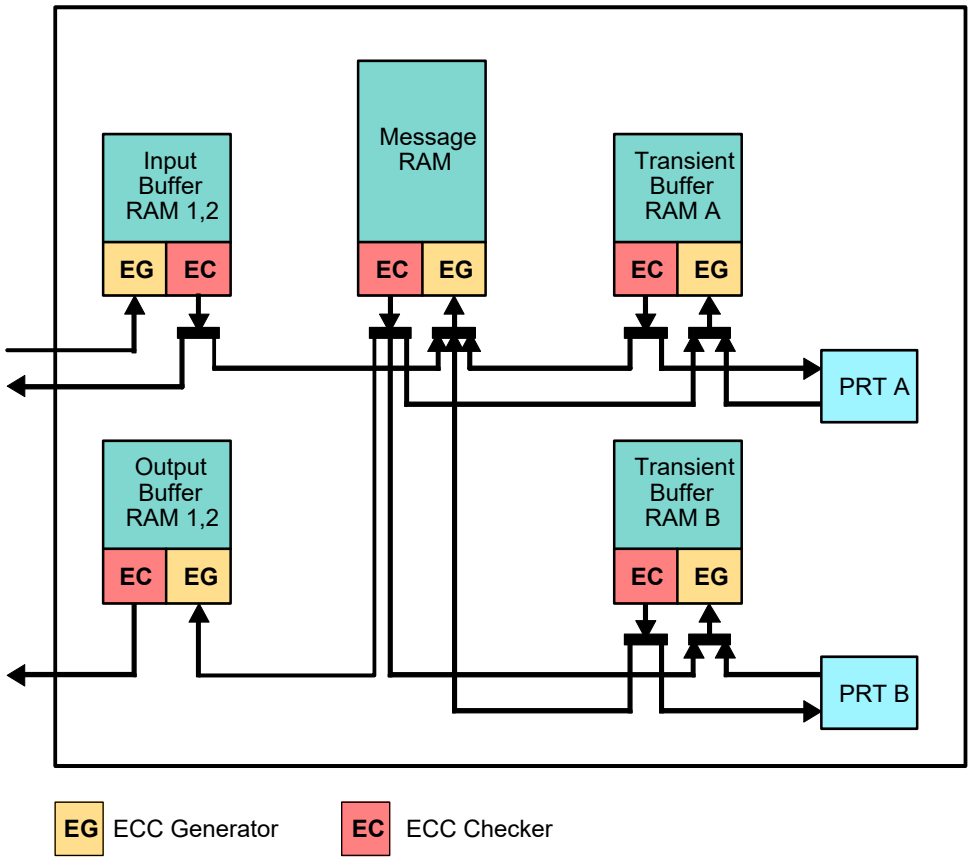


Figure 631 ECC Generation and Check

When an ECC error has been detected the following actions will be performed:

In all cases

- The respective ECC error flag in the Message Handler Status MHDS register is set
- The ECC error flag EIR.EERR in the Error Service Request Register is set, and if enabled, a module service request to the Host will be generated.

Additionally in specific cases

1. ECC error in data transfer from Input Buffer RAM 1,2 ⇒ Message RAM
   (Transfer of Header and Data Section)
   a) MHDS.EIBF bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB has been updated
   c) MHDS.FMB indicates the number of the faulty Message Buffer
   d) Transmit buffer: Transmission request for the respective Message Buffer is not set
2. ECC error in data transfer from Input Buffer RAM 1,2 ⇒ Message RAM
   (Transfer of Data Section only)
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer

c) MHDS.FMB indicates the number of the faulty Message Buffer
d) The Data Section of the respective Message Buffer is not updated
e) Transmit buffer: Transmission request for the respective Message Buffer is not set

3. ECC error during host reading Input Buffer RAM
   a) • MHDS.EIBF bit is set
4. ECC error during scan of Header Sections in Message RAM
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
   d) Ignore Message Buffer (Message Buffer is skipped)
5. ECC error during data transfer from Message RAM ⇒ Transient Buffer RAM A, B
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
   d) Frame not transmitted, Frames already in transmission are invalidated by setting the Frame CRC to zero
6. ECC error during data transfer from Transient Buffer RAM A, B ⇒ Protocol Controller 1, 2
   a) MHDS.ETBF1, MHDS.ETBF2 bit is set
7. ECC error in data transfer from Transient Buffer RAM A, B ⇒ Message RAM
   (ECC error when reading Header Section of respective Message Buffer from Message RAM)
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
   d) The Data Section of the respective Message Buffer is not updated
8. ECC error in data transfer from Transient Buffer RAM A, B ⇒ Message RAM
   (ECC error when reading Transient Buffer RAM A, B)
   a) MHDS.ETBF1, MHDS.ETBF2 bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
9. ECC error during data transfer from Message RAM ⇒ Output Buffer RAM
   a) MHDS.EMR bit is set
   b) MHDS.FMBD bit is set to indicate that MHDS.FMB points to a faulty Message Buffer
   c) MHDS.FMB indicates the number of the faulty Message Buffer
10. ECC error during Host reading Output Buffer RAM
    a) • MHDS.EOBF bit is set
11. ECC error during data read of Transient Buffer RAM A, B

If an ECC error occurs while the Message Handler reads a Frame with Network Management information (PPI = 1) from the Transient Buffer RAM A, B the corresponding Network Management vector registers NMV1 to NMV3 are not updated from that Frame.

41.3.14 Host Handling of Errors

An ECC error caused by transient bit flips can be fixed by:

41.3.14.1 Self-Healing

ECC errors located in

- Input Buffer RAM 1,2
- Output Buffer RAM 1,2
- Data Section of Message RAM
- Transient Buffer RAM A
- Transient Buffer RAM B

are overwritten with the next write access to the disturbed bit(s) caused by Host access or by FlexRay communication.

41.3.14.2 CLEAR_RAMS Command

After Power-On Reset, the E-Ray RAMs hold arbitrary values which causes ECC errors (MHDS) when a read operation is performed in a E-Ray RAM location. Hence the ERAY RAMs should be initialized always after a Power-On reset. When called in DEFAULT_CONFIG or CONFIG state POC command CLEAR_RAMS initializes all module-internal RAMs to zero. Inorder to perform a safe initialization of RAM blocks, the following programming sequence is suggested:

1. Remove EINIT protection for the writing of the CLC register
2. Enable the clock in the CLC register
3. Read the CLC register back to ensure that the clocks are enabled
4. Enable the EINIT protection
5. Enable the test mode. Take care of the unlock sequence. Refer to description of LCK.TMK and TEST1.WRTEN
6. Check if CCSV.POCS is either 0x0 (DEFAULT_CONFIG) or 0xF (CONFIG). If not in any of these states, perform the according command to get to CONFIG state
7. Check if SUCC1.PBSY is equal to 0x0. If 0x1 wait until 0x0.
8. Set SUCC1.CMD to 0xC meaning that the CLEAR_RAMS command is entered
9. Read SUCC1.CMD. If 0x0 the command has not been accepted. Repeat up from step 7. Otherwise continue.
10. Wait for 1024 module cycles.
11. Enable RAM Test mode: TEST1.TMC = 01b. This mode enables access of all RAM blocks in E-Ray module to host.
12. CUST1.IBF1PAG = 1b
13. CUST1.IBF2PAG = 1b
14. Repeat steps 7 to 10 and then proceed to step 15
15. Read atleast one address in all the E-Ray RAM blocks
16. Switch off Test Mode: TEST1.WRTEN = 0b
17. Clear ECC error flags in MHDS and EIR registers

41.3.14.3 Temporary Unlocking of Header Section

An ECC error in the header section of a locked message buffer can be fixed by a transfer from the Input Buffer to the locked buffer Header Section. For this transfer, the write-access to the IBCR (specifying the message buffer number) must be immediately preceded by the unlock sequence normally used to leave CONFIG state (see “Lock Register”).For that single transfer the respective message buffer header is unlocked, regardless whether it belongs to the FIFO or whether its locking is controlled by MRC.SEC[1:0], and will be updated with new data.

41.3.15 Module Service Request

In general, service requests provide a close link to the protocol timing as they are triggered almost immediately when an error or status change is detected by the controller, a Frame is received or transmitted, a configured timer service request is activated, or a stop watch event occurred. This enables the Host to react very quickly on specific error conditions, status changes, or timer events. On the other hand too many service requests can cause the Host to miss deadlines required for the application. Therefore the Communication Controller supports disable / enable controls for each individual service request source separately.

An service request may be triggered when

- An error was detected
- A status flag is set
- A timer reaches a preconfigured value
- A message transfer from Input Buffer to Message RAM or from Message RAM to Output Buffer has completed
- A stop watch event occurred

Tracking status and generating service requests when a status change or an error occurs are two independent tasks. Regardless of whether an service request is enabled or not, the corresponding status is tracked and indicated by the Communication Controller. The Host has access to the actual status and error information by reading the Error Service Request Register EIR and the Status Service Request SIR Register.

Table 404 Module Service Request Flags and Service Request Line Enable

Register	Bit	Function
SIR	WST	Wakeup Status
	CAS	Collision Avoidance Symbol
	CYCS	Cycle Start Service Request
	TXI	Transmit Service Request
	RXI	Receive Service Request
	RFNE	Receive FIFO not Empty
	RFCL	Receive FIFO Critical Level
	NMVC	Network Management Vector Changed
	TI0	Timer Service Request 0
	TI1	Timer Service Request 1
	TIBC	Transfer Input Buffer Completed
	TOBC	Transfer Output Buffer Completed
	SWE	Stop Watch Event
	SUCS	Startup Completed Successfully
	MBSI	Message Buffer Status Interrupt
	SDS	Start of Dynamic Segment
	WUPA	Wakeup Pattern Channel A
	MTSA	MTS Received on Channel A
	WUPB	Wakeup Pattern Channel B
	MTSB	MTS Received on Channel B
ILE	EINT0	Enable Service Request Line 0
	EINT1	Enable Service Request Line 1

Table 404 Module Service Request Flags and Service Request Line Enable (cont'd)

Register	Bit	Function
EIR	PEMC	Protocol Error Mode Changed
	CNA	Command Not Valid
	SFBM	SYNC Frames Below Minimum
	SFO	SYNC Frame Overflow
	CCF	Clock Correction Failure
	CCL	CHI Command Locked
	EERR	ECC Error
	RFO	Receive FIFO Overrun
EIR	EFA	Empty FIFO Access
	IIBA	Illegal Input Buffer Access
	IOBA	Illegal Output Buffer Access
	MHF	Message Handler Constraints Flag
	EDA	Error Detected on Channel A
	LTVA	Latest Transmit Violation Channel A
	TABA	Transmission Across Boundary Channel A
	EDB	Error Detected on Channel B
	LTVB	Latest Transmit Violation Channel B
	TABB	Transmission Across Boundary Channel B

The interrupt lines to the Host INT0SR and INT1SR are controlled by the enabled interrupts. In addition each of the two interrupt lines can be enabled / disabled separately by programming bit ILE.EINT0/INT0SRC.SRE and ILE.EINT1/ INT1SRC.SRE.

The interrupt lines to the Host NDAT0SR and NDAT1SR are controlled by the enabled new data interrupts (NDIC1 to NDIC4). In addition each of the two interrupt lines can be enabled / disabled separately by programming bit NDAT0SRC.SRE and NDAT1SRC.SRE.

The interrupt lines to the Host MBSC0SR and MBSC1SR are controlled by the enabled message buffer status changed interrupts (MSIC1 to MSIC4). In addition each of the two interrupt lines can be enabled / disabled separately by programming bit MBSC0SRC.SRE and MBSC1SRC.SRE.

The two timer service requests generated by service request timer 0 and 1 are available on pins TINT0SR and TINT1SR. They can be configured via the Timer 0 and Timer 1 Configuration register. In addition each of the two interrupt lines can be enabled / disabled separately by programming bit TINT0SRC.SRE and TINT1SRC.SRE.

A stop watch event may be triggered via input pin **STPWn**.

The status of the data transfer between IBF / OBF and the Message RAM is signalled on signals IBUSY and OBUSY. When a transfer has completed bit SIR.TIBC or SIR.TOBC is set.

41.3.16 Restrictions

The following restrictions have to be considered when programming the E-Ray IP-module. A violation of these restrictions may lead to an erroneous behavior of the E-Ray IP-module.

41.3.16.1 Message Buffers with the same Frame ID

If two or more Message Buffers are configured with the same Frame ID, and if they have a matching cycle counter filter value for the same slot, then the Message Buffer with the lowest Message Buffer number is used.

Sharing of a static time slot via cycle counter filtering between different nodes of a FlexRay™ network is **not** allowed.

41.3.16.2 Data Transfers between IBF / OBF and Message RAM

The time required to transfer the contents of a Message Buffer between IBF / OBF and Message RAM depends on the setup time to start the first transfer, the number of 4-byte words to be transferred, and the number of concurrent tasks to be managed by the Message Handler. The number of 4-byte words varies from 4 (Header Section only) to 68 (Header + maximum Data Section) while the number of concurrent task varies from one to three.

The following concurrent tasks are executed under control of the Message Handler:

- Data transfer between IBF or OBF and Message RAM
- Data transfer between TBF1 and Message RAM, search next TX / RX Message Buffer CHA
- Data transfer between TBF2 and Message RAM, search next TX / RX Message Buffer CHB

Transfers between IBF and Message RAM respectively Message RAM and OBF can only be handled one after another. In case that e.g. a transfer between IBF and Message RAM has been started shortly before a transfer between Message RAM and OBF is requested, the OBF transfer has to wait until the IBF transfer has completed.

The relative time is measured in f_{CLC_ERAY} cycles. Absolute time depends on the actual f_{CLC_ERAY} cycle period.

cyclestrans = (remaining cycles of transfer running) + (cycles of requested transfer)

cyclestrans = cyclesrem + cyclesreq

cyclesrem = (number of concurrent tasks) * (setup time + (number of 4-byte words)rem)

cyclesreq = (number of concurrent tasks) * (setup time + (number of 4-byte words)req)

setup time = 2 f_{CLC_ERAY} cycles

Under worst case conditions a transfer is requested directly after the previous transfer started:

Max. number of f_{CLC_ERAY} cycles: cyclestrans = (3 * (2 + 68)) + (3 * (2 + 68)) = 420

Worst case timing: timetrans(40MHz) = 420 * 25ns = 10.5 µs

41.3.17 E-Ray Module Implementation

This section describes the E-Ray interfaces as implemented in AURIX™ TC3xx Platform with the clock control, port and DMA connections, interrupt control, and address decoding.

Figure below shows a detailed view of the E-Ray interface.

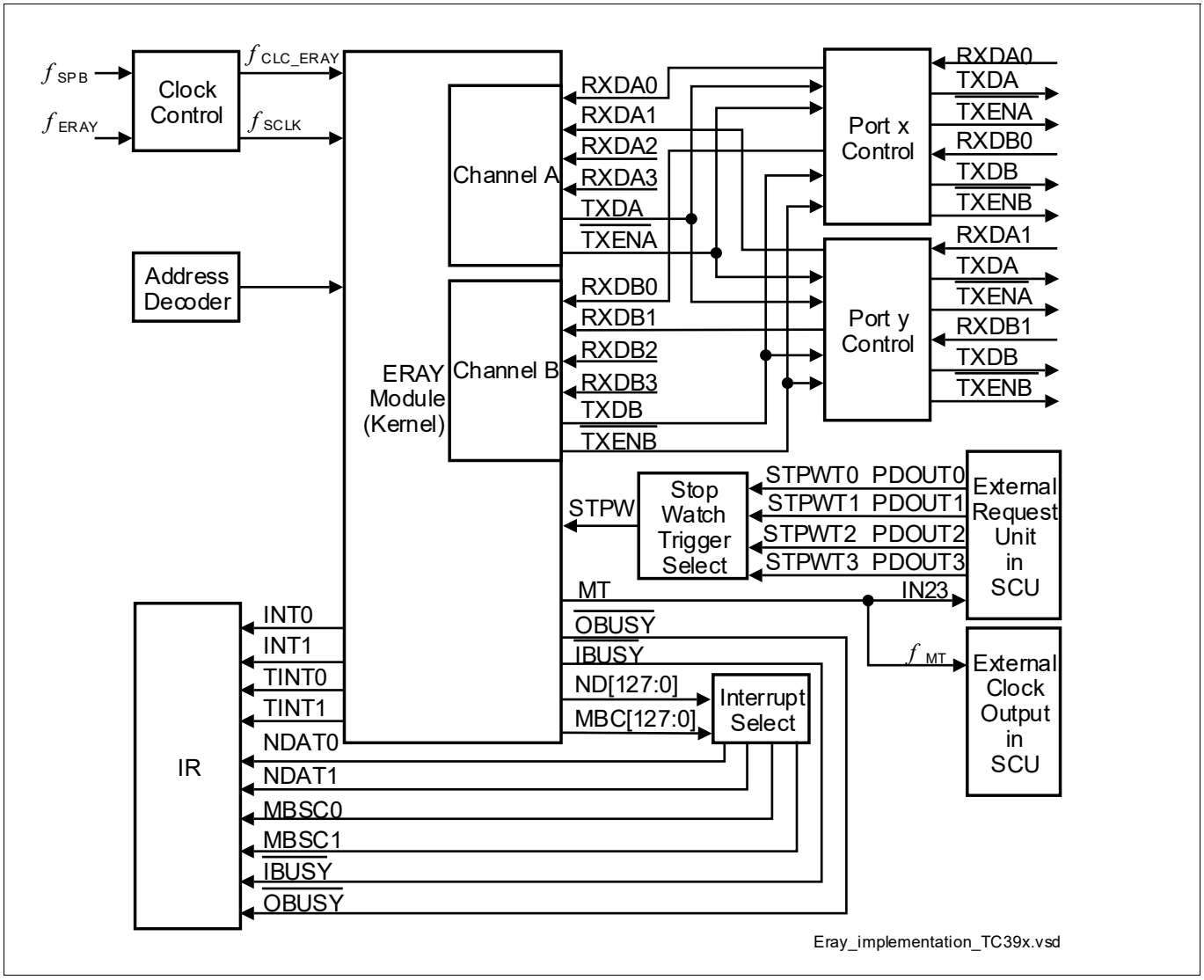


Figure 632 Detailed Block Diagram of the E-Ray Interface

41.3.17.1 Interconnections of the E-Ray Module

The E-Ray module has 2 FlexRay™ communication channels, channel A and channel B. Each channel provides a set of signals to drive a bus driver. The E-Ray module requires two different clocks, a sampling clock of the FlexRay™ bus f_{SCLK}. f_{SCLK} has to be 8 times the baud rate of the FlexRay™ communication. A second clock f_{CLC_ERAY} is used for the main protocol controller state machine and the customer interface logic. To enable deactivation of the E-Ray Module, f_{CLC_ERAY} and f_{SCLK} may be disabled (clock gated) by the CLC.DISR Enable E-Ray (Clock Gating) bit.The following items are described in this section:

- E-Ray module (kernel) external registers
- Port control and connections
  - I/O port line assignment
  - I/O function selection

  - Pad driver characteristics selection
- On-chip connections
  - SCU Connections
  - DMA connections
- Module clock generation
- Interrupt registers
- E-Ray address map

41.3.17.2 Port Control and Connections

This section describes the I/O connections of the E-Ray module.

41.3.17.2.1 Input/Output Function Selection

Table below shows how bits and bit fields must be programmed for the required I/O functionality of the E-Ray I/O lines. This table also shows the values of the peripheral input select registers.

Table 405 E-Ray I/O Control Selection and Setup

Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
FlexRay™ Channel 0A			
RXD0A0/ P14.8	ERAY_CUST1.RISA = 00_B	P14_IOCR8.PC8 = $0XXXX_B$	In
RXD0A1/ P11.9	ERAY_CUST1.RISA = 01_B	P11_IOCR8.PC9 = $0XXXX_B$	In
RXD0A2/ P02.1	ERAY_CUST1.RISA = 10_B	P02_IOCR0.PC1 = $0XXXX_B$	In
RXD0A3/ P14.1	ERAY_CUST1.RISA = 11_B	P14_IOCR0.PC1 = $0XXXX_B$	In
TXD0A/ P02.0	not applicable	P02_IOCR0.PC0 = $1X110_B$	Out
TXD0A/ P11.3	not applicable	P11_IOCR0.PC3 = $1X100_B$	Out
TXD0A/ P14.10	not applicable	P14_IOCR8.PC10 = $1X110_B$	Out
TXD0A/ P14.0	not applicable	P14_IOCR0.PC0 = $1X011_B$	Out
$\overline{\text{TXEN0A}}$/ P02.4	not applicable	P2_IOCR4.PC4 = $1X110_B$	Out
$\overline{\text{TXEN0A}}$/ P11.6	not applicable	P11_IOCR4.PC6 = $1X100_B$	Out
$\overline{\text{TXEN0A}}$/ P14.9	not applicable	P14_IOCR8.PC9 = $1X110_B$	Out
FlexRay™ Channel 0B			
RXD0B0/ P14.7	ERAY_CUST1.RISB = 00_B	P14_IOCR4.PC7 = $0XXXX_B$	In
RXD0B1/ P11.10	ERAY_CUST1.RISB = 01_B	P11_IOCR8.PC10 = $0XXXX_B$	In
RXD0B2/ P02.3	ERAY_CUST1.RISB = 10_B	P02_IOCR0.PC3 = $0XXXX_B$	In
RXD0B3/ P14.1	ERAY_CUST1.RISB = 11_B	P14_IOCR0.PC1 = $0XXXX_B$	In
TXD0B/ P02.2	not applicable	P02_IOCR0.PC2 = $1X110_B$	Out
TXD0B/ P14.0	not applicable	P14_IOCR0.PC0 = $1X100_B$	Out
TXD0B/ P14.5	not applicable	P14_IOCR4.PC5 = $1X110_B$	Out
TXD0B/ P11.12	not applicable	P11_IOCR12.PC12 = $1X100_B$	Out
$\overline{\text{TXEN0B}}$/ P02.5	not applicable	P2_IOCR4.PC5 = $1X110_B$	Out
$\overline{\text{TXEN0B}}$/ P14.6	not applicable	P14_IOCR4.PC6 = $1X110_B$	Out
$\overline{\text{TXEN0B}}$/ P14.9	not applicable	P14_IOCR8.PC9 = $1X101_B$	Out

Table 405 E-Ray I/O Control Selection and Setup (cont'd)

Port Lines	Input Select Register	Input/Output Control Register Bits	I/O
$\overline{\text{TXEN0B}}$/ P11.11	not applicable	P11_IOCR8.PC11 = $1X110_B$	Out
$\overline{\text{TXEN0B}}$/ P11.6	not applicable	P11_IOCR4.PC6 = $1X010_B$	Out
FlexRay™ Channel 1A			
RXD1A0/ P14.8	ERAY_CUST1.RISA = 00_B	P14_IOCR8.PC8 = $0XXXX_B$	In
RXD1A1/ P01.1	ERAY_CUST1.RISA = 01_B	P01_IOCR0.PC1 = $0XXXX_B$	In
RXD1A2/ NC	ERAY_CUST1.RISA = 10_B		In
RXD1A3/ NC	ERAY_CUST1.RISA = 11_B		In
TXD1A/ P01.12	not applicable	P01_IOCR12.PC12 = $1X110_B$	Out
TXD1A/ P14.10	not applicable	P14_IOCR8.PC10 = $1X111_B$	Out
$\overline{\text{TXEN1A}}$/ P01.14	not applicable	P1_IOCR12.PC14 = $1X110_B$	Out
$\overline{\text{TXEN1A}}$/ P14.9	not applicable	P14_IOCR8.PC9 = $1X111_B$	Out
FlexRay™ Channel 1B			
RXD1B0/ P14.7	ERAY_CUST1.RISB = 00_B	P14_IOCR4.PC7 = $0XXXX_B$	In
RXD1B1/ P01.8	ERAY_CUST1.RISB = 01_B	P01_IOCR8.PC8 = $0XXXX_B$	In
RXD1B2/ NC	ERAY_CUST1.RISB = 10_B		In
RXD1B3/ NC	ERAY_CUST1.RISB = 11_B		In
TXD1B/ P01.13	not applicable	P01_IOCR12.PC13 = $1X110_B$	Out
TXD1B/ P14.5	not applicable	P14_IOCR4.PC5 = $1X111_B$	Out
$\overline{\text{TXEN1B}}$/ P02.15	not applicable	P2_IOCR12.PC15 = $1X110_B$	Out
$\overline{\text{TXEN1B}}$/ P14.6	not applicable	P14_IOCR4.PC6 = $1X111_B$	Out

41.3.17.3 On-Chip Connections

This section describes all on-chip interconnections of the E-Ray modules except the connections to I/O ports.

41.3.17.3.1 E-Ray Connections with IR

The E-Ray module of the AURIX™ TC3xx Platform has several on-chip interconnections to the IR. Table below shows these interconnections. These enable the IR to handle different service request of E-Ray module via the DMA or Interrupt Service Routine.

Table 406 Request Assignment for IR

Line #	ERAY Output Signal	IR Request Input Line
00	INT0	SRC_ERAY0INT0
01	INT1	SRC_ERAY0INT1
02	TINT0	SRC_ERAY0TINT0
03	TINT1	SRC_ERAY0TINT1
04	NDAT0	SRC_ERAY0NDAT0
05	NDAT1	SRC_ERAY0NDAT1
06	MBSC0	SRC_ERAY0MBSC0
07	MBSC1	SRC_ERAY0MBSC1
08	OBUSY	SRC_ERAY0OBUSY
09	IBUSY	SRC_ERAY0IBUSY
10	INT0	SRC_ERAY1INT0
11	INT1	SRC_ERAY1INT1
12	TINT0	SRC_ERAY1TINT0
13	TINT1	SRC_ERAY1TINT1
14	NDAT0	SRC_ERAY1NDAT0
15	NDAT1	SRC_ERAY1NDAT1
16	MBSC0	SRC_ERAY1MBSC0
17	MBSC1	SRC_ERAY1MBSC1
18	OBUSY	SRC_ERAY1OBUSY
19	IBUSY	SRC_ERAY1IBUSY

41.3.17.3.2 E-Ray Connections with SMU

The E-Ray module of the AURIX™ TC3xx Platform provides to the SMU the following alarms (ALMx): SRAM ECC single bit correction, SRAM ECC uncorrectable error, SRAM address error, SRAM buffer address error.

41.3.17.3.3 E-Ray Connections with the External Request Unit of SCU

The E-Ray module of the AURIX™ TC3xx Platform has several on-chip interconnections to the External Request Unit (ERU) in the SCU to externally trigger stop watch events and to provide a global time e.g. to the on chip timers. Table 24-29 and Table below show these interconnections.

STPWT0_ are the inputs of ERAY0, STPWT1_ of ERAY1.

Table 407 External Stop Watch Request Assignment

ERAY Input Signal	ERU Request Output Line	Selected by
STPWT0_0/1_0	ERU_PDOUT0	CUST1.STPWTS = 00_B
STPWT0_1/1_1	ERU_PDOUT1	CUST1.STPWTS = 01_B
STPWT0_2/1_2	ERU_PDOUT2	CUST1.STPWTS = 10_B
STPWT0_3/1_3	ERU_PDOUT3	CUST1.STPWTS = 11_B

Table 408 Global Macrotick Connection to ERU

ERAY Output Signal	ERU Request Input Line	Selected by
MT0 (from ERAY0)	ERU_IN23	ERU_EICR1.EXIS0 = 11_B
MT1 (from ERAY1)	ERU_IN73	ERU_EICR3.EXIS1 = 11_B

41.3.17.3.4 E-Ray Connections to GTM

The E-Ray module of the AURIX™ TC3xx Platform has several on-chip interconnections to the Generic Timer Module (GTM). Table below show these interconnection s.

Table 409 Global Macrotick Connection to GTM

ERAY Output Signal	TIM Input Line
MT0	TIM0_7
MT0	TIM1_7
MT0	TIM2_7
MT0	TIM3_7
MT1	TIM4_7
MT1	TIM5_7

41.3.17.3.5 E-Ray Connections with the External Clock Output of SCU

The E-Ray module of the AURIX™ TC3xx Platform has one on-chip interconnections to the External Clock Output Unit in the SCU to distribute externally as also internally the Macro Tick as time base for distributed system control. Table below shows this interconnection.

Table 410 Global Macrotick Connection to External Clock Output

ERAY Output Signal	External Clock Output	Selected by
MT0 (from ERAY0)	f_{MT0}	SCU_EXTCON.SEL0 = 1111_B

41.3.17.4 OCDS Trigger Bus (OTGB) Interface

The E-Ray module has two 16 bit and one 32 bit Trigger Sets (**Table 411**) which are selected with the OTSS register.

Table 411 E-Ray Trigger Sets

Trigger Set	Details
TS16_SEP Service Requests, Errors and POC State	**Table 413**

Table 411 E-Ray Trigger Sets (cont'd)

Trigger Set	Details
TS16_MC Macrotick Counter	Table 414
TS32_SCSC State, Cycle and Slot Counter	Table 415

Table 412 shows all possible Trigger Set mapping options. If OTGB0 and/or OTGB1 is not used for E-Ray, Trigger Sets of other sources can be added in the OTGM module.

Table 412 Trigger Set Mapping Options

Width	OTGB0	OTGB1	OTGB2
32 Bit	TS16_SEP/MC		
		TS16_SEP/MC	
	TS16_SEP/MC	TS16_SEP/MC	
64 Bit			TS32_SCSC
	TS16_SEP/MC		TS32_SCSC
		TS16_SEP/MC	TS32_SCSC
	TS16_SEP/MC	TS16_SEP/MC	TS32_SCSC

Table 413 TS16_SEP Service Requests, Errors and POC State

Bits	Description
0	Interrupt 0 Service Request (INT0SRC)
1	Interrupt 1 Service Request (INT1SRC)
2	Timer Interrupt 0 Service Request (TINT0SRC)
3	Timer Interrupt 1 Service Request (TINT1SRC)
4	New Data 0 Service Request (NDAT0SRC)
5	New Data 1 Service Request (NDAT1SRC)
6	Message Buffer Status Changed 0 Service Request (MBSC0SRC)
7	Message Buffer Status Changed 1 Service Request (MBSC1SRC)
8	Output Buffer Busy Service Request (OBUSYSRC)
9	Input Buffer Busy Service Request (IBUSYSRC)
10	Reserved
11	Error on Channel A (EIR.EDA)
12	Error on Channel B (EIR.EDB)
[15:13]	POC State bits[2:0] (CCSV.POCS)

Table 414 TS16_MC Macrotick Counter

Bits	Description
[13:0]	Macrotick Value (MTCCV.MTV)
[15:14]	Reserved

Table 415 TS32_SCSC State, Cycle and Slot Counter

Bits	Description
[10:0]	Slot Counter Channel A (SCV.SCCA)
[22:12]	Slot Counter Channel B (SCV.SCCB)
[29:24]	Cycle Counter Value (MTCCV.CCV)
30	Transfer Input Buffer Completed (SIR.TIBC)
31	Transfer Output Buffer Completed (SIR.TOBC)
11,23	Reserved

TS32_SCSC becomes valid (posedge on otgb2_valid_o) on a change of any Slot Counter or Transfer Buffer state. This covers also the Cycle Counter which will always change with a slower rate.

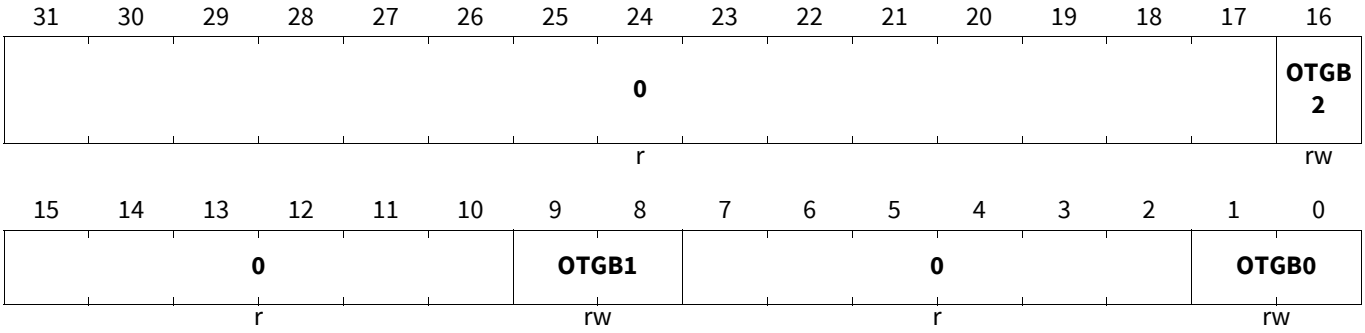
41.3.17.5 OTGB E-Ray Registers

41.3.17.5.1 OCDS Trigger Bus (OTGB)

The OTGB control register is cleared by Debug Reset. Write access is 32 bit wide only and requires Supervisor Mode.

OCDS Trigger Set Select

OTSS
OCDS Trigger Set Select **(0870_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
OTGB0	1:0	rw	**Trigger Set for OTGB0** 00_B No Trigger Set selected 01_B Trigger Set TS16_SEP 10_B Trigger Set TS16_MC 11_B reserved
OTGB1	9:8	rw	**Trigger Set for OTGB1** 00_B No Trigger Set selected 01_B Trigger Set TS16_SEP 10_B Trigger Set TS16_MC 11_B reserved
OTGB2	16	rw	**Trigger Set for OTGB2** 0_B No Trigger Set selected 1_B Trigger Set TS32_SCSC
0	7:2, 15:10, 31:17	r	**Reserved** Read as 0; must be written with 0.

41.3.17.6 BPI_FPI Module Registers

Note: Register bits marked “r” in the following register description are virtual registers and do not contain flip-flops. They are always read as 0.

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock, sleep mode and disable mode for the module.

Note: The number of module clock cycles (wait states) which are required by the kernel to execute a read or write access depends on the selected CLC clock frequency, which is selected via bit field RMC in the CLC register. Therefore, increasing CLC.RMC may result in a longer FPI Bus read cycle access time for kernel registers and can also slow down the write throughput to the kernel registers.

CLC
Clock Control Register **(0000_H)** **Application Reset Value: $0000\ 0003_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					RMC			0				EDIS	0	DISS	DISR
r					rw			r				rw	r	rh	rw

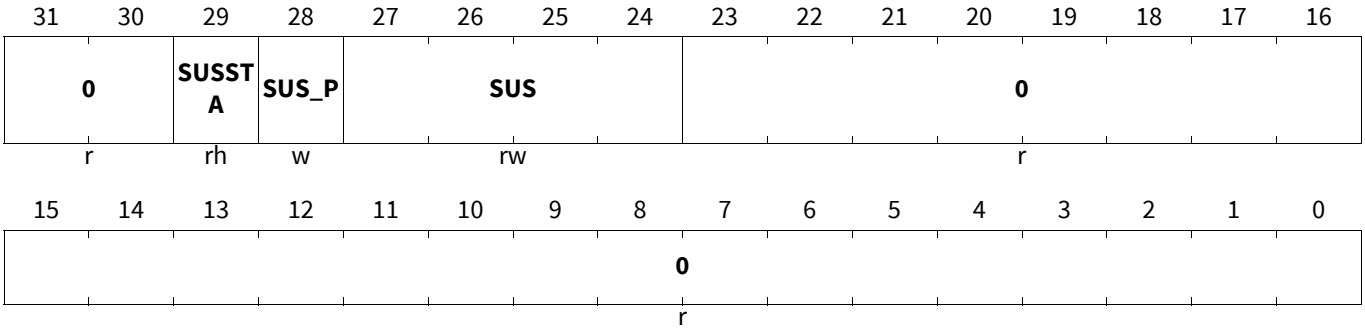
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. *Note: This bit disables the kernel clocks f_{CLC_ERAY} and the sampling clock f_{SCLK}.*
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**External Sleep Mode Request Disable Bit** Used to control module’s sleep mode. *Note: If this bit is cleared the kernel clock f_{CLC_ERAY} and the sampling clock f_{SCLK} are disabled during System Sleep Mode.*

Field	Bits	Type	Description
RMC	10:8	rw	**Clock Divider in Run Mode** *Note: This bit field is not affected by an application reset.* *Note: This bit field only controls the kernel clock f_{CLC_ERAY} and not the sampling clock f_{SCLK}.* 000_B No clock signal f_{CLC_ERAY} generated (default after reset) 001_B Clock $f_{CLC\_ERAY} = f_{SPB}$ selected 010_B Clock $f_{CLC\_ERAY} = f_{SPB}$ /2 selected 011_B reserved, do not use! 100_B Clock $f_{CLC\_ERAY} = f_{SPB}$ /4 selected 101_B reserved, do not use! … 111_B reserved, do not use!
0	2, 7:4, 15:11, 31:16	r	**Reserved** Read as 0; should be written with 0.

OCDS Control and Status

The OCDS Control and Status (OCS) register is cleared by Debug Reset. The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective.

OCS
OCDS Control and Status **($08E8_H$)** **Debug Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clocks f_{CLC_ERAY} and the sampling clock f_{SCLK} are switched off immediately. No read or write access to any registers. 2_H Soft suspend. This bit forces the module into freeze state. **others**, reserved

Field	Bits	Type	Description
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:0, 31:30	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access. [1]for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus Master TAG Assignments Chapter). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, … , EN31 -> TAG ID 011111B.

ACCEN0

Access Enable Register 0 **($08FC_H$)** **Application Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

KRST0

Kernel Reset Register 0 **(08F4$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RSTSTAT	RST
r														rh	rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers (KRST1.RST and KRST0.RST) related to the module kernel that should be reset (kernel 0 or kernel 1). The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1

Kernel Reset Register 1 **(08F0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

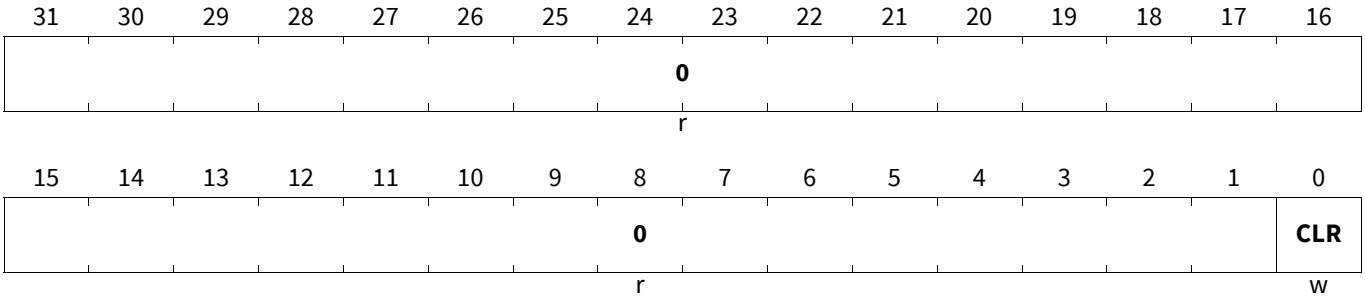
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															RST
r															rwh

Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Register Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR
Kernel Reset Status Clear Register **($08EC_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

41.3.17.7 Interrupt Registers

Two different type of Interrupt Registers are described within this chapter.
The Interrupt Control register enable the selection of the Service Request used to signal an event. The Interrupt Control registers NDIC1 to NDIC4 select the service request node used for New Data Events.The Interrupt Control registers MSIC1 to MSIC4 select the service request node used for Message Buffer Status Changed Events.
The Interrupt Service Request Control Registers control the eight service request nodes.

New Data Interrupt Control 1

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 0 to Message Buffers 31.

NDIC1

New Data Interrupt Control 1 **($03A8_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP31	**NDIP30**	**NDIP29**	**NDIP28**	**NDIP27**	**NDIP26**	**NDIP25**	**NDIP24**	**NDIP23**	**NDIP22**	**NDIP21**	**NDIP20**	**NDIP19**	**NDIP18**	**NDIP17**	**NDIP16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP15	**NDIP14**	**NDIP13**	**NDIP12**	**NDIP11**	**NDIP10**	**NDIP9**	**NDIP8**	**NDIP7**	**NDIP6**	**NDIP5**	**NDIP4**	**NDIP3**	**NDIP2**	**NDIP1**	**NDIP0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n=0-31)	n	rw	**New Data Interrupt Pointer n (n = 0-31)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

New Data Interrupt Control 2

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 32 to Message Buffers 63.

NDIC2

New Data Interrupt Control 2 **($03AC_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP63	**NDIP62**	**NDIP61**	**NDIP60**	**NDIP59**	**NDIP58**	**NDIP57**	**NDIP56**	**NDIP55**	**NDIP54**	**NDIP53**	**NDIP52**	**NDIP51**	**NDIP50**	**NDIP49**	**NDIP48**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP47	**NDIP46**	**NDIP45**	**NDIP44**	**NDIP43**	**NDIP42**	**NDIP41**	**NDIP40**	**NDIP39**	**NDIP38**	**NDIP37**	**NDIP36**	**NDIP35**	**NDIP34**	**NDIP33**	**NDIP32**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n=32-63)	n-32	rw	**New Data Interrupt Pointer n (n = 32-63)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

New Data Interrupt Control 3

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 64 to Message Buffers 95.

NDIC3

New Data Interrupt Control 3 **($03B0_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP95	**NDIP94**	**NDIP93**	**NDIP92**	**NDIP91**	**NDIP90**	**NDIP89**	**NDIP88**	**NDIP87**	**NDIP86**	**NDIP85**	**NDIP84**	**NDIP83**	**NDIP82**	**NDIP81**	**NDIP80**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP79	**NDIP78**	**NDIP77**	**NDIP76**	**NDIP75**	**NDIP74**	**NDIP73**	**NDIP72**	**NDIP71**	**NDIP70**	**NDIP69**	**NDIP68**	**NDIP67**	**NDIP66**	**NDIP65**	**NDIP64**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n=64-95)	n-64	rw	**New Data Interrupt Pointer n (n = 64-95)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

New Data Interrupt Control 4

This New Data Interrupt Control register controls the interrupt that becomes active (NDAT0SRC or NDAT1SRC) on a ND flag turning active of all configured Message Buffers 96 to Message Buffers 127.

NDIC4

New Data Interrupt Control 4 **($03B4_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NDIP127	**NDIP126**	**NDIP125**	**NDIP124**	**NDIP123**	**NDIP122**	**NDIP121**	**NDIP120**	**NDIP119**	**NDIP118**	**NDIP117**	**NDIP116**	**NDIP115**	**NDIP114**	**NDIP113**	**NDIP112**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NDIP111	**NDIP110**	**NDIP109**	**NDIP108**	**NDIP107**	**NDIP106**	**NDIP105**	**NDIP104**	**NDIP103**	**NDIP102**	**NDIP101**	**NDIP100**	**NDIP99**	**NDIP98**	**NDIP97**	**NDIP96**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NDIPn (n=96-127)	n-96	rw	**New Data Interrupt Pointer n (n = 96-127)** NDIPn determines the interrupt (NDAT0SRC or NDAT1SRC) of the service request output that becomes active on a New Data Flag becoming active. 0_B NDAT0SRC selected for New Data Service Request 1_B NDAT1SRC selected for New Data Service Request

Message Buffer Status Changed Interrupt Control 1

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 0 to Message Buffer 31 turning active.

MSIC1

Message Buffer Status Changed Interrupt Control 1(03B8$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP31	**MSIP30**	**MSIP29**	**MSIP28**	**MSIP27**	**MSIP26**	**MSIP25**	**MSIP24**	**MSIP23**	**MSIP22**	**MSIP21**	**MSIP20**	**MSIP19**	**MSIP18**	**MSIP17**	**MSIP16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP15	**MSIP14**	**MSIP13**	**MSIP12**	**MSIP11**	**MSIP10**	**MSIP9**	**MSIP8**	**MSIP7**	**MSIP6**	**MSIP5**	**MSIP4**	**MSIP3**	**MSIP2**	**MSIP1**	**MSIP0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n=0-31)	n	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 0-31)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

Message Buffer Status Changed Interrupt Control 2

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 32 to Message Buffer 63 turning active.

MSIC2

Message Buffer Status Changed Interrupt Control 2(03BC$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP63	**MSIP62**	**MSIP61**	**MSIP60**	**MSIP59**	**MSIP58**	**MSIP57**	**MSIP56**	**MSIP55**	**MSIP54**	**MSIP53**	**MSIP52**	**MSIP51**	**MSIP50**	**MSIP49**	**MSIP48**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP47	**MSIP46**	**MSIP45**	**MSIP44**	**MSIP43**	**MSIP42**	**MSIP41**	**MSIP40**	**MSIP39**	**MSIP38**	**MSIP37**	**MSIP36**	**MSIP35**	**MSIP34**	**MSIP33**	**MSIP32**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n=32-63)	n-32	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 32-63)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

Message Buffer Status Changed Interrupt Control 3

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 64 to Message Buffer 95 turning active.

MSIC3

Message Buffer Status Changed Interrupt Control 3($03C0_H$) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP95	**MSIP94**	**MSIP93**	**MSIP92**	**MSIP91**	**MSIP90**	**MSIP89**	**MSIP88**	**MSIP87**	**MSIP86**	**MSIP85**	**MSIP84**	**MSIP83**	**MSIP82**	**MSIP81**	**MSIP80**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP79	**MSIP78**	**MSIP77**	**MSIP76**	**MSIP75**	**MSIP74**	**MSIP73**	**MSIP72**	**MSIP71**	**MSIP70**	**MSIP69**	**MSIP68**	**MSIP67**	**MSIP66**	**MSIP65**	**MSIP64**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n=64-95)	n-64	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 64-95)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

Message Buffer Status Changed Interrupt Control 4

This Message Buffer Status Change Interrupt Control register controls the interrupt that becomes active (MBSC0SRC or MBSC1SRC) on a MBC flag of all configured Message Buffer 96 to Message Buffer 127 turning active.

MSIC4

Message Buffer Status Changed Interrupt Control 4($03C4_H$) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSIP127	**MSIP126**	**MSIP125**	**MSIP124**	**MSIP123**	**MSIP122**	**MSIP121**	**MSIP120**	**MSIP119**	**MSIP118**	**MSIP117**	**MSIP116**	**MSIP115**	**MSIP114**	**MSIP113**	**MSIP112**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSIP111	**MSIP110**	**MSIP109**	**MSIP108**	**MSIP107**	**MSIP106**	**MSIP105**	**MSIP104**	**MSIP103**	**MSIP102**	**MSIP101**	**MSIP100**	**MSIP99**	**MSIP98**	**MSIP97**	**MSIP96**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSIPn (n=96-127)	n-96	rw	**Message Buffer Status Changed Interrupt Pointer n (n = 96-127)** MSIPn determines the interrupt (MBSC0SRC or MBSC1SRC) of the service request output that becomes active on a Message Buffer Status Changed Flag becoming active. 0_B MBSC0SRC selected for Message Buffer Status Changed Service Request 1_B MBSC1SRC selected for Message Buffer Status Changed Service Request

41.4 Registers

The programmer's model of the E-Ray module follows the principle of memory mapped peripheral. Some portion of the memory follows the principle of segmented/paged memory organization.

41.4.1 Register Map

The E-Ray module allocates an address space of 4 Kbyte (000_H to $FFFF_H$). The registers are organized as 32-bit registers. 8/16-bit accesses are also supported. Host access to the Message RAM is done via the Input and Output Buffers. They buffer data to be transferred to and from the Message RAM under control of the Message Handler, avoiding conflicts between Host accesses and message reception / transmission. Addresses 0004_H - $000F_H$,$03C8_H$ - $03EC_H$ and 0800_H - $0FFF_H$ are reserved for customer specific purposes. All functions related to these addresses are located in the Customer Host Interface. The test registers located on address 0010_H and 0014_H are writable only under the conditions described in **Special Registers**.

The assignment of the Message Buffers is done according to the scheme shown in **Table 416** below. The number N of available Message Buffers depends on the payload length of the configured Message Buffers. The maximum number of Message Buffers is 128. The maximum payload length supported is 254 byte.

The Message Buffers are separated into three consecutive groups:

- Static Buffers: Transmit / Receive Buffers assigned to static segment
- Static and Dynamic Buffers: Transmit / Receive Buffers assigned to static or dynamic segment
- FIFO- Receive FIFO

The Message Buffer separation configuration can be changed only in "DEFAULT_CONFIG" or "CONFIG" state only by programming the Message RAM Configuration register (MRC).

The first group starts with Message Buffer 0 and consists of static Message Buffers only. Message Buffer 0 is dedicated to hold the startup / SYNC Frame or the single slot Frame, if node transmit one, as configured by SUCC1.TXST, SUCC1.TXSY, and SUCC1.TSM in the SUC Configuration Register 1 (SUCC1). In addition, Message Buffer 1 may be used for SYNC Frame transmission in case that SYNC Frames or single-slot Frames should have different payloads on the two channels. In this case bit MRC.SPLM has to be programmed to 1 and Message Buffers 0 and 1 have to be configured with the key slot ID and can be (re)configured in “DEFAULT_CONFIG” or “CONFIG” state only.

The second group consists of Message Buffers assigned to the static or to the dynamic segment. Message Buffers belonging to this group may be reconfigured during run time from dynamic to static or vice versa depending on the state of MRC.SEC.

The Message Buffers belonging to the third group are concatenated to a single receive FIFO.

Table 416 Assignment of Message Buffers

Message Buffer 0	⇓ Static Buffers	
Message Buffer 1		
…		
	⇓ Static + Dynamic Buffers	⇐ **FDB**
	⇓ FIFO	⇐ **FFB**
Message Buffer N-1		
Message Buffer N		⇐ **LCB**

41.4.2 E-Ray Kernel Registers

This chapter describes all registers of the E-Ray kernel.

All registers in the E-Ray address spaces are reset with the application reset with one exception: OCS is reset with debug reset only.

Table 417 Register Overview - ERAY (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	0000_H	SV,U	SV,E,P	Application Reset	69
CUST1	Busy and Input Buffer Control Register	0004_H	SV,U	SV,U,P	Application Reset	86
ID	Module Identification Register	0008_H	SV,U	BE	Application Reset	86
CUST3	Customer Interface Timeout Counter Register	$000C_H$	SV,U	SV,U,P	Application Reset	88
TEST1	Test Register 1	0010_H	SV,U	SV,U,P	Application Reset	91
TEST2	Test Register 2	0014_H	SV,U	SV,U,P	Application Reset	95
LCK	Lock Register	$001C_H$	SV,U	SV,U,P	Application Reset	97
EIR	Error Service Request Select Register	0020_H	SV,U	SV,U,P	Application Reset	99
SIR	Status Service Request Register	0024_H	SV,U	SV,U,P	Application Reset	102
EILS	Error Service Request Line Select	0028_H	SV,U	SV,U,P	Application Reset	106
SILS	Status Service Request Line Select	$002C_H$	SV,U	SV,U,P	Application Reset	108
EIES	Error Service Request Enable Set	0030_H	SV,U	SV,U,P	Application Reset	110
EIER	Error Service Request Enable Reset	0034_H	SV,U	SV,U,P	Application Reset	112
SIES	Status Service Request Enable Set	0038_H	SV,U	SV,U,P	Application Reset	115
SIER	Status Service Request Enable Reset	$003C_H$	SV,U	SV,U,P	Application Reset	118
ILE	Service Request Line Enable	0040_H	SV,U	SV,U,P	Application Reset	120
T0C	Timer 0 Configuration	0044_H	SV,U	SV,U,P	Application Reset	121
T1C	Timer 1 Configuration	0048_H	SV,U	SV,U,P	Application Reset	122

Table 417 Register Overview - ERAY (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
STPW1	Stop Watch Register 1	$004C_H$	SV,U	SV,U,P	Application Reset	123
STPW2	Stop Watch Register 2	0050_H	SV,U	SV,U,P	Application Reset	125
SUCC1	SUC Configuration Register 1	0080_H	SV,U	SV,U,P	Application Reset	125
SUCC2	SUC Configuration Register 2	0084_H	SV,U	SV,U,P	Application Reset	131
SUCC3	SUC Configuration Register 3	0088_H	SV,U	SV,U,P	Application Reset	132
NEMC	NEM Configuration Register	$008C_H$	SV,U	SV,U,P	Application Reset	133
PRTC1	PRT Configuration Register 1	0090_H	SV,U	SV,U,P	Application Reset	133
PRTC2	PRT Configuration Register 2	0094_H	SV,U	SV,U,P	Application Reset	134
MHDC	MHD Configuration Register	0098_H	SV,U	SV,U,P	Application Reset	135
GTUC01	GTU Configuration Register 1	$00A0_H$	SV,U	SV,U,P	Application Reset	136
GTUC02	GTU Configuration Register 2	$00A4_H$	SV,U	SV,U,P	Application Reset	136
GTUC03	GTU Configuration Register 3	$00A8_H$	SV,U	SV,U,P	Application Reset	137
GTUC04	GTU Configuration Register 4	$00AC_H$	SV,U	SV,U,P	Application Reset	138
GTUC05	GTU Configuration Register 5	$00B0_H$	SV,U	SV,U,P	Application Reset	139
GTUC06	GTU Configuration Register 6	$00B4_H$	SV,U	SV,U,P	Application Reset	140
GTUC07	GTU Configuration Register 7	$00B8_H$	SV,U	SV,U,P	Application Reset	140
GTUC08	GTU Configuration Register 8	$00BC_H$	SV,U	SV,U,P	Application Reset	141
GTUC09	GTU Configuration Register 9	$00C0_H$	SV,U	SV,U,P	Application Reset	141
GTUC10	GTU Configuration Register 10	$00C4_H$	SV,U	SV,U,P	Application Reset	142
GTUC11	GTU Configuration Register 11	$00C8_H$	SV,U	SV,U,P	Application Reset	143

Table 417 Register Overview - ERAY (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CCSV	Communication Controller Status Vector	0100_H	SV,U	BE	Application Reset	145
CCEV	Communication Controller Error Vector	0104_H	SV,U	BE	Application Reset	148
SCV	Slot Counter Value	0110_H	SV,U	BE	Application Reset	149
MTCCV	Macrotick and Cycle Counter Value	0114_H	SV,U	BE	Application Reset	150
RCV	Rate Correction Value	0118_H	SV,U	BE	Application Reset	151
OCV	Offset Correction Value	$011C_H$	SV,U	BE	Application Reset	151
SFS	SYNC Frame Status	0120_H	SV,U	BE	Application Reset	151
SWNIT	Symbol Window and Network Idle Time Status	0124_H	SV,U	BE	Application Reset	153
ACS	Aggregated Channel Status	0128_H	SV,U	SV,U,P	Application Reset	155
ESIDn	Even Sync ID Symbol Window n	0130_H+(n -1)*4	SV,U	BE	Application Reset	157
OSIDn	Odd Sync ID Symbol Window n	0170_H+(n -1)*4	SV,U	BE	Application Reset	158
NMVx	Network Management Vector x	$01B0_H$+(x -1)*4	SV,U	BE	Application Reset	159
MRC	Message RAM Configuration	0300_H	SV,U	SV,U,P	Application Reset	161
FRF	FIFO Rejection Filter	0304_H	SV,U	SV,U,P	Application Reset	163
FRFM	FIFO Rejection Filter Mask	0308_H	SV,U	SV,U,P	Application Reset	164
FCL	FIFO Critical Level	$030C_H$	SV,U	SV,U,P	Application Reset	165
MHDS	Message Handler Status	0310_H	SV,U	SV,U,P	Application Reset	166
LDTS	Last Dynamic Transmit Slot	0314_H	SV,U	SV,U,P	Application Reset	167
FSR	FIFO Status Register	0318_H	SV,U	SV,U,P	Application Reset	168
MHDF	Message Handler Constraints Flags	$031C_H$	SV,U	SV,U,P	Application Reset	169

Table 417 Register Overview - ERAY (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
TXRQ1	Transmission Request Register 1	0320_H	SV,U	BE	Application Reset	171
TXRQ2	Transmission Request Register 2	0324_H	SV,U	BE	Application Reset	172
TXRQ3	Transmission Request Register 3	0328_H	SV,U	BE	Application Reset	172
TXRQ4	Transmission Request Register 4	$032C_H$	SV,U	BE	Application Reset	173
NDAT1	New Data Register 1	0330_H	SV,U	BE	Application Reset	173
NDAT2	New Data Register 2	0334_H	SV,U	BE	Application Reset	174
NDAT3	New Data Register 3	0338_H	SV,U	BE	Application Reset	175
NDAT4	New Data Register 4	$033C_H$	SV,U	BE	Application Reset	175
MBSC1	Message Buffer Status Changed 1	0340_H	SV,U	BE	Application Reset	176
MBSC2	Message Buffer Status Changed 2	0344_H	SV,U	BE	Application Reset	177
MBSC3	Message Buffer Status Changed 3	0348_H	SV,U	BE	Application Reset	177
MBSC4	Message Buffer Status Changed 4	$034C_H$	SV,U	BE	Application Reset	178
NDIC1	New Data Interrupt Control 1	$03A8_H$	SV,U	SV,U,P	Application Reset	73
NDIC2	New Data Interrupt Control 2	$03AC_H$	SV,U	SV,U,P	Application Reset	74
NDIC3	New Data Interrupt Control 3	$03B0_H$	SV,U	SV,U,P	Application Reset	75
NDIC4	New Data Interrupt Control 4	$03B4_H$	SV,U	SV,U,P	Application Reset	75
MSIC1	Message Buffer Status Changed Interrupt Control 1	$03B8_H$	SV,U	SV,U,P	Application Reset	76
MSIC2	Message Buffer Status Changed Interrupt Control 2	$03BC_H$	SV,U	SV,U,P	Application Reset	76
MSIC3	Message Buffer Status Changed Interrupt Control 3	$03C0_H$	SV,U	SV,U,P	Application Reset	77
MSIC4	Message Buffer Status Changed Interrupt Control 4	$03C4_H$	SV,U	SV,U,P	Application Reset	77

Table 417 Register Overview - ERAY (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CREL	Core Release Register	$03F0_H$	SV,U	nBE	Application Reset	180
ENDN	Endian Register	$03F4_H$	SV,U	nBE	Application Reset	181
WRDSn	Write Data Section n	0400_H+(n -1)*4	SV,U	SV,U,P	Application Reset	182
WRHS1	Write Header Section 1	0500_H	SV,U	SV,U,P	Application Reset	183
WRHS2	Write Header Section 2	0504_H	SV,U	SV,U,P	Application Reset	184
WRHS3	Write Header Section 3	0508_H	SV,U	SV,U,P	Application Reset	185
IBCM	Input Buffer Command Mask	0510_H	SV,U	SV,U,P	Application Reset	185
IBCR	Input Buffer Command Request	0514_H	SV,U	SV,U,P	Application Reset	187
RDDSn	Read Data Section n	0600_H+(n -1)*4	SV,U	BE	Application Reset	189
RDHS1	Read Header Section 1	0700_H	SV,U	BE	Application Reset	189
RDHS2	Read Header Section 2	0704_H	SV,U	BE	Application Reset	190
RDHS3	Read Header Section 3	0708_H	SV,U	BE	Application Reset	191
MBS	Message Buffer Status	$070C_H$	SV,U	BE	Application Reset	193
OBCM	Output Buffer Command Mask	0710_H	SV,U	SV,U,P	Application Reset	197
OBCR	Output Buffer Command Request	0714_H	SV,U	SV,U,P	Application Reset	198
OTSS	OCDS Trigger Set Select	0870_H	SV,U	SV,E,P	Application Reset	68
OCS	OCDS Control and Status	$08E8_H$	SV,U	SV,P,OEN	Debug Reset	70
KRSTCLR	Kernel Reset Status Clear Register	$08EC_H$	SV,U	SV,E,P	Application Reset	73
KRST1	Kernel Reset Register 1	$08F0_H$	SV,U	SV,E,P	Application Reset	72
KRST0	Kernel Reset Register 0	$08F4_H$	SV,U	SV,E,P	Application Reset	71
ACCEN0	Access Enable Register 0	$08FC_H$	SV,U	SV,SE	Application Reset	71

Note: Registers covered by the ACCENx write protection mechanism are marked with ´P´ in Access Mode, Write.

41.4.2.1 Customer Registers

The addresses 0004_H - $000F_H$,$03C8_H$ - $03EC_H$ and 0800_H - $0FFF_H$ are reserved for customer-specific registers.

Module Identification Register

This register contains bit fields identifying the E-Ray module in Infineon's Module portfolio and is read only.

ID
Module Identification Register (0008_H) Application Reset Value: 0044 C0XX$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MOD_NUMBER															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOD_TYPE								MOD_REV							
r								r							

Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision Number** MOD_REV defines the module revision number. The value of a module revision starts with 01_H (first revision).
MOD_TYPE	15:8	r	**Module Type** The value of this bit field is $C0_H$. It defines the module as a 32-bit module.
MOD_NUMBER	31:16	r	**Module Number Value** This bit field defines a module identification number. For the E-Ray module the module identification number is 44_H.

Busy and Input Buffer Control Register

The Busy Control Register enables the automatic delay scheme. Furthermore it signals a time-out service request for the automatic delay scheme. IBFS changes 2 clock cycles after any reset to '1'. Thus the application will read this bit as '1' already at first access after any reset.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register.

CUST1
Busy and Input Buffer Control Register (0004_H) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STPWTS		RISB		RISA		0		IBF2PAG	0		IBF1PAG	IBFS	IEN	OEN	INT0
rw		rw		rw		r		rw	r		rw	rh	rw	rw	rwh

Field	Bits	Type	Description
INT0	0	rwh	**CIF Timeout Service Request Status** INT0 will be set if a timeout has occurred during the auto delay scheme and must be reset by writing zero to INT0. Software can also set this bit field. *Note: In case hardware sets INT0 and at the same point of time software clears INT0, INT0 is cleared.*
OEN	1	rw	**Enable auto delay scheme for Output Buffer Control Register (OBCR)** This control bit controls the delay scheme for Output Buffer Control Register (OBCR) read accesses. 0_B Disable auto delay scheme for Output Buffer Control Register (OBCR) 1_B Enable auto delay scheme for Output Buffer Control Register (OBCR)
IEN	2	rw	**Enable auto delay scheme for Input Buffer Control Register (IBCR)** This control bit controls the auto delay scheme for Input Buffer Control Register (IBCR) read accesses. 0_B Disable auto delay scheme for Input Buffer Control Register (IBCR) 1_B Enable auto delay scheme for Input Buffer Control Register (IBCR)
IBFS	3	rh	**Input Buffer Status Register** This status bit indicates which of the two Input Buffer RAMs (IBF) is accessible by the host (via CIF) as Input Buffer. The other non accessible buffer RAM is currently used as shadow buffer RAM by the ERAY message handler and therefore not accessible by the host.After reset, it is set by hardware. 0_B Input Buffer RAM 2 (IBF2) is accessible as Input Buffer by the host (CIF) 1_B Input Buffer RAM 1 (IBF1) is accessible as Input Buffer by the host (CIF)
IBF1PAG	4	rw	**Input Buffer 1 Page Select Register** This control bit selects if the upper page or lower page of Input Buffer 1 (IBF1) currently active. *Note: Write is only possible, if Input Buffer RAM 1 is currently accessible by the host (via CIF) and therefore IBFS set.* 0_B Read: Lower Page (256 Bytes) of Input Buffer RAM 1 selected Write: Select Lower Page (256 Bytes) of Input Buffer RAM 1 1_B Read: Upper Page (256 Bytes) of Input Buffer RAM 1 selected Write: Select Upper Page (256 Bytes) of Input Buffer RAM 1

Field	Bits	Type	Description
IBF2PAG	7	rw	**Input Buffer 2 Page Select Register** This control bit selects if the upper page or lower page of Input Buffer 2 (IBF2) currently active. *Note: Write is only possible, if Input Buffer RAM 2 is currently accessible by the host (via CIF) and therefore IBFS cleared.* 0_B Read: Lower Page (256 Bytes) of Input Buffer RAM 2 selected Write: Select Lower Page (256 Bytes) of Input Buffer RAM 2 1_B Read: Upper Page (256 Bytes) of Input Buffer RAM 2 selected Write: Select Upper Page (256 Byte) of Input Buffer RAM 2
RISA	11:10	rw	**Receive Input Select Channel A** 00_B Channel A receiver input RXDA0 selected 01_B Channel A receiver input RXDA1 selected 10_B Channel A receiver input RXDA2 selected 11_B Channel A receiver input RXDA3 selected
RISB	13:12	rw	**Receive Input Select Channel B** 00_B Channel B receiver input RXDB0 selected 01_B Channel B receiver input RXDB1 selected 10_B Channel B receiver input RXDB2 selected 11_B Channel B receiver input RXDB3 selected
STPWTS	15:14	rw	**Stop Watch Trigger Input Select** 00_B Stop Watch Trigger input STPWT0 selected 01_B Stop Watch Trigger input STPWT1 selected 10_B Stop Watch Trigger input STPWT2 selected 11_B Stop Watch Trigger input STPWT3 selected
0	6:5, 9:8, 31:16	r	**Reserved** Returns 0 if read; should be written with 0.

Customer Interface Timeout Counter Register

The Time-out Counter Register is realizing the time-out counter reload (startup) value for the automatic delay scheme (not the time-out down counter itself).

CUST3

Customer Interface Timeout Counter Register (000C$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TO															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TO															
rw															

Field	Bits	Type	Description
TO	31:0	rw	**CIF Timeout Reload Value** The 32-bit down counter reload (start-up) value must be setup for the automatic delay scheme.

Automatic Delayed Write Access to OBCR and IBCR

Write and read accesses to the Output Buffer Control Register (OBCR) can be automatically stalled due to a ongoing transfer from the Message Buffer to the Output Buffer. Also write and read accesses to the Input Buffer Control Register (IBCR) may be automatically delayed due to a ongoing transfer from the Input Buffer to the Message Buffer.

This delay scheme can be controlled (enabled or disabled) by CUST1.IEN and CUST1.OEN. The maximum time to stall a write or read access is determined by a single time-out counter preluded with the 32-bit value specified in the bit field CUST3.TO. If the time-out counter counts down to zero before the transfer to/from the Message Buffer is completed, the access (read or write) will be canceled and a service request will be generated. A canceled read access provides a 0 value. A canceled write access does not modify any bits in the OBCR or IBCR. In addition the bit CUST1.INT0 of the service request status register will be set and must be reset by the host to disable the service request line.

The read and write access to the Output Buffer Control Register (OBCR) may be configured without automatic delay by clearing CUST1.OEN. Setting OBCR.REQ and immediately afterwards reading or writing OBCR, e.g. to set OBCR.VIEW will lead to a canceled read or write operation, e.g. OBCR.VIEW remains cleared, and an error is signalled by a set EIR.IOBA. Besides canceling the erroneous read or write operation, and setting the error bit, no further state change happens. So full operation is granted. OBCR remains read and write inaccessible until the transfer of data from the Message Buffer to the Output Buffer (MBF⇒ OBF) is completed. During this time span all read and write accesses to the Output Buffer Control Register (OBCR) are canceled. The transfer is completed when OBCR.OBSYS is cleared. Additionally signal TOBC may be used, e.g. for service request triggering, DMA triggering, or driving a pin, to communicate the access status.

The read and write access to the Output Buffer Control Register (OBCR) may be configured to be automatic delayed by setting CUST1.OEN and configuring CUST3.TO to the maximum stall time acceptable to the system. If setting OBCR.REQ and immediately afterwards reading or writing to OBCR, e.g. to set the OBCR.VIEW bit, this read or write will be stalled until either the maximum delay time elapsed (in this case the read or write operation is cancelled after the stall time, e.g. OBCR.VIEW remains cleared, and an error is signalled by setting EIR.IOBA) or the read or write completes normally, e.g. set OBCR.VIEW after the transfer of data from the Message Buffer to the Output Buffer (MBF⇒ OBF) is finalized. During this time the bus is locked and no further access to the E-Ray module is possible due to the ongoing stalled read or write operation. Because no access is possible to the E-Ray module, read or write stall may only be detected through the signal TOBC or due to other not processed read or write accesses to the E-Ray module.

The read and write access to the Input Buffer Control Register (IBCR) may also be configured without automatic delay by clearing CUST1.IEN. By writing to IBCR.IBRH the Input Buffers are swapped (shadow IBF changes to host IBF and host IBF to shadow IBF), the content of the shadow IBF is copied into the MBF (IBF⇒ MBF), and IBCR.IBSYS is set. Writing to IBCR.IBRH a second time while IBCR.IBSYS remained set (previously initiated copy process IBF⇒ MBF ongoing) will correctly update IBCR.IBRH and set IBCR.IBSYH. This will set the signal IBUSY. A third access, read or write, to IBCR while IBCR.IBSYH remains set will cancel this third access and an error is signalled by setting EIR.IIBA. Besides canceling this last access to IBCR and setting the error bit, no further state change happens. So full operation is granted. IBCR remains read and write inaccessible until the transfer of data from the Input Shadow Buffer to the Message Buffer (IBFÞMBF) is completed and once more the Input Buffers are swapped (shadow IBF changes to host IBF and host IBF to shadow IBF). During this time span all read and write accesses to the Input Buffer Control Register (IBCR) are canceled. The transfer is completed when IBCR.IBSYH is

cleared. Additionally signal TIBC may be used, e.g. for service request triggering, DMA triggering, or driving a pin, to communicate the access status.

The read and write access to the Input Buffer Control Register (IBCR) may be configured for being automatically delayed by setting CUST1.IEN and configuring CUST3.TO to the maximum stall time acceptable to the system. By writing to IBCR.IBRH the Input Buffers are swapped (shadow IBF changes to host IBF and host IBF to shadow IBF), the content of the shadow IBF copied into the MBF, and IBCR.IBSYS is set. Writing to IBCR.IBRH a second time while IBCR.IBSYS remains set (previously initiated copy process ongoing) will correctly update IBCR.IBRH and set IBCR.IBSYH. A third access to IBCR while IBCR.IBSYH remains set will stall this read or write until either the maximum delay time elapsed (in this case the read or write operation is cancelled after the stall time and an error is signalled by setting EIR.IOBA) or the read or write completes normally, after the transfer of data from the Input Shadow Buffer to the Message Buffer (IBF$\Rightarrow$ MBF) is finalized and once more the Input Buffers are swapped (shadow IBF changes to host IBF and host IBF to shadow IBF). During this time the bus is locked and no further access to E-Ray module is possible due to the ongoing stalled read or write operation. Because no access is possible to the E-Ray module, read or write stall may only be detected through the signal TIBC or due to other not processed read or write accesses to the E-Ray module.

So setting CUST3.TO = $FFFFFFFF_H$, CUST1.IEN = 1, and CUST1.OEN = 1 will always grant a consistent data access of the host to the Output and Input Buffers without the need of reading and taking into account the status of OBCR.OBSYS or IBCR.IBSYH. But this simplified access may cause system latencies and system performance loss.

41.4.2.2 Special Registers

Test Register 1

The Test Register 1 holds the control bits to configure the test modes of the E-Ray module. Write access to these bits is only possible if bit TEST1.WRTEN is set.

The Test Register 1 bits therefore can be used to test the interface to the physical layer (connectivity test) by driving / reading the respective pins.

When the E-Ray IP is operated in one of its test modes that requires TEST1.WRTEN to be set (RAM Test Mode, I/O Test Mode, Asynchronous Transmit Mode, and Loop Back Mode) only the selected test mode functionality is available.

The test functions are not available in addition to the normal operational mode functions, they change the functions of parts of the E-Ray module. Therefore normal operation as specified outside this chapter and as required by the FlexRay™ protocol specification and the FlexRay™ conformance test is not possible. Test mode functions may not be combined with each other or with FlexRay™ protocol functions.

The test mode features are intended for hardware testing or for FlexRay™ bus analyzer tools. They are not intended to be used in FlexRay™ applications

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register. In CERA and CERB, Coding errors are also signalled when the Communication Controller is in "MONITOR_MODE". The error codes regarding CAS / MTS symbols concern only the monitored bit pattern, irrelevant whether those bit patterns are seen in the symbol window or elsewhere.

TEST1
Test Register 1 **(0010_H)** **Application Reset Value: $0000\ 0300_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CERB				CERA				0		TXENB	TXENA	TXB	TXA	RXB	RXA
rh				rh				r		rwh	rwh	rwh	rwh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						AOB	AOA	0		TMC		0		ELBE	WRTEN
r						rh	rh	r		rw		r		rw	rw

Field	Bits	Type	Description
WRTEN	0	rw	**Write Test Register Enable** Enables write access to the test registers. To set the bit from 0 to 1 the test mode key has to be written as defined on “Lock Register”. The unlock sequence is not required when TEST1.WRTEN is kept at 1 while other bits of the register are changed. The bit can be reset to 0 at any time. 0_B Write access to test registers disabled. 1_B Write access to test registers enabled.

Field	Bits	Type	Description
ELBE	1	rw	**External Loop Back Enable** There are two possibilities to perform a loop back test. External loop back via physical layer or internal loop back for in-system self-test (default). In case of an internal loop back pins $\overline{\text{TXENA}}$ and $\overline{\text{TXENB}}$ are in their inactive state, pins TXDA and TXDB are set to HIGH, pins RXDA and RXDB are not evaluated. Bit ELBE is evaluated only when POC is in loop back mode and test multiplexer control is in non multiplexed mode TMC = 00. 0_B Internal loop back (default) 1_B External loop back
TMC	5:4	rw	**Test Multiplexer Control** 00_B Normal signal path (default). 01_B RAM Test Mode: Internal busses are multiplexed to make all RAM blocks of the E-Ray module directly accessible by the Host. This mode is intended to enable testing of the embedded RAM blocks during production testing. 10_B I/O Test Mode: Output pins are driven to the values defined by bits TXA, TXB, $\overline{\text{TXENA}}$, $\overline{\text{TXENB}}$. The values applied to the input pins can be read from register bits RXA and RXB. 11_B Reserved; should not be used.
AOA	8	rh	**Activity on A** The channel idle condition is specified in the FlexRay™ protocol spec v2.1, chapter 3, BITSTRB process (zChannelIdle). 0_B No activity detected, channel A idle 1_B Activity detected, channel A not idle
AOB	9	rh	**Activity on B** The channel idle condition is specified in the FlexRay™ protocol spec v2.1, chapter 3, BITSTRB process (zChannelIdle). 0_B No activity detected, channel B idle 1_B Activity detected, channel B not idle
RXA	16	rh	**Read Channel A Receive Pin** This bit field shows the current logic state of RXDA. 0_B RXDA = 0 1_B RXDA = 1
RXB	17	rh	**Read Channel B Receive Pin** This bit field shows the current logic state of RXDB. 0_B RXDB = 0 1_B RXDB = 1
TXA	18	rwh	**Read or Write to Channel A Transmit Pin** A write to this bit field sets the TXDA to corresponding logic state. A read from this bit field shows the current logic state of TXDA. 0_B TXDA = 0 1_B TXDA = 1

Field	Bits	Type	Description
TXB	19	rwh	**Read or Write to Channel B Transmit Pin** A write to this bit field sets the TXDB to corresponding logic state. A read from this bit field shows the current logic state of TXDB. 0_B TXDB = 0 1_B TXDB = 1
TXENA	20	rwh	**Read or Write to Channel A Transmit Enable Pin** A write to this bit field sets the TXENA to corresponding logic state. A read from this bit field shows the current logic state of TXENA. 0_B $\overline{\text{TXENA}}$= 0 1_B $\overline{\text{TXENA}}$= 1
TXENB	21	rwh	**Read or Write to Channel B Transmit Enable Pin** A write to this bit field sets the TXENB to corresponding logic state. A read from this bit field shows the current logic state of TXENB. 0_B $\overline{\text{TXENB}}$ = 0 1_B $\overline{\text{TXENB}}$= 1
CERA	27:24	rh	**Coding Error Report Channel A** Set when a coding error is detected on channel A. Reset to zero when register TEST1 is read or written. Once the CERA is set it will remain unchanged until the Host accesses the TEST1 register. Other combinations are reserved. 0_H No coding error detected 1_H Header CRC error detected 2_H Frame CRC error detected 3_H Frame Start Sequence FSS too long 4_H First bit of Byte Start Sequence BSS seen LOW 5_H Second bit of Byte Start Sequence BSS seen HIGH 6_H First bit of Frame End Sequence FES seen HIGH 7_H Second bit of Frame End Sequence FES seen LOW 8_H CAS / MTS symbol seen too short 9_H CAS / MTS symbol seen too long
CERB	31:28	rh	**Coding Error Report Channel B** Set when a coding error is detected on channel B. Reset to zero when register TEST1 is read or written. Once the CERB is set it will remain unchanged until the Host accesses the TEST1 register. Other combinations are reserved. 0_H No coding error detected 1_H Header CRC error detected 2_H Frame CRC error detected 3_H Frame Start Sequence FSS too long 4_H First bit of Byte Start Sequence BSS seen LOW 5_H Second bit of Byte Start Sequence BSS seen HIGH 6_H First bit of Frame End Sequence FES seen HIGH 7_H Second bit of Frame End Sequence FES seen LOW 8_H CAS / MTS symbol seen too short 9_H CAS / MTS symbol seen too long

Field	Bits	Type	Description
0	3:2, 7:6, 15:10, 23:22	r	**Reserved** Returns 0 if read; should be written with 0.

Asynchronous Transmit Mode (ATM)

The asynchronous transmit mode is entered by writing 1110_B to the CHI Command Vector SUCC1.CMD in the SUC Configuration Register 1 (CHI command: ATM) while the Communication Controller is in “CONFIG” state and bit TEST1.WRTEN in the Test Register 1 is set. This write operation has to be directly preceded by two consecutive write accesses to the Configuration Lock Key (unlock sequence). When called in any other state or when bit TEST1.WRTEN is not set, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. CCSV.POCS in the Communication Controller Status Vector will return 1110_B while the E-Ray module is in ATM mode. Asynchronous Transmit mode can be left by writing 0001_B (CHI command: “CONFIG”) to the CHI Command Vector SUCC1.CMD in the SUC Configuration Register 1.

In ATM mode transmission of a FlexRay™ Frame is triggered by writing the number of the respective Message Buffer to the Input Buffer Command Request register (IBCR.IBRH) while bit IBCM.STXRS in the Input Buffer Command Mask register is set to 1. In this mode wake-up, startup, and clock synchronization are bypassed. The CHI command SEND_MTS results in the immediate transmission of an MTS symbol.

The cycle counter value of Frames send in ATM mode can be programmed via MTCCV.CCV (writable in ATM and loop back mode only).

Loop Back Mode

The loop back mode is entered by writing 1111_B to the CHI Command Vector SUCC1.CMD in the SUC Configuration Register 1 (CHI command: LOOP_BACK) while the Communication Controller is in “CONFIG” state and bit TEST1.WRTEN in the Test Register 1 is set. This write operation has to be directly preceded by two consecutive write accesses to the Configuration Lock Key (unlock sequence). When called in any other state or when bit TEST1.WRTEN is not set, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. CCSV.POCS in the Communication Controller Status Vector will show $0000\ 1101_H$ while the E-Ray module is in loop back mode.

Loop Back mode can be left by writing 0001_B (CHI command: “CONFIG”) to the CHI Command Vector SUCC1.CMD in the SUC Configuration Register 1.

The loop back test mode is intended to check the module’s internal data paths. Normal, time triggered operation is not possible in loop back mode.

There are two possibilities to perform a loop back test. External loop back via physical layer (TEST1.ELBE = 1) or internal loop back for in-system self-test (TEST1.ELBE = 0). In case of an internal loop back pins $\overline{\text{TXENA}}$, $\overline{\text{TXENB}}$are in their inactive state, pins TXDA and TXDB are set to HIGH, pins RXDAn and RXDBn are not evaluated.

When the Communication Controller is in loop back mode, a loop back test is started by the Host writing a message to the Input Buffer and requesting the transmission by writing to the Input Buffer Command Request register IBCR. The Message Handler will transfer the message into the Message RAM and then into the Transient Buffer of the selected channel. The Channel Protocol Controller (PRT) will read (in 32-bit words) the message from the transmit part of the Transient Buffer and load it into its Rx / Tx shift register. The serial transmission is looped back into the shift register; its content is written into the receive part of the channels’s Transient Buffer before the next word is loaded.

The PRT and the Message Handler will then treat this transmitted message like a received message, perform an acceptance filtering on Frame ID and receive channel, and store the message into the Message RAM if it passed acceptance filtering. The loop back test ends with the Host requesting this received message from the Message RAM and then checking the contents of the Output Buffer.

Each FlexRay™ channel is tested separately. The E-Ray cannot receive messages from the FlexRay™ bus while it is in the loop back mode.

The cycle counter value of Frames used in loop back mode can be programmed via MTCCV.CCV (writable in ATM and loop back mode only).

Note that in case of an odd payload the last two bytes of the looped-back payload will be shifted by 16 bits to the right inside the last 32-bit data word.

Test Register 2

The Test Register 2 holds all bits required for the RAM test of the seven embedded RAM blocks of the E-Ray module. Write access to this register is only possible when TEST1.WRTEN in the Test Register 1 is set to 1.

TEST2
Test Register 2 **(0014_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WREC	0							SSEL			0	RS		
r	rw	r							rw			r	rw		

Field	Bits	Type	Description
RS	2:0	rw	**RAM Select** In RAM Test mode the RAM blocks selected by RS are mapped to module address $0000\ 0400_H$ to $0000\ 07FF_H$ (1024 byte addresses). 000_B Input Buffer RAM 1 (IBF1) 001_B Input Buffer RAM 2 (IBF2) 010_B Output Buffer RAM 1 (OBF1) 011_B Output Buffer RAM 2 (OBF2) 100_B Transient Buffer RAM A (TBF1) 101_B Transient Buffer RAM B (TBF2) 110_B Message RAM (MBF) 111_B Reserved; should not be used.
SSEL	6:4	rw	**Segment Select** To enable access to the complete Message RAM (8192 byte addresses) the Message RAM is segmented. 000_B access to RAM byte 0000_H to $03FF_H$ enabled 001_B access to RAM byte 0400_H to $07FF_H$ enabled 010_B access to RAM byte 0800_H to $0BFF_H$ enabled 011_B access to RAM byte $0C00_H$ to $0FFF_H$ enabled 100_B access to RAM byte 1000_H to $13FF_H$ enabled 101_B access to RAM byte 1400_H to $17FF_H$ enabled 110_B access to RAM byte 1800_H to $1BFF_H$ enabled 111_B access to RAM byte $1C00_H$ to $1FFF_H$ enabled

Field	Bits	Type	Description
WRECC	14	rw	**Write ECC Data Enable** Content of ECCW is transferred to the RAM: *Note: Test mode must be entered. See "Test Register 1"* 0_B disabled 1_B enabled
0	3, 13:7, 15, 31:16	r	**Reserved** Returns 0 if read; should be written with 0.

RAM Test Mode

In RAM test mode (TEST1.TMC = 1), one of the seven RAM blocks can be selected for direct RD/WR access by programming TEST2.RS.

For external access the selected RAM block is mapped to address space 400_H to $7FF_H$ (1024 byte addresses or 256 word addresses).

Because the length of the Message RAM exceeds the available address space, the Message RAM is segmented into segments of 1024 byte. The segments can be selected by programming TEST2.SSEL in the Test Register 2.

In RAM test mode the memory must be accessed with 32Bit accesses only.

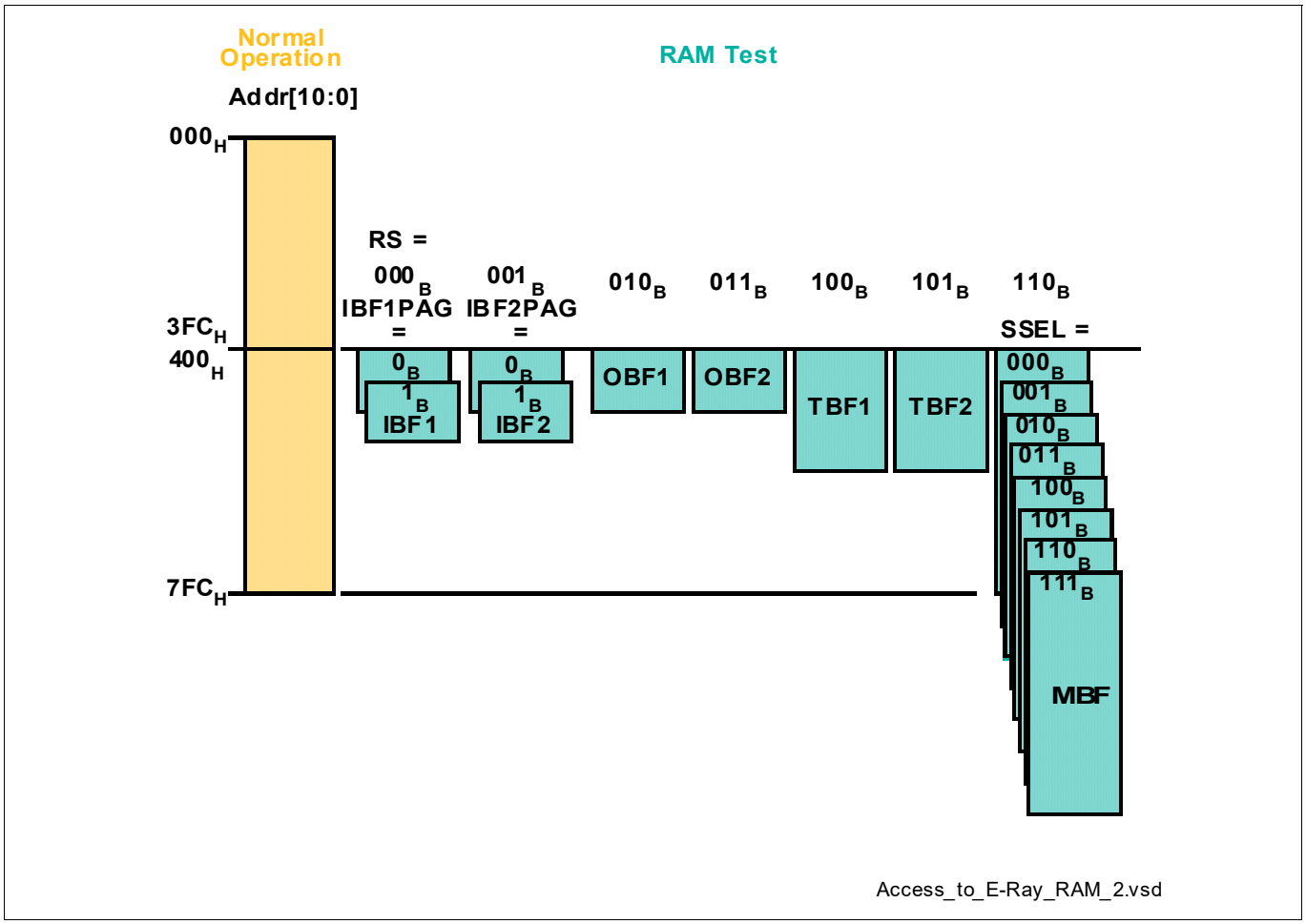


Figure 633 RAM test mode Access to E-Ray RAM Blocks

Lock Register

The Lock Register is write-only. Reading the register will return $0000\ 0000_H$.

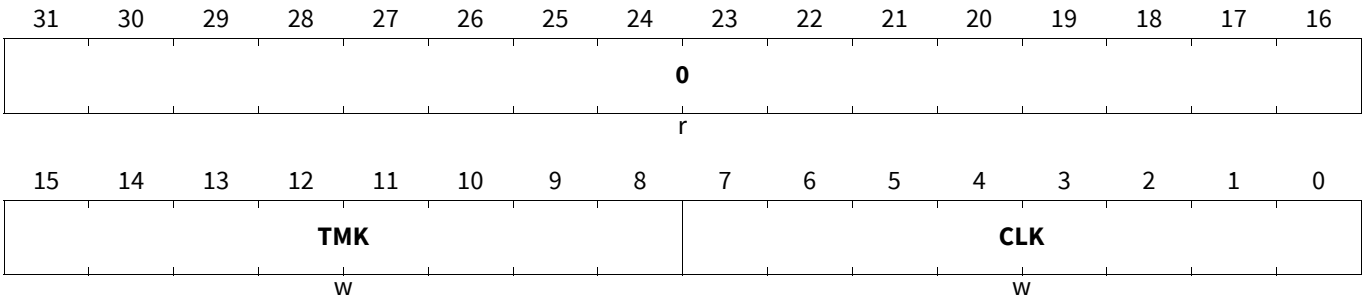
Note: In case the Host uses 8/16-bit accesses to write the listed bit fields, the programmer has to ensure that no "dummy accesses" e.g. to the remaining register bytes / words are inserted by the compiler.

To exit "CONFIG" state by writing to SUCC1.CMD in the SUC Configuration Register 1, the write operation has to be directly preceded by two consecutive write accesses to the Configuration Lock Key. If this write sequence is service requested by read accesses or write accesses to other locations, the Communication Controller remains in "CONFIG" state and the sequence has to be repeated.

First write: LCK.CLK = CE_H = $1100\ 1110_B$

Second write: LCK.CLK = 31_H = 0011 0001_B

LCK
Lock Register **($001C_H$)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CLK	7:0	w	**Configuration Lock Key** To leave “CONFIG” state by writing to SUCC1.CMD commands READY, MONITOR_MODE, ATM, LOOP_BACK) in the SUC Configuration Register 1, the write operation has to be directly preceded by two consecutive write accesses to the Configuration Lock Key (unlock sequence). If the write sequence below is interrupted by other write accesses between the second write to the Configuration Lock Key and the write access to the SUCC1 register, the Communication Controller remains in “CONFIG” state and the sequence has to be repeated. First write: LCK.CLK = CE_H = 1100 1110_B Second write: LCK.CLK = 31_H = 0011 0001_B Third write: SUCC1.CMD Returns 0 if read
TMK	15:8	w	**Test Mode Key** To set bit TEST1.WRTEN the write operation has to be directly preceded by two consecutive write accesses to the Test Mode Key. If the write sequence is interrupted by other write accesses between the second write to the Test Mode Key and the write access to the Test1 register, bit TEST1.WRTEN is not set to 1 and the sequence has to be repeated. First write: LCK.TMK = 75_H = 0111 0101_B Second write: LCK.TMK = $8A_H$ = 1000 1010_B Second write: TEST1.WRTEN = 1 Returns 0 if read
0	31:16	r	**Reserved** Returns 0 if read; should be written with 0.

41.4.2.3 Service Request Registers

The address space from 0020_H to $007F_H$ is reserved for service request registers.

Error Service Request Select Register

The flags are set when the Communication Controller detects one of the listed error conditions. They remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. An application reset will also clear the register.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register.

EIR
Error Service Request Select Register (0020_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					TABB	LTVB	EDB	0					TABA	LTVA	EDA
r					rwh	rwh	rwh	r					rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MHF	IOBA	IIBA	EFA	RFO	EERR	CCL	CCF	SFO	SFBM	CNA	PEMC
r				rwh	rwh	rwh	rwh	rh	rh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
PEMC	0	rwh	**POC Error Mode Changed** This flag is set whenever the error mode signalled by CCEV.ERRM in the Communication Controller Error Vector register has changed. This flag is cleared by writing a 1. 0_B Error mode has not changed 1_B Error mode has changed
CNA	1	rwh	**Command Not Accepted** The flag signals that the write access to the CHI command vector SUCC1.CMD in the SUC Configuration Register 1 was not successful because the requested command was not valid in the actual POC state, or because the CHI command was locked (CCL = 1). This flag is cleared by writing a 1. 0_B CHI command accepted 1_B CHI command not accepted
SFBM	2	rwh	**SYNC Frames Below Minimum** This flag signals that the number of SYNC Frames received during the last communication cycle was below the limit required by the FlexRay™ protocol. May be set during startup and therefore should be cleared by the Host after the Communication Controller entered “NORMAL_ACTIVE” state. This flag is cleared by writing a 1. 0_B Sync node: 1 or more SYNC Frames received Non-sync node: 2 or more SYNC Frames received 1_B Less than the required minimum of SYNC Frames received

Field	Bits	Type	Description
SFO	3	rwh	**SYNC Frame Overflow** Set when either the number of SYNC Frames received during the last communication cycle or the total number of SYNC Frames received during the last double cycle exceeds the maximum number of SYNC Frames as defined by GTUC02.SNM in the GTU Configuration Register 2. This flag is cleared by writing a 1. 0_B Number of received SYNC Frames ≤ GTUC02.SNM 1_B More SYNC Frames received than configured by GTUC02.SNM
CCF	4	rwh	**Clock Correction Failure** This flag is set at the end of the cycle whenever one of the following errors occurred: • Missing offset and / or rate correction • Clock Correction limit reached The clock correction status is monitored in registers CCEV and SFS. A failure may occur during startup, therefore bit CCF should be cleared by the Host after the Communication Controller entered "NORMAL_ACTIVE" state. This flag is cleared by writing a 1. 0_B Clock correction successful so far 1_B Clock correction failed
CCL	5	rwh	**CHI Command Locked** The flag signals that the write access to the CHI command vector SUCC1.CMD was not successful because the execution of the previous CHI command has not yet completed. In this case bit EIR.CNA is also set to 1. This flag is cleared by writing a 1. 0_B CHI command accepted 1_B CHI command not accepted
EERR	6	rh	**ECC Error** The flag signals an ECC error to the Host. It is set whenever one of the flags MHDS.EIBF, MHDS.EOBF, MHDS.EMR, MHDS.ETBF1, MHDS.ETBF2 changes from 0 to 1. See also "Message Handler Status". This bit must be cleared at initialization of the module! 0_B No error detected 1_B Error detected
RFO	7	rh	**Receive FIFO Overrun** The flag is set by the Communication Controller when a receive FIFO overrun is detected. When a receive FIFO overrun occurs, the oldest message is overwritten with the actual received message. The actual state of the FIFO is monitored in register FSR. 0_B No receive FIFO overrun detected 1_B A receive FIFO overrun has been detected

Field	Bits	Type	Description
EFA	8	rwh	**Empty FIFO Access** This flag is set by the Communication Controller when the Host requests the transfer of a message from the receive FIFO via Output Buffer while the receive FIFO is empty. 0_B No Host access to empty FIFO occurred 1_B Host access to empty FIFO occurred
IIBA	9	rwh	**Illegal Input Buffer Access** This flag is set by the Communication Controller when the Host wants to modify a Message Buffer via Input Buffer while the Communication Controller is not in “CONFIG” or “DEFAULT_CONFIG” state and one of the following conditions applies: 1. The Host writes to the Input Buffer Command Request register to modify the: a) Header Section of Message Buffer 0, 1 if configured for transmission in key slot b) Header Section of static Message Buffers with buffer number < MRC.FDB while MRC.SEC = 01_B c) Header Section of any static or dynamic Message Buffer while MRC.SEC = $1x_B$ d) Header and / or Data Section of any message buffer belonging to the receive FIFO 2. The Host writes to any register of the Input Buffer while IBCR.IBSYS is set. 0_B No illegal Host access to Input Buffer occurred 1_B Illegal Host access to Input Buffer occurred
IOBA	10	rwh	**Illegal Output Buffer Access** This flag is set by the Communication Controller when the Host requests the transfer of a Message Buffer from the Message RAM to the Output Buffer while OBCR.OBSYS is set to 1. 0_B No illegal Host access to Output Buffer occurred 1_B Illegal Host access to Output Buffer occurred
MHF	11	rwh	**Message Handler Constraints Flag** The flag signals a Message Handler constraints violation condition. It is set whenever one of the flags MHDF.SNUA, MHDF.SNUB, MHDF.FNFA, MHDF.FNFB, MHDF.TBFA, MHDF.TBFB, MHDF.TNSA, MHDF.TNSB, MHDF.WAHP changes from 0 to 1. 0_B No Message Handler failure detected 1_B Message Handler failure detected
EDA	16	rwh	**Error Detected on Channel A** This bit is set whenever one of the flags ACS.SEDA, ACS.CEDA, ACS.CIA, ACS.SBVA changes from 0 to 1. This flag is cleared by writing a 1. 0_B No error detected on channel A 1_B Error detected on channel A

Field	Bits	Type	Description
LTVA	17	rwh	**Latest Transmit Violation Channel A** The flag signals a latest transmit violation on channel A to the Host. This flag is cleared by writing a 1. 0_B No latest transmit violation detected on channel A 1_B Latest transmit violation detected on channel A
TABA	18	rwh	**Transmission Across Boundary Channel A** The flag signals to the Host that a transmission across a slot boundary occurred for channel A. This flag is cleared by writing a 1. 0_B No transmission across slot boundary detected on channel A 1_B Transmission across slot boundary detected on channel A
EDB	24	rwh	**Error Detected on Channel B** This bit is set whenever one of the flags ACS.SEDB, ACS.CEDB, ACS.CIB, ACS.SBVB changes from 0 to 1. This flag is cleared by writing a 1. 0_B No error detected on channel B 1_B Error detected on channel B
LTVB	25	rwh	**Latest Transmit Violation Channel B** The flag signals a latest transmit violation on channel B to the Host. This flag is cleared by writing a 1. 0_B No latest transmit violation detected on channel B 1_B Latest transmit violation detected on channel B
TABB	26	rwh	**Transmission Across Boundary Channel B** The flag signals to the Host that a transmission across a slot boundary occurred for channel B. This flag is cleared by writing a 1. 0_B No transmission across slot boundary detected on channel B 1_B Transmission across slot boundary detected on channel B
0	15:12, 23:19, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

Status Service Request Register

The flags are set whenever the Communication Controller detects one of the listed events. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. An application reset will also clear the register.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register.

SIR

Status Service Request Register **(0024_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MTSB	WUPB	0						MTSA	WUPA
r						rwh	rwh	r						rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDS	MBSI	SUCS	SWE	TOBC	TIBC	TI1	TI0	NMVC	RFCL	RFNE	RXI	TXI	CYCS	CAS	WST
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rh	rh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
WST	0	rwh	**Wakeup Status** This flag is set when the wakeup status vector CCSV.WSV in the Communication Controller Status Vector register changes to a value other than UNDEFINED. This flag is cleared by writing a 1. 0_B Wake-up status unmodified 1_B Wake-up status modified (and not UNDEFINED)
CAS	1	rwh	**Collision Avoidance Symbol** This flag is set by the Communication Controller during STARTUP state when a CAS or potential CAS was received. This flag is cleared by writing a 1. 0_B No bit pattern matching the CAS symbol received 1_B Bit pattern matching the CAS symbol received
CYCS	2	rwh	**Cycle Start Service Request** This flag is set by the Communication Controller when a communication cycle starts This flag is cleared by writing a 1. 0_B No communication cycle started 1_B Communication cycle started
TXI	3	rwh	**Transmit Service Request** This flag is set by the Communication Controller at the end of Frame transmission if bit WRHS1.MBI in the respective Message Buffer is set. This flag is cleared by writing a 1. 0_B No Frame transmitted from a transmit buffer with WRHS1.MBI = 1 1_B At least one Frame was transmitted from a transmit buffer with WRHS1.MBI = 1
RXI	4	rwh	**Receive Service Request** This flag is set by the Communication Controller whenever the set condition of a Message Buffer ND flag is fulfilled and if bit WRHS1.MBI of that Message Buffer is set to 1. This flag is cleared by writing a 1. 0_B No ND flag of a receive buffer with WRHS1.MBI = 1 has been set to 1 1_B At least one ND flag of a receive buffer with WRHS1.MBI = 1 has been set to 1

Field	Bits	Type	Description
RFNE	5	rh	**Receive FIFO Not Empty** This flag is set by the Communication Controller when a received valid Frame was stored into the empty receive FIFO.m The actual state of the receive FIFO is monitored in register FSR 0_B Receive FIFO is empty 1_B Receive FIFO is not empty
RFCL	6	rh	**Receive FIFO Critical Level** This flag is set when a valid receive FIFO fill level FSR.RFFL is equal or greater than the critical level as configured by FCL.CL. 0_B Receive FIFO below critical level 1_B Receive FIFO critical level reached
NMVC	7	rwh	**Network Management Vector Changed** This service request flag signals a change in the Network Management Vector visible to the Host. This flag is cleared by writing a 1. 0_B No change in the Network Management vector 1_B Network Management vector changed
TI0	8	rwh	**Timer Service Request 0** This flag is set whenever timer 0 matches the conditions configured in the Timer Service Request 0 Configuration Register T0C. A Timer Service Request 0 is also signalled by TINT0SRC. This flag is cleared by writing a 1. 0_B No Timer Service Request 0 1_B Timer Service Request 0 occurred
TI1	9	rwh	**Timer Service Request 1** This flag is set whenever the conditions programmed in the Timer Service Request 1 Configuration Register T1C are met. A Timer Service Request 1 is also signalled by TINT1SRC. This flag is cleared by writing a 1. 0_B No Timer Service Request 1 1_B Timer Service Request 1 occurred
TIBC	10	rwh	**Transfer Input Buffer Completed** This flag is set whenever a transfer from Input Buffer to the Message RAM has completed and bit IBCR.IBSYS in the Input Buffer Command Request register has been reset by the Message Handler. This flag is cleared by writing a 1. 0_B No transfer completed 1_B Transfer between Input Buffer and Message RAM completed
TOBC	11	rwh	**Transfer Output Buffer Completed** This flag is set whenever a transfer from Message RAM to the Output Buffer has completed and bit OBCR.OBSYS in the Output Buffer Command Request register has been reset by the Message Handler. This flag is cleared by writing a 1. 0_B No transfer completed 1_B Transfer between Message RAM and the Output Buffer completed

Field	Bits	Type	Description
SWE	12	rwh	**Stop Watch Event** This flag is set after a stop watch activation when the current cycle counter and Macrotick value are stored in the Stop Watch Register 1 (STPW1). This flag is cleared by writing a 1. 0_B No Stop Watch Event 1_B Stop Watch Event occurred
SUCS	13	rwh	**Startup Completed Successfully** This flag is set whenever a startup completed successfully and the Communication Controller entered “NORMAL_ACTIVE” state. This flag is cleared by writing a 1. 0_B No startup completed successfully 1_B Startup completed successfully
MBSI	14	rwh	**Message Buffer Status Service Request** This flag is set by the Communication Controller when the Message Buffer status MBS has changed and if bit RDHS1.MBI of that Message Buffer is set. This flag is cleared by writing a 1. 0_B No Message Buffer status change of Message Buffer with RDHS1.MBI= 1 has changed 1_B Message Buffer status of at least one Message Buffer with RDHS1.MBI= 1 has changed
SDS	15	rwh	**Start of Dynamic Segment** This flag is set by the Communication Controller when the dynamic segment starts. 0_B Dynamic segment not yet started 1_B Dynamic segment started
WUPA	16	rwh	**Wakeup Pattern Channel A** This flag is set by the Communication Controller when a wakeup pattern was received on channel A. Only set when the Communication Controller is in “WAKEUP”, “READY”, or “STARTUP” state, or when in Monitor mode. This flag is cleared by writing a 1. 0_B No wake-up pattern received on channel A 1_B Wake-up pattern received on channel A
MTSA	17	rwh	**MTS Received on Channel A(vSS!ValidMTSA)** Media Access Test symbol received on channel A during the proceeding symbol window. Updated by the Communication Controller for each channel at the end of the symbol window. This flag is cleared by writing a 1. 0_B No MTS symbol received on channel A 1_B MTS symbol received on channel A

Field	Bits	Type	Description
WUPB	24	rwh	**Wakeup Pattern Channel B** This flag is set by the Communication Controller when a wakeup pattern was received on channel B. Only set when the Communication Controller is in “WAKEUP”, “READY”, or “STARTUP” state, or when in Monitor mode. This flag is cleared by writing a 1. 0_B No wake-up pattern received on channel B 1_B Wake-up pattern received on channel B
MTSB	25	rwh	**MTS Received on Channel B** Media Access Test symbol received on channel B during the proceeding symbol window. Updated by the Communication Controller for each channel at the end of the symbol window. This flag is cleared by writing a 1. 0_B No MTS symbol received on channel B 1_B MTS symbol received on channel B
0	23:18, 31:26	r	**Reserved** Returns 0 if read; should be written with 0.

Error Service Request Line Select

The Error Service Request Line Select register assigns a service request generated by a specific error service request flag from register EIR to one of the two module service request lines INT0SRC or INT1SRC:

0 = Interrupt assigned to interrupt line (INT0SRC)

1 = Interrupt assigned to interrupt line (INT1SRC)

EILS

Error Service Request Line Select **(0028_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					TABBL	LTVBL	EDBL	0					TABAL	LTVAL	EDAL
r					rw	rw	rw	r					rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MHFL	IOBAL	IIBAL	EFAL	RFOL	EERRL	CCLL	CCFL	SFOL	SFBML	CNAL	PEMCL
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PEMCL	0	rw	**POC Error Mode Changed Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CNAL	1	rw	**Command Not Accepted Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
SFBML	2	rw	**SYNC Frames Below Minimum Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC

Field	Bits	Type	Description
SFOL	3	rw	**SYNC Frame Overflow Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CCFL	4	rw	**Clock Correction Failure Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CCLL	5	rw	**CHI Command Locked Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
EERRL	6	rw	**ECC Error Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
RFOL	7	rw	**Receive FIFO Overrun Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
EFAL	8	rw	**Empty FIFO Access Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
IIBAL	9	rw	**Illegal Input Buffer Access Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
IOBAL	10	rw	**Illegal Output Buffer Access Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
MHFL	11	rw	**Message Handler Constrains Flag Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
EDAL	16	rw	**Error Detected on Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
LTVAL	17	rw	**Latest Transmit Violation Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TABAL	18	rw	**Transmission Across Boundary Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
EDBL	24	rw	**Error Detected on Channel B Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
LTVBL	25	rw	**Latest Transmit Violation Channel B Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TABBL	26	rw	**Transmission Across Boundary Channel B Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC

Field	Bits	Type	Description
0	15:12, 23:19, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

Status Service Request Line Select

The Status Service Request Line Select register assign an service request generated by a specific status service request flag from register SIR to one of the two module service request lines INT0SRC or INT1SRC:

0 = Interrupt assigned to interrupt line INT0SRC

1 = Interrupt assigned to interrupt line INT1SRC

SILS

Status Service Request Line Select **(002C$_H$)** **Application Reset Value: 0303 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MTSBL	WUPBL	0						MTSAL	WUPAL
r						rw	rw	r						rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDSL	MBSIL	SUCSL	SWEL	TOBCL	TIBCL	TI1L	TI0L	NMVCL	RFCLL	RFNEL	RXIL	TXIL	CYCSL	CASL	WSTL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
WSTL	0	rw	**Wakeup Status Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CASL	1	rw	**Collision Avoidance Symbol Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
CYCSL	2	rw	**Cycle Start Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TXIL	3	rw	**Transmit Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
RXIL	4	rw	**Receive Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
RFNEL	5	rw	**Receive FIFO Not Empty Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
RFCLL	6	rw	**Receive FIFO Critical Level Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC

Field	Bits	Type	Description
NMVCL	7	rw	**Network Management Vector Changed Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TI0L	8	rw	**Timer Service Request 0 Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TI1L	9	rw	**Timer Service Request 1 Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TIBCL	10	rw	**Transfer Input Buffer Completed Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
TOBCL	11	rw	**Transfer Output Buffer Completed Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
SWEL	12	rw	**Stop Watch Event Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
SUCSL	13	rw	**Startup Completed Successfully Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
MBSIL	14	rw	**Message Buffer Status Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
SDSL	15	rw	**Start of Dynamic Segment Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
WUPAL	16	rw	**Wakeup Pattern Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
MTSAL	17	rw	**Media Access Test Symbol Channel A Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
WUPBL	24	rw	**Wakeup Pattern Channel B Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
MTSBL	25	rw	**Media Access Test Symbol Channel B Service Request Line** 0_B Service Request assigned to service request line INT0SRC 1_B Service Request assigned to service request line INT1SRC
0	23:18, 31:26	r	**Reserved** Returns 0 if read; should be written with 0.

Error Service Request Enable Set

The settings in the Error Service Request Enable register determine which status changes in the Error Service Request Register will result in an service request. The enable bits are set by writing to EIES and reset by writing to EIER. Writing a 1 sets the specific enable bit, a 0 has no effect.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register.

EIES
Error Service Request Enable Set (0030$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					TABBE	LTVBE	EDBE	0					TABAE	LTVAE	EDAE
r					rwh	rwh	rwh	r					rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MHFE	IOBAE	IIBAE	EFAE	RFOE	EERRE	CCLE	CCFE	SFOE	SFBME	CNAE	PEMCE
r				rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
PEMCE	0	rwh	**POC Error Mode Changed Service Request Enable** 0_B Read: Protocol Error Mode Changed Service Request disabled Write: Unchanged 1_B Read: Protocol Error Mode Changed Service Request enabled Write: Enable Protocol Error Mode Changed Service Request
CNAE	1	rwh	**Command Not Accepted Service Request Enable** 0_B Read: Command Not Valid Service Request disabled Write: Unchanged 1_B Read: Command Not Valid Service Request enabled Write: Enable Command Not Valid Service Request
SFBME	2	rwh	**SYNC Frames Below Minimum Service Request Enable** 0_B Read: SYNC Frames Below Minimum Service Request disabled Write: Unchanged 1_B Read: SYNC Frames Below Minimum Service Request enabled Write: Enable SYNC Frames Below Minimum Service Request
SFOE	3	rwh	**SYNC Frame Overflow Service Request Enable** 0_B Read: SYNC Frame Overflow Service Request disabled Write: Unchanged 1_B Read: SYNC Frame Overflow Service Request enabled Write: Enable Protocol Error Mode Changed Service Request
CCFE	4	rwh	**Clock Correction Failure Service Request Enable** 0_B Read: Clock Correction Failure Service Request disabled Write: Unchanged 1_B Read: Clock Correction Failure Service Request enabled Write: Enable Clock Correction Failure Service Request

Field	Bits	Type	Description
CCLE	5	rwh	**CHI Command Locked Service Request Enable** 0_B Read: CHI Command Locked Service Request disabled Write: Unchanged 1_B Read: CHI Command Locked Service Request enabled Write: Enable CHI Command Locked Service Request
EERRE	6	rwh	**ECC Error Service Request Enable** 0_B Read: ECC Error Service Request disabled Write: Unchanged 1_B Read: ECC Error Service Request enabled Enable ECC Error Service Request Write: Unchanged
RFOE	7	rwh	**Receive FIFO Overrun Service Request Enable** 0_B Read: Receive FIFO Overrun Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Overrun Service Request enabled Write: Enable Receive FIFO Overrun Service Request
EFAE	8	rwh	**Empty FIFO Access Service Request Enable** 0_B Read: Empty FIFO Access Service Request disabled Write: Unchanged 1_B Read: Empty FIFO Access Service Request enabled Write: Enable Empty FIFO Access Service Request
IIBAE	9	rwh	**Illegal Input Buffer Access Service Request Enable** 0_B Read: Illegal Input Buffer Access Service Request disabled Write: Unchanged 1_B Read: Illegal Input Buffer Access Service Request enabled Write: Enable Illegal Input Buffer Access Service Request
IOBAE	10	rwh	**Illegal Output Buffer Access Service Request Enable** 0_B Read: Illegal Output Buffer Access Service Request disabled Write: Unchanged 1_B Read: Illegal Output Buffer Access Service Request enabled Write: Enable Illegal Output Buffer Access Service Request
MHFE	11	rwh	**Message Handler Constraints Flag Service Request Enable** 0_B Read: Message Handler Constraints Flag Service Request disabled Write: Unchanged 1_B Read: Message Handler Constraints Flag Service Request enabled Write: Enable Message Handler Constraints Flag Service Request
EDAE	16	rwh	**Error Detected on Channel A Service Request Enable** 0_B Read: Error Detected on Channel A Service Request disabled Write: Unchanged 1_B Read: Error Detected on Channel A Service Request enabled Write: Enable Error Detected on Channel A Service Request
LTVAE	17	rwh	**Latest Transmit Violation Channel A Service Request Enable** 0_B Read: Latest Transmit Violation Channel A Service Request disabled Write: Unchanged 1_B Read: Latest Transmit Violation Channel A Service Request enabled Write: Enable Latest Transmit Violation Channel A Service Request

Field	Bits	Type	Description
TABAE	18	rwh	**Transmission Across Boundary Channel A Service Request Enable** 0_B Read: Transmission Across Boundary Channel A Service Request disabled Write: Unchanged 1_B Read: Transmission Across Boundary Channel A Service Request enabled Write: Enable Transmission Across Boundary Channel A Service Request
EDBE	24	rwh	**Error Detected on Channel B Service Request Enable** 0_B Read: Error Detected on Channel B Service Request disabled Write: Unchanged 1_B Read: Error Detected on Channel B Service Request enabled Write: Enable Error Detected on Channel B Service Request
LTVBE	25	rwh	**Latest Transmit Violation Channel B Service Request Enable** 0_B Read: Latest Transmit Violation Channel B Service Request disabled Write: Unchanged 1_B Read: Latest Transmit Violation Channel B Service Request enabled Write: Enable Latest Transmit Violation Channel B Service Request
TABBE	26	rwh	**Transmission Across Boundary Channel B Service Request Enable** 0_B Read: Transmission Across Boundary Channel B Service Request disabled Write: Enable Transmission Across Boundary Channel B Service Request 1_B Read: Transmission Across Boundary Channel B Service Request enabled Write: Enable Transmission Across Boundary Channel B Service Request
0	15:12, 23:19, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

Error Service Request Enable Reset

The settings in the Error Service Request Enable register determine which status changes in the Error Service Request Register will result in a service request. The enable bits are set by writing to EIES and reset by writing to EIER. Writing a 1 resets the specific enable bit, a 0 has no effect.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register.

EIER

Error Service Request Enable Reset (0034_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					TABBE	LTVBE	EDBE	0					TABAE	LTVAE	EDAE
r					rwh	rwh	rwh			r			rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				MHFE	IOBAE	IIBAE	EFAE	RFOE	EERRE	CCLE	CCFE	SFOE	SFBME	CNAE	PEMCE
r				rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
PEMCE	0	rwh	**POC Error Mode Changed Service Request Enable** 0_B Read: Protocol Error Mode Changed Service Request disabled Write: Unchanged 1_B Read: Protocol Error Mode Changed Service Request enabled Write: Disable Protocol Error Mode Changed Service Request
CNAE	1	rwh	**Command Not Accepted Service Request Enable** 0_B Read: Command Not Accepted Service Request disabled Write: Unchanged 1_B Read: Command Not Accepted Service Request enabled Write: Disable Command Not Accepted Service Request
SFBME	2	rwh	**SYNC Frames Below Minimum Service Request Enable** 0_B Read: SYNC Frames Below Minimum Service Request disabled Write: Unchanged 1_B Read: SYNC Frames Below Minimum Service Request enabled Write: Disable SYNC Frames Below Minimum Service Request
SFOE	3	rwh	**SYNC Frame Overflow Service Request Enable** 0_B Read: SYNC Frame Overflow Service Request disabled Write: Unchanged 1_B Read: SYNC Frame Overflow Service Request enabled Write: Disable Protocol Error Mode Changed Service Request
CCFE	4	rwh	**Clock Correction Failure Service Request Enable** 0_B Read: Clock Correction Failure Service Request disabled Write: Unchanged 1_B Read: Clock Correction Failure Service Request enabled Write: Disable Clock Correction Failure Service Request
CCLE	5	rwh	**CHI Command Locked Service Request Enable** 0_B Read: CHI Command Locked Service Request disabled Write: Unchanged 1_B Read: CHI Command Locked Service Request enabled Write: Disable CHI Command Locked Service Request

Field	Bits	Type	Description
EERRE	6	rwh	**ECC Error Service Request Enable** 0_B Read: ECC Error Service Request disabled Write: Unchanged 1_B ERead: CC Error Service Request enabled Write: Disable ECC Error Service Request
RFOE	7	rwh	**Receive FIFO Overrun Service Request Enable** 0_B Read: Receive FIFO Overrun Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Overrun Service Request enabled Write: Disable Receive FIFO Overrun Service Request
EFAE	8	rwh	**Empty FIFO Access Service Request Enable** 0_B Read: Empty FIFO Access Service Request disabled Write: Unchanged 1_B Read: Empty FIFO Access Service Request enabled Write: Disable Empty FIFO Access Service Request
IIBAE	9	rwh	**Illegal Input Buffer Access Service Request Enable** 0_B Read: Illegal Input Buffer Access Service Request disabled Write: Unchanged 1_B Read: Illegal Input Buffer Access Service Request enabled Write: Disable Illegal Input Buffer Access Service Request
IOBAE	10	rwh	**Illegal Output Buffer Access Service Request Enable** 0_B Read: Illegal Output Buffer Access Service Request disabled Write: Unchanged 1_B Read: Illegal Output Buffer Access Service Request enabled Write: Disable Illegal Output Buffer Access Service Request
MHFE	11	rwh	**Message Handler Constraints Flag Service Request Enable** 0_B Read: Message Handler Constraints Flag Service Request disabled Write: Unchanged 1_B Read: Message Handler Constraints Flag Service Request enabled Write: Disable Message Handler Constraints Flag Service Request
EDAE	16	rwh	**Error Detected on Channel A Service Request Enable** 0_B Read: Error Detected on Channel A Service Request disabled Write: Unchanged 1_B Read: Error Detected on Channel A Service Request enabled Write: Disable Error Detected on Channel A Service Request
LTVAE	17	rwh	**Latest Transmit Violation Channel A Service Request Enable** 0_B Read: Latest Transmit Violation Channel A Service Request disabled Write: Unchanged 1_B Read: Latest Transmit Violation Channel A Service Request enabled Write: Disable Latest Transmit Violation Channel A Service Request
TABAE	18	rwh	**Transmission Across Boundary Channel A Service Request Enable** 0_B Read: Transmission Across Boundary Channel A Service Request disabled Write: Unchanged 1_B Read: Transmission Across Boundary Channel A Service Request enabled Write: Enable Transmission Across Boundary Channel A Service Request

Field	Bits	Type	Description
EDBE	24	rwh	**Error Detected on Channel B Service Request Enable** 0_B Read: Error Detected on Channel B Service Request disabled Write: Unchange 1_B Read: Error Detected on Channel B Service Request enabled Write: Disable Error Detected on Channel B Service Request
LTVBE	25	rwh	**Latest Transmit Violation Channel B Service Request Enable** 0_B Read: Latest Transmit Violation Channel B Service Request disabled Write: Unchanged 1_B Read: Latest Transmit Violation Channel B Service Request enabled Write: Disable Latest Transmit Violation Channel B Service Request
TABBE	26	rwh	**Transmission Across Boundary Channel B Service Request Enable** 0_B Read: Transmission Across Boundary Channel B Service Request disabled Write: Unchanged 1_B Read: Transmission Across Boundary Channel B Service Request enabled Write: Disable Transmission Across Boundary Channel B Service Request
0	15:12, 23:19, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

Status Service Request Enable Set

The settings in the Status Service Request Enable Set register determine which status changes in the Status Service Request Register will result in a service request. The enable bits are set by writing to SIES and reset by writing to SIER. Writing a 1 sets the specific enable bit, a 0 has no effect.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register.

SIES

Status Service Request Enable Set (0038$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MTSBE	WUPBE	0						MTSAE	WUPAE
r						rwh	rwh	r						rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDSE	MBSIE	SUCSE	SWEE	TOBCE	TIBCE	TI1E	TI0E	NMVCE	RFCLE	RFNEE	RXIE	TXIE	CYCSE	CASE	WSTE
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
WSTE	0	rwh	**Wakeup Status Service Request Enable** 0_B Read: Wake-up Status Service Request disabled Write: Unchanged 1_B Read:Wake-up Status Service Request enabled Write: Enable Wakeup Status Service Request
CASE	1	rwh	**Collision Avoidance Symbol Service Request Enable** 0_B Read: Collision Avoidance Symbol Service Request disabled Write: Unchanged 1_B Read: Collision Avoidance Symbol Service Request enabled Write: Enable Collision Avoidance Symbol Service Request
CYCSE	2	rwh	**Cycle Start Service Request Enable** 0_B Read: Cycle Start Service Request disabled Write: Unchanged 1_B Read: Cycle Start Service Request enabled Write: Enable Cycle Start Service Request
TXIE	3	rwh	**Transmit Service Request Enable** 0_B Read: Transmit Service Request disabled Write: Unchanged 1_B Transmit Service Request enabled Write: Enable Transmit Service Request
RXIE	4	rwh	**Receive Service Request Enable** 0_B Read: Receive Service Request disabled Write: Unchanged 1_B Read: Receive Service Request enabled Write: Enable Receive Service Request
RFNEE	5	rwh	**Receive FIFO Not Empty Service Request Enable** 0_B Read: Receive FIFO Not Empty Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Not Empty Service Request enabled Write: Enable Receive FIFO Not Empty Service Request
RFCLE	6	rwh	**Receive FIFO Critical Level Service Request Enable** 0_B Read: Receive FIFO Critical Level Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Critical Level Service Request enabled Write: Enable Receive FIFO Critical Level Service Request
NMVCE	7	rwh	**Network Management Vector Changed Service Request Enable** 0_B Read: Network Management Vector Changed Service Request disabled Write: Unchanged 1_B Read: Network Management Vector Changed Service Request enabled Write: Enable Network Management Vector Changed Service Request
TI0E	8	rwh	**Timer Service Request 0 Enable** 0_B Read: Timer Service Request 0disabled Write: Unchanged 1_B Read: Timer Service Request 0 enabled Write: Enable Timer Service Request 0
TI1E	9	rwh	**Timer Service Request 1 Enable** 0_B Read: Timer Service Request 1disabled Write: Unchanged 1_B Read: Timer Service Request 1 enabled Write: Enable Timer Service Request 1

Field	Bits	Type	Description
TIBCE	10	rwh	**Transfer Input Buffer Completed Service Request Enable** 0_B Read: Wakeup Status Service Request disabled Write: Unchanged 1_B Read: Wakeup Status Service Request enabled Write: Enable Wakeup Status Service Request
TOBCE	11	rwh	**Transfer Output Buffer Completed Service Request Enable** 0_B Read: Transfer Input Buffer Completed Service Request disabled Write: Unchanged 1_B Read: Transfer Input Buffer Completed Service Request enabled Write: Enable Transfer Input Buffer Completed Service Request
SWEE	12	rwh	**Stop Watch Event Service Request Enable** 0_B Read: Stop Watch Event Service Request disabled Write: Unchanged 1_B Read: Stop Watch Event Service Request enabled Write: Enable Stop Watch Event Service Request
SUCSE	13	rwh	**Startup Completed Successfully Service Request Enable** 0_B Read: Startup Completed Successfully Service Request disabled Write: Unchanged 1_B Read: Startup Completed Successfully Service Request enabled Write: Enable Startup Completed Successfully Service Request
MBSIE	14	rwh	**Message Buffer Status Service Request Enable** 0_B Read: Message Buffer Status Service Request disabled Write: Unchanged 1_B Read: Message Buffer Status Service Request enabled Write: Enable Message Buffer Status Service Request
SDSE	15	rwh	**Start of Dynamic Segment Service Request Enable** 0_B Read: Start of Dynamic Service Request disabled Write: Unchanged 1_B Read: Start of Dynamic Service Request enabled Write: Enable Start of Dynamic Service Request
WUPAE	16	rwh	**Wakeup Pattern Channel A Service Request Enable** 0_B Read: Wakeup Pattern Channel A Service Request disabled Write: Unchanged 1_B Read: Wakeup Pattern Channel A Service Request enabled Write: Enable Wakeup Pattern Channel A Service Request
MTSAE	17	rwh	**Media Access Test Symbol Channel A Service Request Enable** 0_B Read: Media Access Test Symbol Channel A Service Request disabled Write: Unchanged 1_B Read: Media Access Test Symbol Channel A Service Request enabled Write: Enable Media Access Test Symbol Channel A Service Request
WUPBE	24	rwh	**Wakeup Pattern Channel B Service Request Enable** 0_B Read: Wakeup Pattern Channel B Service Request disabled Write: Unchanged 1_B Read: Wakeup Pattern Channel B Service Request enabled Write: Enable Wakeup Pattern Channel B Service Request

Field	Bits	Type	Description
MTSBE	25	rwh	**Media Access Test Symbol Channel B Service Request Enable** 0_B Read: Media Access Test Symbol Channel B Service Request disabled Write: Unchanged 1_B Read: Media Access Test Symbol Channel B Service Request enabled Write: Enable Media Access Test Symbol Channel B Service Request
0	23:18, 31:26	r	**Reserved** Returns 0 if read; should be written with 0.

Status Service Request Enable Reset

The settings in the Status Service Request Enable Reset register determine which status changes in the Status Service Request Register will result in a service request. The enable bits are set by writing to SIES and reset by writing to SIER. Writing a 1 resets the specific enable bit, a 0 has no effect.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register.

SIER
Status Service Request Enable Reset (003C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MTSBE	WUPBE	0						MTSAE	WUPAE
r						rwh	rwh	r						rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SDSE	MBSIE	SUCSE	SWEE	TOBCE	TIBCE	TI1E	TI0E	NMVCE	RFCLE	RFNEE	RXIE	TXIE	CYCSE	CASE	WSTE
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
WSTE	0	rwh	**Wakeup Status Service Request Enable** 0_B Read: Wakeup Status Service Request disabled Write: Unchanged 1_B Read: Wakeup Status Service Request enabled Write: Disable Wakeup Status Service Request
CASE	1	rwh	**Collision Avoidance Symbol Service Request Enable** 0_B Read: Collision Avoidance Symbol Service Request disabled Write: Unchanged 1_B Read: Collision Avoidance Symbol Service Request enabled Write: Disable Collision Avoidance Symbol Service Request
CYCSE	2	rwh	**Cycle Start Service Request Enable** 0_B Read: Cycle Start Service Request disabled Write: Unchanged 1_B Read: Cycle Start Service Request enabled Write: Disable Cycle Start Service Request

Field	Bits	Type	Description
TXIE	3	rwh	**Transmit Service Request Enable** 0_B Read: Transmit Service Request disabled Write: Unchanged 1_B Read: Transmit Service Request enabled Write: Disable Transmit Service Request
RXIE	4	rwh	**Receive Service Request Enable** 0_B Read: Receive Service Request disabled Write: Unchanged 1_B Read: Receive Service Request enabled Write: Disable Receive Service Request
RFNEE	5	rwh	**Receive FIFO Not Empty Service Request Enable** 0_B Read: Receive FIFO Not Empty Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Not Empty Service Request enabled Write: Disable Receive FIFO Not Empty Service Request
RFCLE	6	rwh	**Receive FIFO Critical Level Service Request Enable** 0_B Read: Service Request disabled Write: Unchanged 1_B Read: Receive FIFO Critical Level Service Request enabled Write: Disable Receive FIFO Critical Level Service Request
NMVCE	7	rwh	**Network Management Vector Changed Service Request Enable** 0_B Read: Network Management Vector Changed Service Request disabled Write: Unchanged 1_B Read: Network Management Vector Changed Service Request enabled Write: Disable Network Management Vector Changed Service Request
TI0E	8	rwh	**Timer Service Request 0 Enable** 0_B Read: Timer Service Request 0 disabled Write: Unchanged 1_B Read: Timer Service Request 0 enabled Write: Disable Service Request 0
TI1E	9	rwh	**Timer Service Request 1 Enable** 0_B Read: Timer Service Request 1 disabled Write: Unchanged 1_B Read: Timer Service Request 1 enabled Write: Disable Timer Service Request 1
TIBCE	10	rwh	**Transfer Input Buffer Completed Service Request Enable** 0_B Read: Wakeup Status Service Request disabled Write: Unchanged 1_B Read: Wakeup Status Service Request enabled Write: Disable Wakeup Status Service Request
TOBCE	11	rwh	**Transfer Output Buffer Completed Service Request Enable** 0_B Read: Transfer Input Buffer Completed Service Request disabled Write: Unchanged 1_B Read: Transfer Input Buffer Completed Service Request enabled Write: Disable Transfer Input Buffer Completed Service Request
SWEE	12	rwh	**Stop Watch Event Service Request Enable** 0_B Read: Stop Watch Event Service Request disabled Write: Unchanged 1_B Read: Stop Watch Event Service Request enabled Write: Disable Stop Watch Event Service Request

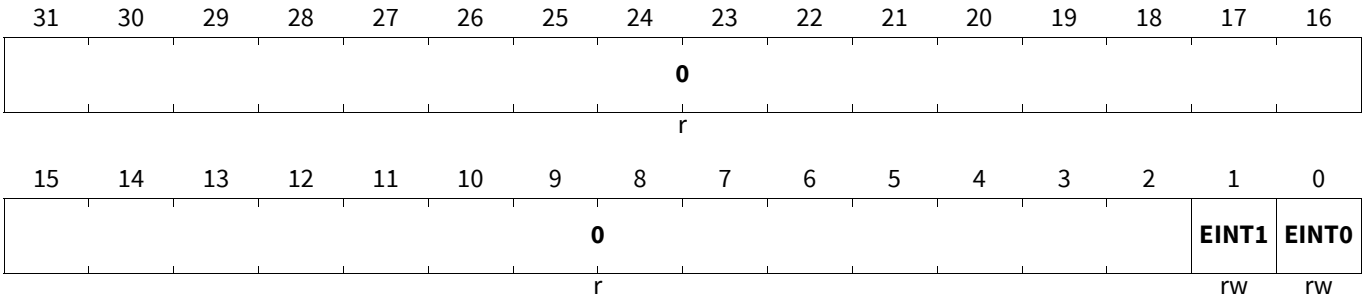
Field	Bits	Type	Description
SUCSE	13	rwh	**Startup Completed Successfully Service Request Enable** 0_B Read: Startup Completed Successfully Service Request disabled Write: Unchanged 1_B Read: Startup Completed Successfully Service Request enabled Write: Disable Startup Completed Successfully Service Request
MBSIE	14	rwh	**Message Buffer Status Service Request Enable** 0_B Read: Message Buffer Status Service Request disabled Write: Unchanged 1_B Read: Message Buffer Status Service Request enabled Write: Disable Message Buffer Status Service Request
SDSE	15	rwh	**Start of Dynamic Segment Service Request Enable** 0_B Read: Start of Dynamic Service Request disabled Write: Unchanged 1_B Read: Start of Dynamic Service Request enabled Write: Disable Start of Dynamic Service Reques
WUPAE	16	rwh	**Wakeup Pattern Channel A Service Request Enable** 0_B Read: Wakeup Pattern Channel A Service Request disabled Write: Unchanged 1_B Read: Wakeup Pattern Channel A Service Request enabled Write: Disable Wakeup Pattern Channel A Service Request
MTSAE	17	rwh	**Media Access Test Symbol Channel A Service Request Enable** 0_B Read: Media Access Test Symbol Channel A Service Request disabled Write: Unchanged 1_B Read: Media Access Test Symbol Channel A Service Request enabled Write: Disable Media Access Test Symbol Channel A Service Request
WUPBE	24	rwh	**Wakeup Pattern Channel B Service Request Enable** 0_B Read: Wakeup Pattern Channel B Service Request disabled Write: Unchanged 1_B Read: Wakeup Pattern Channel B Service Request enabled Write: Disable Wakeup Pattern Channel B Service Request
MTSBE	25	rwh	**Media Access Test Symbol Channel B Service Request Enable** 0_B Read: Media Access Test Symbol Channel B Service Request disabled Write: Unchanged 1_B Read: Media Access Test Symbol Channel B Service Request enabled Write: Disable Media Access Test Symbol Channel B Service Request
0	23:18, 31:26	r	**Reserved** Returns 0 if read; should be written with 0.

Service Request Line Enable

Each of the two service request lines to the Host INT0SRC, INT1SRC can be enabled / disabled separately by programming bit EINT0 and EINT1.

ILE

Service Request Line Enable **(0040_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
EINT0	0	rw	**Enable Service Request Line 0(INT0SRC)** 0_B Service Request line disabled 1_B Service Request line enabled
EINT1	1	rw	**Enable Service Request Line 1(INT1SRC)** 0_B Service Request line disabled 1_B Service Request line enabled
0	31:2	r	**Reserved** Returns 0 if read; should be written with 0.

Timer 0 Configuration

Absolute timer. Specifies in terms of cycle count and Macrotick the point in time when the timer 0 service request occurs. When the timer 0 service request is asserted, output signal TINT0SR is set to 1 for the duration of one Macrotick and SIR.TI0 is set to 1.

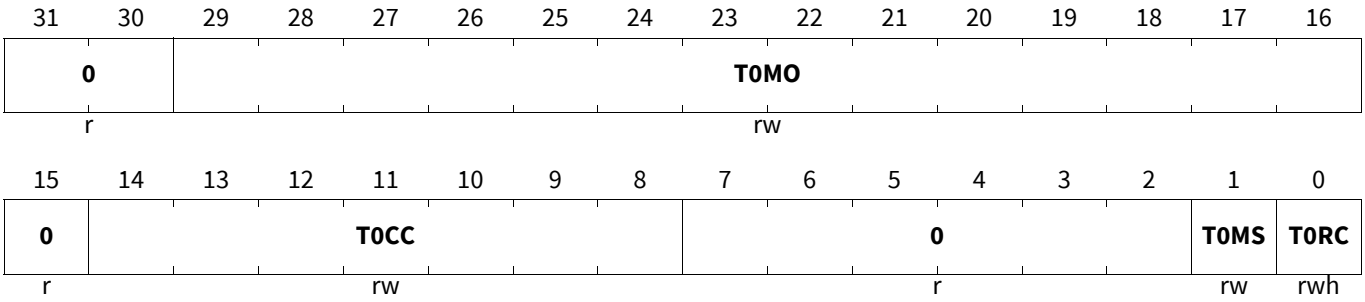
Timer 0 can be activated as long as the POC is either in “NORMAL_ACTIVE” state or in “NORMAL_PASSIVE” state. Timer 0 is deactivated when leaving “NORMAL_ACTIVE” state or “NORMAL_PASSIVE” state except for transitions between the two states.

Before reconfiguration of the timer, the timer has to be halted first by writing 0 to bit T0RC.

Note: The configuration of timer 0 is compared against the Macrotick counter value, there is no separate counter for timer 0. In case the Communication Controller leaves “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state, or if timer 0 is halted by Host command, output signal TINT0SR is reset to 0 immediately.

T0C

Timer 0 Configuration **(0044_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
T0RC	0	rwh	**Timer 0 Run Control** 0_B Timer 0 halted 1_B Timer 0 running
T0MS	1	rw	**Timer 0 Mode Select** 0_B Single-shot mode 1_B Continuous mode
T0CC	14:8	rw	**Timer 0 Cycle Code** The 7-bit timer 0 cycle code determines the cycle set used for generation of the timer 0 service request. For details about the configuration of the cycle code see “Cycle Counter Filtering”.
T0MO	29:16	rw	**Timer 0 Macrotick Offset** Configures the Macrotick offset from the beginning of the cycle where the service request is to occur. The Timer 0 Service Request occurs at this offset for each cycle of the cycle set.
0	7:2, 15, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

Timer 1 Configuration

Relative timer. After the specified number of Macroticks has expired, the timer 1 service request is asserted, output signal TINT1SR is set to 1 for the duration of one Macrotick and SIR.TI1 is set to 1.

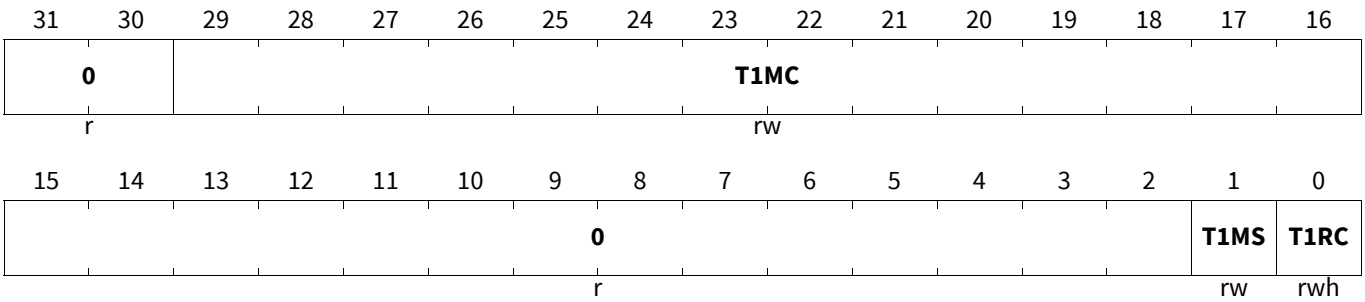
Timer 1 can be activated as long as the POC is either in “NORMAL_ACTIVE” state or in “NORMAL_PASSIVE” state. Timer 1 is deactivated when leaving “NORMAL_ACTIVE” state or “NORMAL_PASSIVE” state except for transitions between the two states.

Before reconfiguration of the timer, the timer has to be halted first by resetting bit T1RC to 0.

Note: In case the Communication Controller leaves “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state, or if timer 1 is halted by Host command, output signal TINT1SR is reset to 0 immediately.

T1C

Timer 1 Configuration (0048$_H$) Application Reset Value: 0002 0000$_H$



Field	Bits	Type	Description
T1RC	0	rwh	**Timer 1 Run Control** 0_B Timer 1 halted 1_B Timer 1 running

Field	Bits	Type	Description
T1MS	1	rw	**Timer 1 Mode Select** 0_B Single-shot mode 1_B Continuous mode
T1MC	29:16	rw	**Timer 1 Macrotick Count** When the configured Macrotick count is reached the timer 1 service request is generated. Valid values are: 2_H …$3FFF_H$ Macroticks in continuous mode 1_H …$3FFF_H$ Macroticks in single-shot mode
0	15:2, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

Stop Watch Register 1

The stop watch is activated by a rising or falling edge on signal STPW, by a service request 0 or 1 event (rising edge on signal INT0SR or INT1SR) or by the Host by writing bit STPW1.SSWT to 1. With the Macrotick counter increment following next to the stop watch activation the actual cycle counter and Macrotick value are captured in the Stop Watch Register 1 STPW1 while the slot counter values for channel A and B are captured in the Stop Watch Register 2 STPW2.

Note: Bits ESWT and SSWT cannot be set to 1 simultaneously. In this case the write access is ignored, and both bits keep their previous values. Therefore either the external stop watch triggers or the software stop watch trigger may be used.

STPW1

Stop Watch Register 1 **($004C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**SMTV**													
r		rh													

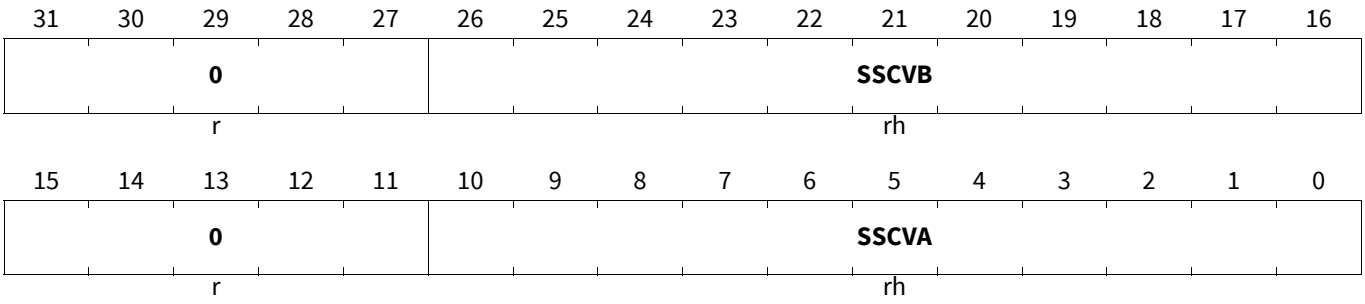
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**SCCV**						**0**	**EINT1**	**EINT0**	**EETP**	**SSWT**	**EDGE**	**SWMS**	**ESWT**
r		rh						r	rw	rw	rw	rwh	rw	rw	rwh

Field	Bits	Type	Description
ESWT	0	rwh	**Enable Stop Watch Trigger** If enabled an edge on input STPW or a service request 0 or 1 event (rising edge on signal INT0SR or INT1SR) activates the stop watch. In single-shot mode this bit is reset to 0 after the actual cycle counter and Macrotick value are stored in the Stop Watch register. 0_B Stop watch trigger disabled 1_B Stop watch trigger enabled

Field	Bits	Type	Description
SWMS	1	rw	**Stop Watch Mode Select** It is not possible to change the Stop Watch Mode during enabled stop watch trigger (STPW1.ESWT) 0_B Single-shot mode 1_B Continuous mode
EDGE	2	rw	**Stop Watch Trigger Edge Select** 0_B Falling Edge 1_B Rising Edge
SSWT	3	rwh	**Software Stop Watch Trigger** When the Host writes this bit to 1 the stop watch is activated. After the actual cycle counter and Macrotick value are stored in the Stop Watch register this bit is reset to 0. The bit is only writeable while ESWT = 0. 0_B Software trigger reset 1_B Stop watch activated by software trigger
EETP	4	rw	**Enable External Trigger Pin** Enables stop watch trigger event via signal STPW if ESWT = 1. 0_B Stop watch trigger via signal STPW disabled 1_B Edge on signal STPW triggers stop watch
EINT0	5	rw	**Enable Service Request 0 Trigger** Enables stop watch trigger by service request 0 event if ESWT = 1. 0_B Stop watch trigger by service request 0 disabled 1_B Service Request 0 event triggers stop watch
EINT1	6	rw	**Enable Service Request 1 Trigger** Enables stop watch trigger by service request 1 event if ESWT = 1. 0_B Stop watch trigger by service request 1 disabled 1_B Service Request 1 event triggers stop watch
SCCV	13:8	rh	**Stopped Cycle Counter Value** State of the cycle counter when the stop watch event occurred. Valid values are: $0…3F_H$ Valid Values
SMTV	29:16	rh	**Stopped Macrotick Value** State of the Macrotick counter when the stop watch event occurred. Valid values are: $0…3F_H$ Valid Values
0	7, 15:14, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

Stop Watch Register 2

STPW2
Stop Watch Register 2 **(0050_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
SSCVA	10:0	rh	**Stop Watch Captured Slot Counter Value Channel A** State of the slot counter for channel A when the stop watch event occurred. Valid values are 0 to 2047 (0_H to $7FF_H$).
SSCVB	26:16	rh	**Stop Watch Captured Slot Counter Value Channel B** State of the slot counter for channel B when the stop watch event occurred. Valid values are 0 to 2047 (0_H to $7FF_H$).
0	15:11, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

41.4.2.4 Communication Controller Control Registers

This section describes the registers provided by the Communication Controller to allow the Host to control the operation of the Communication Controller. The FlexRay™ protocol specification requires the Host to write application configuration data in “CONFIG” state only. Please consider that the configuration registers are not locked for writing in “DEFAULT_CONFIG” state.

The configuration data is reset when “DEFAULT_CONFIG” state is entered from application reset. To change POC state from “DEFAULT_CONFIG” to “CONFIG” state the Host has to apply CHI command “CONFIG”. If the Host wants the Communication Controller to leave “CONFIG” state, the Host has to proceed as described on **LCK** Register.

SUC Configuration Register 1

Note:

3. *This bit can be updated in “DEFAULT_CONFIG” or “CONFIG” state only!*
4. *The protocol requires that both bits TXST and TXSY are set for coldstart nodes.*
5. *MTSA and MTSB may also be changed outside “DEFAULT_CONFIG” or “CONFIG” state when the write to SUCC1 register is directly preceded by the unlock sequence as described in Lock Register (LCK). This may be combined with CHI command “SEND_MTS”. If both bits MTSA and MTSB are set to 1 an MTS symbol will be transmitted on both channels when requested by writing SUCC1.CMD = 1000B.*

SUCC1

SUC Configuration Register 1 **(0080_H)** **Application Reset Value: $0C40\ 1000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				CCHB	CCHA	MTSB	MTSA	HCSE	TSM	WUCS	PTA				
r				rw	rw	rw	rw	rw	rw	rw	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSA					0	TXSY	TXST	PBSY	0			CMD			
rw					r	rw	rw	rh	r			rwh			

Field	Bits	Type	Description
CMD	3:0	rwh	**CHI Command Vector** The host may write any CHI command at any time, but certain commands are only enabled in specific POC states. A disabled command will not be executed, the CHI command vector CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”, and flag EIR.CNA in the Error Service Request register will be set to 1. In case the previous CHI command has not yet completed, EIR.CCL is set to 1 together with EIR.CNA; the CHI command needs to be repeated. Except for HALT state, POC state change command applied while the Communication Controller is already in the requested POC state will be ignored. Reading SUCC1.CMD shows whether the last CHI command was accepted. CCSV.POCS monitors the actual POC state. The reserved CHI commands code hardware test functions. 0_H COMMAND_NOT_ACCEPTED” 1_H CONFIG 2_H READY 3_H WAKEUP 4_H RUN 5_H ALL_SLOTS 6_H HALT 7_H FREEZE 8_H SEND_MTS 9_H ALLOW_COLDSTART A_H RESET_STATUS_INDICATORS B_H MONITOR_MODE C_H CLEAR_RAMS D_H Reserved … F_H Reserved
PBSY	7	rh	**POC Busy** Signals that the POC is busy and cannot accept a command from the Host. SUCC1.CMD is locked against write accesses. Set to 1 after hard reset during initialization of internal RAM blocks. 0_B POC not busy, SUCC1.CMD writable 1_B POC is busy, SUCC1.CMD locked

Field	Bits	Type	Description
TXST	8	rw	**Transmit Startup Frame in Key Slot (pKeySlotUsedForStartup)** Defines whether the key slot is used to transmit startup Frames. The bit can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B No Startup Frame transmission in key slot, node is non-coldstarter 1_B Key slot used to transmit startup Frame, node is leading or following coldstarter
TXSY	9	rw	**Transmit SYNC Frame in Key Slot (pKeySlotUsedForSync)** Defines whether the key slot is used to transmit SYNC Frames. The bit can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. [1) 2)] 0_B No SYNC Frame transmission in key slot, node is neither sync nor coldstart node 1_B Key slot used to transmit SYNC Frames, node is sync node
CSA	15:11	rw	**Cold Start Attempts (gColdStartAttempts)** Configures the maximum number of attempts that a cold starting node is permitted to try to start up the network without receiving any valid response from another node. It can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. Must be identical in all nodes of a cluster. Valid values are 2 to 31.
PTA	20:16	rw	**Passive to Active (pAllowPassiveToActive)** Defines the number of consecutive even / odd cycle pairs that must have valid clock correction terms before the Communication Controller is allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. If set to 00000_B the Communication Controller is not allowed to transit from “NORMAL_PASSIVE” to “NORMAL_ACTIVE” state. It can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. Valid values are 0 to 31 even / odd cycle pairs.
WUCS	21	rw	**Wakeup Channel Select (pWakeupChannel)** With this bit the Host selects the channel on which the Communication Controller sends the Wakeup pattern. The Communication Controller ignores any attempt to change the status of this bit when not in “DEFAULT_CONFIG” or “CONFIG” state. 0_B Send wakeup pattern on channel A 1_B Send wakeup pattern on channel B

Field	Bits	Type	Description
TSM	22	rw	**Transmission Slot Mode (pSingleSlotEnabled)** Selects the initial transmission slot mode. In SINGLE slot mode the Communication Controller may only transmit in the preconfigured key slot. The key slot ID is configured in the Header Section of Message Buffer 0 respectively Message Buffers 0 and 1 depending on bit MRC.SPLM. In case SUCC1.TSM = 1, Message Buffer 0 respectively Message Buffers 0,1 can be (re)configured in “DEFAULT_CONFIG” or “CONFIG” state only. In ALL slot mode the Communication Controller may transmit in all slots. The bit can be written in “DEFAULT_CONFIG” or “CONFIG” state only. The communication controller changes to ALL slot mode when the Host successfully applied the ALL_SLOTS command by writing SUCC1.CMD = 0101_B in POC states “NORMAL_ACTIVE” or “NORMAL_PASSIVE”. The actual slot mode is monitored by CCSV.SLM. 0_B ALL Slot Mode 1_B SINGLE Slot Mode (default after application reset)
HCSE	23	rw	**Halt due to Clock Sync Error (pAllowHaltDueToClock)** Controls the transition to “HALT” state due to a clock synchronization error. The bit can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B Communication Controller will enter / remain in “NORMAL_PASSIVE” 1_B Communication Controller will enter “HALT” state
MTSA	24	rw	**Select Channel A for MTS Transmission** The bit selects channel A for MTS symbol transmission. The flag is reset by default and may be modified only in “DEFAULT_CONFIG” or “CONFIG” state. 0_B Channel A disabled for MTS transmission 1_B Channel A selected for MTS transmission
MTSB	25	rw	**Select Channel B for MTS Transmission** The bit selects channel B for MTS symbol transmission. The flag is reset by default and may be modified only in “DEFAULT_CONFIG” or “CONFIG” state. 0_B Channel B disabled for MTS transmission 1_B Channel B selected for MTS transmission
CCHA	26	rw	**Connected to Channel A (pChannels)** Configures whether the node is connected to channel A. 0_B Not connected to channel A 1_B Node connected to channel A (default after application reset)
CCHB	27	rw	**Connected to Channel B (pChannels)** Configures whether the node is connected to channel B. 0_B Not connected to channel B 1_B Node connected to channel B (default after application reset)
0	6:4, 10, 31:28	r	**Reserved** Returns 0 if read; should be written with 0.

COMMAND_NOT_ACCEPTED

SUCC1.CMD is reset to 0000_B due to one of the following conditions:

- Illegal command applied by the Host
- Host writes COMMAND_NOT_ACCEPTED
- Host applied new command while execution of the previous Host command has not completed

When SUCC1.CMD is reset to 0000_B, bit EIR.CNA in the Error Service Request register is set, and - if enabled - an service request is generated. Commands which are not accepted are not executed.

CONFIG

Go to POC state “CONFIG” when called in POC states “DEFAULT_CONFIG“, “READY“, or in “MONITOR_MODE“. When called in “HALT” state transits to POC state “DEFAULT_CONFIG“. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

READY

Go to POC state “READY” when called in POC states “CONFIG”, “NORMAL_ACTIVE”, “NORMAL_PASSIVE”, “STARTUP”, or “WAKEUP”. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

WAKEUP

Go to POC state WAKEUP when called in POC state “READY”. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

RUN

Go to POC state “STARTUP” when called in POC state “READY”. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

ALL_SLOTS

Leave SINGLE slot mode after successful startup / integration at the next end of cycle when called in POC states “NORMAL_ACTIVE” or “NORMAL_PASSIVE”. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

HALT

Set the halt request CCSV.HRQ bit in the Communication Controller Status Vector register and go to POC state “HALT” at the next end of cycle when called in POC states “NORMAL_ACTIVE” or “NORMAL_PASSIVE“. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED“.

FREEZE

Set the freeze status indicator CCSV.FSI and go to POC state “HALT” immediately. Can be called from any state.

SEND_MTS

Send single MTS symbol during the next following symbol window on the channel configured by SUCC1.MTSA, SUCC1.MTSB, when called in POC state “NORMAL_ACTIVE”. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

ALLOW_COLDSTART

The command resets bit CCSV.CSI to enable the node to become cold starter. When called in states “DEFAULT_CONFIG“, “CONFIG”, “HALT”, or “MONITOR_MODE“. SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. To become leading coldstarter it is also required that both TXST and TXSY are set.

RESET_STATUS_INDICATORS

Resets status flags CCSV.CSNI, CCSV.CSAI, CCSV.WSV to their default values. May be called in POC state READY. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”.

MONITOR_MODE

Enter MONITOR_MODE when called in POC state CONFIG. In this mode the Communication Controller is able to receive FlexRay™ Frames and wakeup pattern. It is also able to detect coding errors. The temporal integrity of received Frames is not checked. This mode can be used for debugging purposes, e.g. in case that the startup of a FlexRay™ network fails. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. For details see “MONITOR_MODE”.

CLEAR_RAMS

Sets bit MHDS.CRAM in the Message Handler Status register when called in “DEFAULT_CONFIG” or “CONFIG” state. When called in any other state, SUCC1.CMD will be reset to 0000_B = “COMMAND_NOT_ACCEPTED”. By setting MHDS.CRAM all internal RAM blocks are initialized to zero. Note that only the currently active IBF bank is cleared. To clear the 2nd bank as well, CUST1.IBF1PAG and CUST1.IBF2PAG need to be set and command CLEAR_RAMS need to be issued again. This is required in particular after an application reset. If the 2nd bank of IBF is left unused, this procedure is not required. During the initialization of the RAMs, SUCC1.PBSY will show POC busy. Access to the configuration and status registers is possible during execution of CHI command CLEAR_RAMS.

The initialization of the E-Ray internal RAM blocks requires 2048 f_{CLC_ERAY} cycles. There should be no Host access to IBF or OBF during initialization of the internal RAM blocks after application reset or after assertion of CHI command CLEAR_RAMS. Before asserting CHI command CLEAR_RAMS the Host should make sure that no transfer between Message RAM and IBF / OBF or the Transient Buffer RAMs is ongoing. This command also resets the Message Buffer Status registers MHDS, LDTS, FSR, MHDF, TXRQ1, TXRQ2, TXRQ3, TXRQ4, NDAT1, NDAT2, NDAT3, NDAT4, MBSC1, MBSC2, MBSC3, and MBSC4.

Note: All accepted commands with exception of CLEAR_RAMS and SEND_MTS will cause a change of register CCSV after at most 8 cycles of the slower of the two clocks f_{CLC_ERAY} and f_{SCLK}, assumed that POC was not busy when the command was applied and that no POC state change was forced by bus activity in that time Frame. Reading register CCSV will show data that is delayed by synchronization from f_{SCLK} to f_{CLC_ERAY} domain and by the Host-specific CPU interface.

Table below references the CHI commands from the FlexRay™ Protocol Specification v2.1 (section 2.2.1.1, Table 2-2) to the E-Ray CHI command vector CMD.

Table 418 Reference to CHI Host command summary from FlexRay™ protocol specification

CHI Command	Where processed (POC State)	CHI Command Vector CMD
ALL_SLOT	POC:NORMAL_ACTIVE, POC:NORMAL_PASSIVE	ALL_SLOTS
ALLOW_COLDSTART	All except POC:DEFAULT_CONFIG, POC:CONFIG, POC:HALT	ALLOW_COLDSTART

Table 418 Reference to CHI Host command summary from FlexRay™ protocol specification (cont'd)

CHI Command	Where processed (POC State)	CHI Command Vector CMD
CONFIG	POC:DEFAULT_CONFIG, POC:READY	CONFIG
CONFIG_COMPLETE	POC:CONFIG	Unlock sequence & READY
DEFAULT_CONFIG	POC:HALT	CONFIG
FREEZE	All	FREEZE
HALT	POC:NORMAL_ACTIVE, POC:NORMAL_PASSIVE	HALT
READY	All except POC:DEFAULT_CONFIG, POC:CONFIG, POC:READY, POC:HALT	READY
RUN	POC:READY	RUN
WAKEUP	POC:READY	WAKEUP

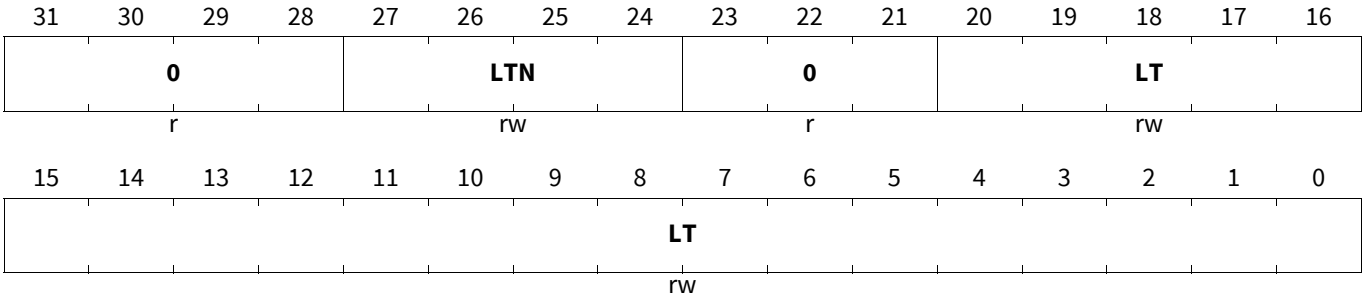
SUC Configuration Register 2

The Communication Controller accepts modifications of the register in "DEFAULT_CONFIG" or "CONFIG" state only.

Note: The wakeup / startup noise time-out is calculated as follows: The wakeup / startup noise time-out = pdListenTimeout • gListenNoise = LT • (LTN+ 1)

SUCC2

SUC Configuration Register 2 (0084_H) Application Reset Value: $0100\ 0504_H$



Field	Bits	Type	Description
LT	20:0	rw	**Listen Timeout(pdListenTimeout)** Configures wakeup / startup listen timeout in Microticks. The range for wakeup / startup listen timeout (pdListenTimeout) is 1284 to 1283846 (504_H to 139706_H) Microticks

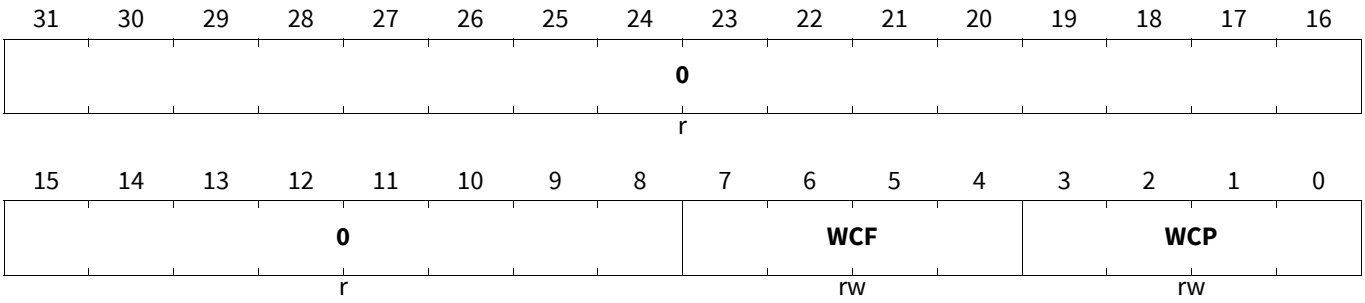
Field	Bits	Type	Description
LTN	27:24	rw	**Listen Time-out Noise (gListenNoise - 1)** Configures the upper limit for startup and wakeup listen timeout in the presence of noise expressed as a multiple of the cluster constant pdListenTimeout. The range of pdListenTimeout 2 to 16. LTN must be configured identical in all nodes of a cluster. This bit can be updated in "DEFAULT_CONFIG" or "CONFIG" state only. 1_H Listen Time-out Noise is equal 2 2_H Listen Time-out Noise is equal 3 3_H … E_H Listen Time-out Noise is equal 3 … 15 F_H Listen Time-out Noise is equal 16
0	23:21, 31:28	r	**Reserved** Returns 0 if read; should be written with 0.

SUC Configuration Register 3

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

SUCC3

SUC Configuration Register 3 (0088_H) Application Reset Value: $0000\ 0011_H$

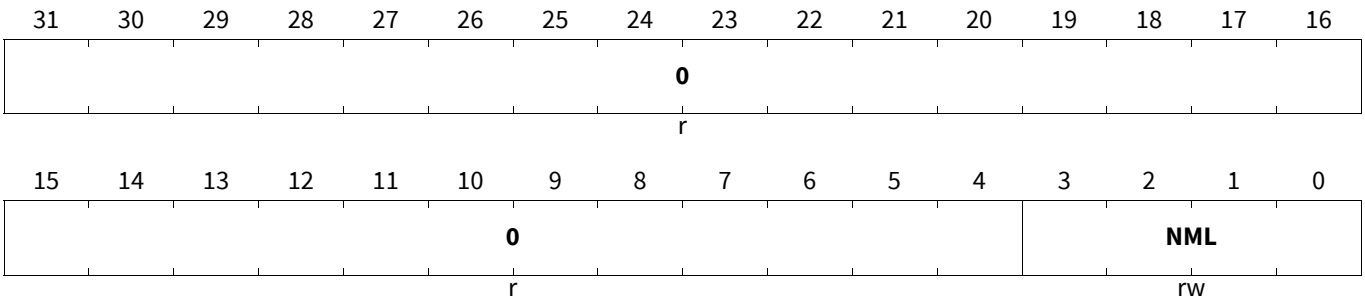


Field	Bits	Type	Description
WCP	3:0	rw	**Maximum Without Clock Correction Passive(gMaxWithoutClockCorrectionPassive)** Defines the number of consecutive even / odd cycle pairs with missing clock correction terms that will cause a transition from “NORMAL_ACTIVE” to “NORMAL_PASSIVE” state. Must be identical in all nodes of a cluster. Valid values are 1 to 15 (1_H to F_H) cycle pairs.
WCF	7:4	rw	**Maximum Without Clock Correction Fatal (gMaxWithoutClockCorrecti on Fatal)** Defines the number of consecutive even / odd cycle pairs with missing clock correction terms that will cause a transition from “NORMAL_ACTIVE” or “NORMAL_PASSIVE” to “HALT” state. Must be identical in all nodes of a cluster. Valid values are 1 to 15 (1_H to F_H)cycle pairs.
0	31:8	r	**Reserved** Returns 0 if read; should be written with 0.

NEM Configuration Register

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

NEMC
NEM Configuration Register **($008C_H$)** **Application Reset Value: $0000\ 0000_H$**

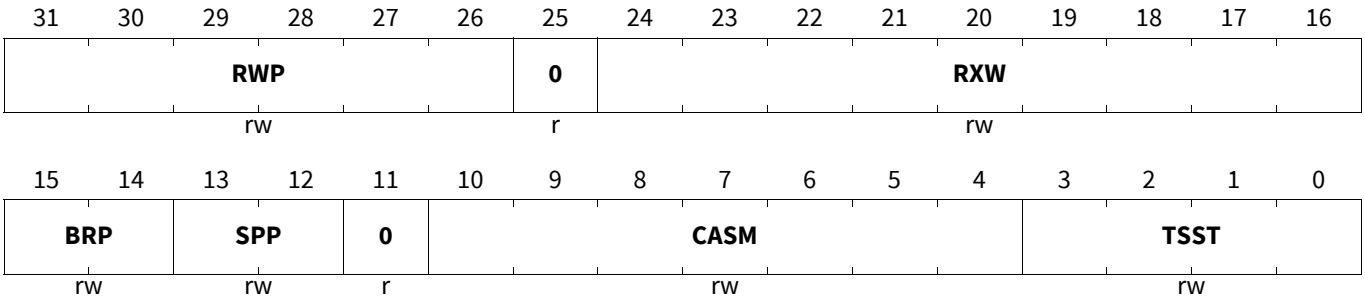


Field	Bits	Type	Description
NML	3:0	rw	**Network Management Vector Length(gNetworkManagementVectorLength)** These bits configure the length of the NM Vector. The configured length must be identical in all nodes of a cluster. Valid values are 0 to 12 (0_H to C_H) bytes.
0	31:4	r	**Reserved** Returns 0 if read; should be written with 0.

PRT Configuration Register 1

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

PRTC1
PRT Configuration Register 1 **(0090_H)** **Application Reset Value: $084C\ 0633_H$**



Field	Bits	Type	Description
TSST	3:0	rw	**Transmission Start Sequence Transmitter(gdTSSTransmitter)** Configures the duration of the Transmission Start Sequence (TSS) in terms of Bit Times (1 bit time = 4 Microticks = 100ns at 10Mbps). Must be identical in all nodes of a cluster. Valid values are 3 to 15 (3_H to F_H) Bit Times.

Field	Bits	Type	Description
CASM	10:4	rw	**Collision Avoidance Symbol Maximum (gdCASRxLowMax)** Configures the upper limit of the acceptance window for a collision avoidance symbol (CAS). Valid values are 67 to 99 (43_H to 63_H). Most significant bit of CASM is hard wired to 1 and can not be modified.
SPP	13:12	rw	**Strobe Point Position** Defines the sample count value for strobing. The strobed bit value is set to the voted value when the sample count is incremented to the value configured by SPP. *Note: The current revision 2.1 of the FlexRay™ protocol requires that SPP = 00_B. The alternate strobe point positions could be used to compensate for asymmetries in the physical layer.* 00_B Sample 5 (default) 01_B Sample 4 10_B Sample 6 11_B Reserved; should not be used.
BRP	15:14	rw	**Baud Rate Prescaler (gdSampleClockPeriod, pSamplePerMicrotick)** The baud rate prescaler configures the baud rate on the FlexRay™ bus. The baud rates listed below are valid with a sample clock f_{SCLK}= 80 MHz. One bit time always consists of 8 samples independent of the configured baud rate. 00_B 10 Mbit/s (1 Microtick= 25 ns; twice sampled with f_{SCLK}) gdSampleClockPeriod = 12.5 ns = 1 / $f_{SCLK\ pSamplesPerMicrotick = 2}$ 01_B 5 Mbit/s (1 Microtick= 25ns; single sampled with f_{SCLK} / 2) gdSampleClockPeriod = 25 ns = 2 / $f_{SCLK\ pSamplesPerMicrotick = 1}$ 10_B 2.5 Mbit/s (1 Microtick = 50ns; single sampled with f_{SCLK} / 4) gdSampleClockPeriod = 50 ns = 4 / $f_{SCLK\ pSamplesPerMicrotick = 1}$ 11_B Reserved; should not be used (2.5 Mbit/s (1 Microtick = 50 ns; single sampled with f_{SCLK} / 4) gdSampleClockPeriod = 50 ns = 4 / f_{SCLK} pSamplesPerMicrotick = 1
RXW	24:16	rw	**Wakeup Symbol Receive Window Length (gdWakeupSymbolRxWindow)** Configures the number of Bit Times used by the node to test the duration of the received wakeup pattern. Must be identical in all nodes of a cluster. Valid values are 76 to 301 ($4C_H$ to $12D_H$) Bit Times.
RWP	31:26	rw	**Repetitions of Tx Wakeup Pattern (pWakeupPattern)** Configures the number of repetitions (sequences) of the Tx wakeup symbol. Valid values are 2 to 63 (2_H to $3F_H$).
0	11, 25	r	**Reserved** Returns 0 if read; should be written with 0.

PRT Configuration Register 2

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

PRTC2

PRT Configuration Register 2 **(0094_H)** **Application Reset Value: $0F2D\ 0A0E_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**TXL**						**TXI**							
r		rw						rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**RXL**						**0**		**RXI**					
r		rw						r		rw					

Field	Bits	Type	Description
RXI	5:0	rw	**Wakeup Symbol Receive Idle(gdWakeupSymbolRxIdle)** Configures the number of Bit Times used by the node to test the duration of the idle phase of the received wakeup symbol. Must be identical in all nodes of a cluster. Valid values are 14 to 59 (E_H to $3B_H$) Bit Times.
RXL	13:8	rw	**Wakeup Symbol Receive Low (gdWakeupSymbolRxLow)** Configures the number of Bit Times used by the node to test the duration of the low phase of the received wakeup symbol. Must be identical in all nodes of a cluster. Valid values are 10 to 55 (A_H to 37_H) Bit Times.
TXI	23:16	rw	**Wakeup Symbol Transmit Idle (gdWakeupSymbolTxIdle)** Configures the number of Bit Times used by the node to transmit the idle phase of the wakeup symbol. Must be identical in all nodes of a cluster. Valid values are 45 to 180 ($2D_H$ to $B4_H$)Bit Times.
TXL	29:24	rw	**Wakeup Symbol Transmit Low (gdWakeupSymbolTxLow)** Configures the number of Bit Times used by the node to transmit the low phase of the wakeup symbol. Must be identical in all nodes of a cluster. Valid values are 15 to 60 (F_H to $3C_H$) Bit Times.
0	7:6, 15:14, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

MHD Configuration Register

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

MHDC

MHD Configuration Register **(0098_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**SLT**												
r			rw												

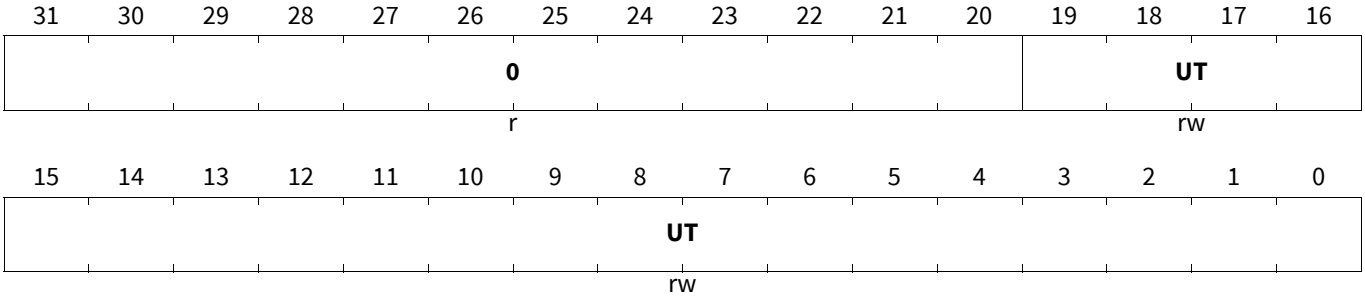
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									**SFDL**						
r									rw						

Field	Bits	Type	Description
SFDL	6:0	rw	**Static Frame Data Length(gPayloadLengthStatic)** Configures the cluster-wide payload length for all Frames sent in the static segment in double byte. The payload length must be identical in all nodes of a cluster. Valid values are 0 to 127 (0 to $7F_H$).
SLT	28:16	rw	**Start of Latest Transmit(pLatestTx)** Configures the maximum minislot value allowed before inhibiting Frame transmission in the dynamic segment of the cycle. There is no transmission dynamic segment if SLT is reset to zero. Valid values are 0 to 7981 (0 to $1F2D_H$) minislots.
0	15:7, 31:29	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 1

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC01
GTU Configuration Register 1 **($00A0_H$)** **Application Reset Value: $0000\ 0280_H$**



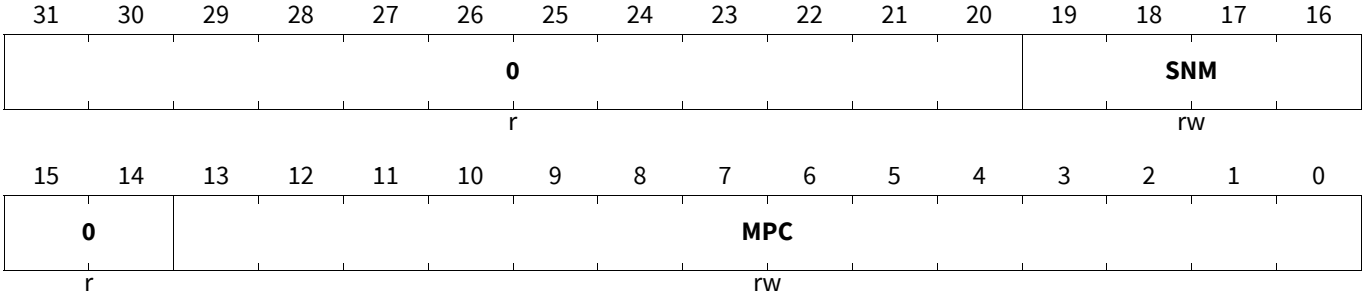
Field	Bits	Type	Description
UT	19:0	rw	**Microtick per Cycle(pMicroPerCycle)** Configures the duration of the communication cycle in Microticks. Valid values are 640 to 640000 (280_H to $9C400_H$) Microticks.
0	31:20	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 2

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC02

GTU Configuration Register 2 ($00A4_H$) Application Reset Value: $0002\ 000A_H$



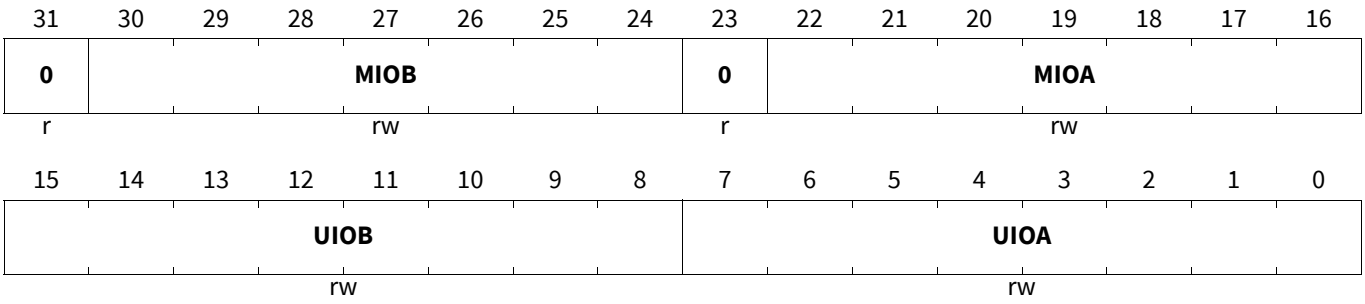
Field	Bits	Type	Description
MPC	13:0	rw	**Macrotick Per Cycle(gMacroPerCycle)** Configures the duration of one communication cycle in Macroticks. The cycle length must be identical in all nodes of a cluster. Valid values are 10 to 16000 (A_H to $3E80_H$) Macroticks.
SNM	19:16	rw	**Sync Node Max(gSyncNodeMax)** Maximum number of Frames within a cluster with SYNC Frame indicator bit SYN set to 1. Must be identical in all nodes of a cluster. Valid values are 2 to 15 (2_H to F_H).
0	15:14, 31:20	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 3

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC03

GTU Configuration Register 3 ($00A8_H$) Application Reset Value: $0202\ 0000_H$



Field	Bits	Type	Description
UIOA	7:0	rw	**Microtick Initial Offset Channel A(pMicroInitialOffset[A])** Configures the number of Microticks between the actual time reference point on channel A and the subsequent Macrotick boundary of the secondary time reference point. The parameter depends on pDelayCompensation[A] and therefore has to be set for each channel independently. Valid values are 0 to 240 (0_H to $F0_H$) Microticks.

Field	Bits	Type	Description
UIOB	15:8	rw	**Microtick Initial Offset Channel B (pMicroInitialOffset[B])** Configures the number of Microticks between the actual time reference point on channel B and the subsequent Macrotick boundary of the secondary time reference point. The parameter depends on pDelayCompensation[B] and therefore has to be set for each channel independently. Valid values are 0 to 240 (0_H to $F0_H$) Microticks.
MIOA	22:16	rw	**Macrotick Initial Offset Channel A(gMacroInitialOffset[A])** Configures the number of Macroticks between the static slot boundary and the subsequent Macrotick boundary of the secondary time reference point based on the nominal Macrotick duration. Must be identical in all nodes of a cluster. Valid values are 2 to 72 (2_H to 48_H) Macroticks.
MIOB	30:24	rw	**Macrotick Initial Offset Channel B(gMacroInitialOffset[B])** Configures the number of Macroticks between the static slot boundary and the subsequent Macrotick boundary of the secondary time reference point based on the nominal Macrotick duration. Must be identical in all nodes of a cluster. Valid values are 2 to 72 (2_H to 48_H) Macroticks.
0	23, 31	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 4

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only. For details about configuration of NIT and OCS see “Configuration of Network Idle Time (NIT) Start and Offset Correction Start”.

GTUC04

GTU Configuration Register 4 **($00AC_H$)** **Application Reset Value: $0008\ 0007_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**OCS**													
r		rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**NIT**													
r		rw													

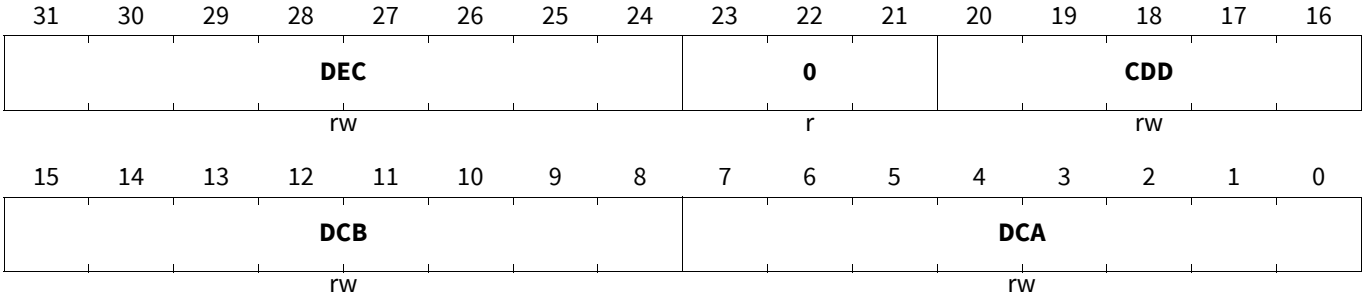
Field	Bits	Type	Description
NIT	13:0	rw	**Network Idle Time Start(gMacroPerCycle - gdNIT - 1)** Configures the starting point of the Network Idle Time (NIT) at the end of the communication cycle expressed in terms of Macroticks from the beginning of the cycle. The start of network idle time (NIT) is recognized if Macrotick = gMacroPerCycle - gdNIT -1 and the increment pulse of Macrotick is set. Must be identical in all nodes of a cluster. Valid values are 7 to 15997 (7_H to $3E7D_H$) Macroticks.

Field	Bits	Type	Description
OCS	29:16	rw	**Offset Correction Start (gOffsetCorrectionStart - 1)** Determines the start of the offset correction within the network idle time (NIT) phase, calculated from start of cycle. Must be identical in all nodes of a cluster. For cluster consisting of E-Ray implementations only, it is sufficient to program OCS = NIT + 1. Valid values are 8 to 15998 (8_H to $3E7E_H$) Macroticks.
0	15:14, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 5

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC05
GTU Configuration Register 5 **($00B0_H$)** **Application Reset Value: $0E00\ 0000_H$**



Field	Bits	Type	Description
DCA	7:0	rw	**Delay Compensation Channel A(pDelayCompensation[A])** Used to compensate for reception delays on channel A. This covers assumed propagation delay up to cPropagationDelayMax for Microticks in the range of 0.0125 µs to 0.05 µs. In practice, the minimum of the propagation delays of all sync nodes should be applied. Valid values are 0 to 200 (0_H to $C8_H$) Microticks.
DCB	15:8	rw	**Delay Compensation Channel B (pDelayCompensation[B])** Used to compensate for reception delays on channel B. This covers assumed propagation delay up to cPropagationDelayMax for Microticks in the range of 0.0125 to 0.05 µs. In practice, the minimum of the propagation delays of all sync nodes should be applied. Valid values are 0 to 200 (0_H to $C8_H$) Microticks.
CDD	20:16	rw	**Cluster Drift Damping(pClusterDriftDamping)** Configures the cluster drift damping value used in clock synchronization to minimize accumulation of rounding errors. Valid values are 0 to 20 (0_H to 14_H) Microticks.
DEC	31:24	rw	**Decoding Correction(pDecodingCorrection)** Configures the decoding correction value used to determine the primary time reference point. Valid values are 14 to 143 (E_H to $8F_H$) Microticks.

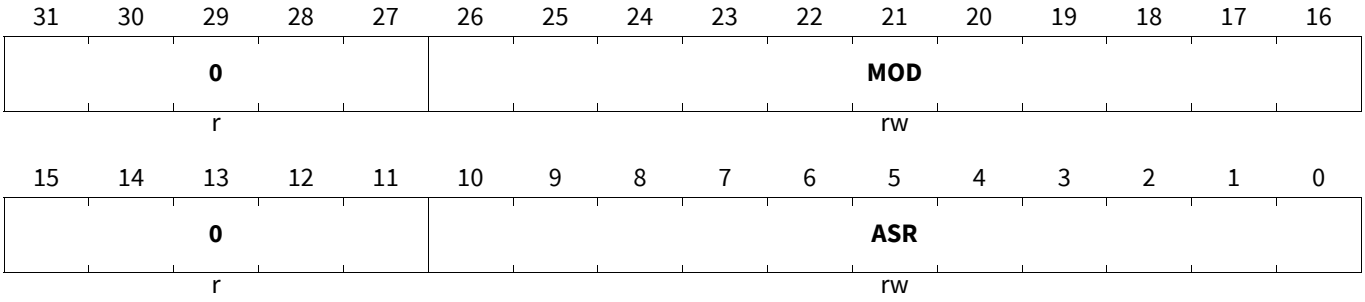
Field	Bits	Type	Description
0	23:21	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 6

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC06

GTU Configuration Register 6 (00B4$_H$) Application Reset Value: 0002 0000$_H$



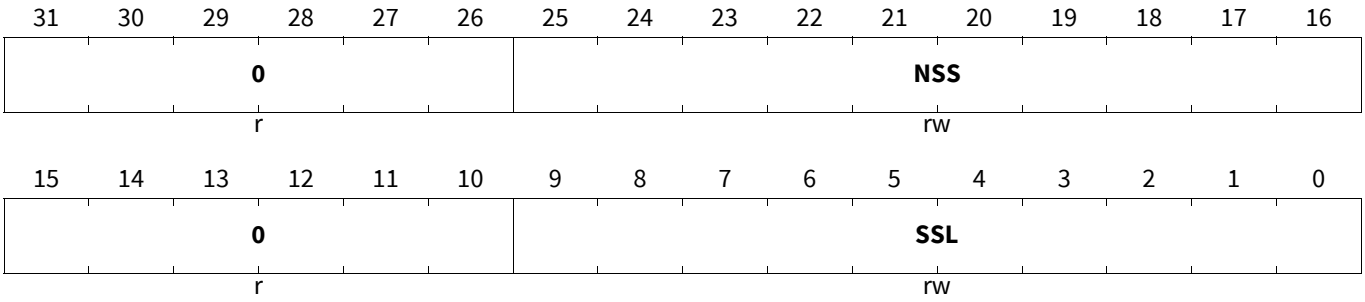
Field	Bits	Type	Description
ASR	10:0	rw	**Accepted Startup Range(pdAcceptedStartupRange)** Number of Microticks constituting the expanded range of measured deviation for startup Frames during integration. Valid values are 0 to 1875 (0_H to 753_H) Microticks.
MOD	26:16	rw	**Maximum Oscillator Drift(pdMaxDrift)1)** Maximum drift offset between two nodes that operate with unsynchronized clocks over one communication cycle in Microticks. Valid values are 2 to 1923 (2_H to 783_H) Microticks.
0	15:11, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 7

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC07

GTU Configuration Register 7 (00B8$_H$) Application Reset Value: 0002 0004$_H$



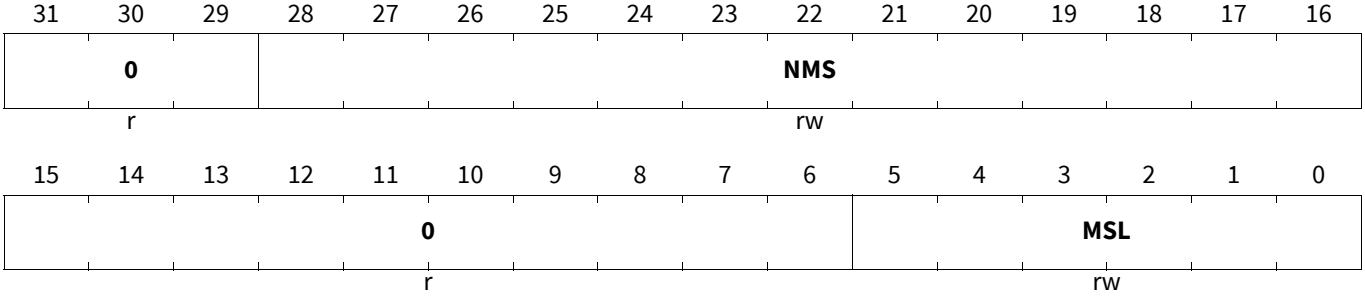
Field	Bits	Type	Description
SSL	9:0	rw	**Static Slot Length(gdStaticSlot)** Configures the duration of a static slot in Macroticks. The static slot length must be identical in all nodes of a cluster. Valid values are 4 to 659 (4_H to 293_H) Macroticks.
NSS	25:16	rw	**Number of Static Slots(gNumberOfStaticSlots)** Configures the number of static slots in a cycle. At least 2 coldstart nodes must be configured to startup a FlexRay™ network. The number of static slots must be identical in all nodes of a cluster. Valid values are 2 to 1023 (2_H to $3FF_H$).
0	15:10, 31:26	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 8

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC08

GTU Configuration Register 8 ($00BC_H$) Application Reset Value: $0000\ 0002_H$



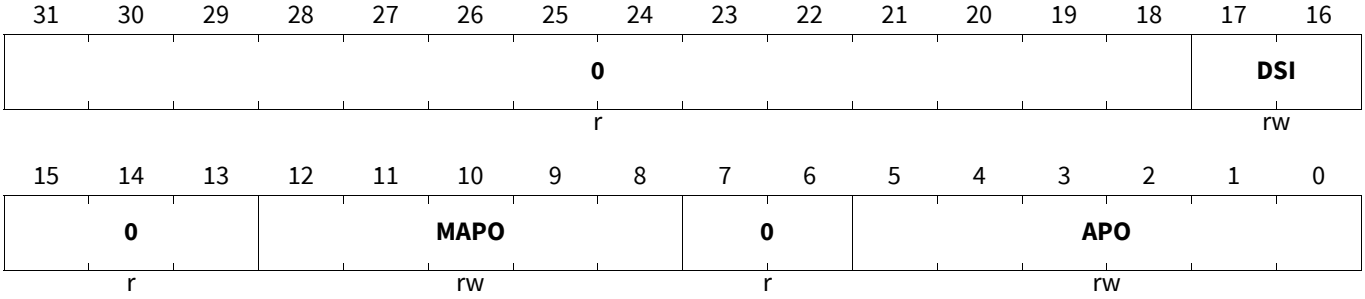
Field	Bits	Type	Description
MSL	5:0	rw	**Minislot Length(gdMinislot)** Configures the duration of a minislot in Macroticks. The minislot length must be identical in all nodes of a cluster. Valid values are 2 to 63 (2_H to $3F_H$) Macroticks.
NMS	28:16	rw	**Number of Minislots(gNumberOfMinislots)** Configures the number of minislots within the dynamic segment of a cycle. The number of minislots must be identical in all nodes of a cluster. Valid values are 0 to 7986 (0_H to $1F32_H$).
0	15:6, 31:29	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 9

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC09

GTU Configuration Register 9 ($00C0_H$) Application Reset Value: $0000\ 0101_H$



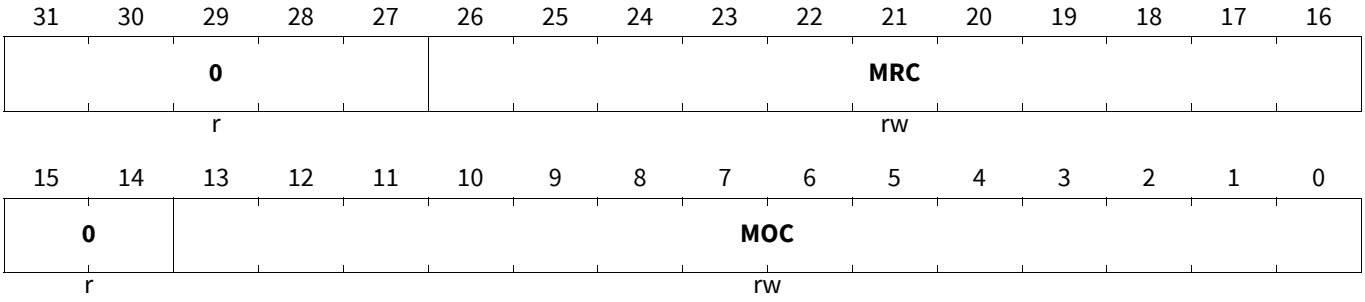
Field	Bits	Type	Description
APO	5:0	rw	**Action Point Offset(gdActionPointOffset)** Configures the action point offset in Macroticks within static slots and symbol window. Must be identical in all nodes of a cluster. Valid values are 1 to 63 (1_H to $3F_H$) Macroticks.
MAPO	12:8	rw	**Minislot Action Point Offset (gd Minislot Action Point Offset)** Configures the action point offset in Macroticks within the minislots of the dynamic segment. Must be identical in all nodes of a cluster. Valid values are 1 to 31 (1_H to $1F_H$) Macroticks.
DSI	17:16	rw	**Dynamic Slot Idle Phase (gdDynamicSlotIdlePhase)** The duration of the dynamic slot idle phase has to be greater or equal than the idle detection time. Must be identical in all nodes of a cluster. Valid values are 0 to 2 Minislot.
0	7:6, 15:13, 31:18	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 10

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

GTUC10

GTU Configuration Register 10 ($00C4_H$) Application Reset Value: $0002\ 0005_H$



Field	Bits	Type	Description
MOC	13:0	rw	**Maximum Offset Correction(pOffsetCorrectionOut)** Holds the maximum permitted offset correction value to be applied by the internal clock synchronization algorithm (absolute value). The Communication Controller checks only the internal offset correction value against the maximum offset correction value. Valid values are 5 to 15266 (5_H to $3BA2_H$) Microticks.
MRC	26:16	rw	**Maximum Rate Correction (pRateCorrectionOut)** Holds the maximum permitted rate correction value to be applied by the internal clock synchronization algorithm. The communication controller checks only the internal rate correction value against the maximum rate correction value (absolute value). Valid values are 2 to 1923 (2_H to 783_H) Microticks.
0	15:14, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

GTU Configuration Register 11

GTUC11
GTU Configuration Register 11 ($00C8_H$) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					ERC			0					EOC		
r					rw			r					rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						ERCC		0						EOCC	
r						rw		r						rw	

Field	Bits	Type	Description
EOCC	1:0	rw	**External Offset Correction Control(pExternOffsetControl)** By writing to EOCC the external offset correction is enabled as specified below. Should be modified only outside network idle time (NIT). 00_B No external clock correction 01_B No external clock correction 10_B External offset correction value subtracted from calculated offset correction value 11_B External offset correction value added to calculated offset correction value

Field	Bits	Type	Description
ERCC	9:8	rw	**External Rate Correction Control(pExternRateControl)** By writing to ERCC the external rate correction is enabled as specified below. Should be modified only outside network idle time (NIT). 00_B No external rate correction 01_B No external rate correction 10_B External rate correction value subtracted from calculated rate correction value 11_B External rate correction value added to calculated rate correction value
EOC	18:16	rw	**External Offset Correction(pExternOffsetCorrection)** Holds the external clock offset correction value in Microticks to be applied by the internal synchronization algorithm. The value is subtracted / added from / to the calculated offset correction value. The value is applied during network idle time (NIT). May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. Valid values are 0 to 7 Microticks.
ERC	26:24	rw	**External Rate Correction (pExternRateCorrection)** Holds the external rate correction value in Microticks to be applied by the internal clock synchronization algorithm. The value is subtracted / added from / to the calculated rate correction value. The value is applied during network idle time (NIT). Can be modified in “DEFAULT_CONFIG” or “CONFIG” state only. Valid values are 0 to 7 Microticks.
0	7:2, 15:10, 23:19, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

41.4.2.5 Communication Controller Status Registers

During 8/16-bit accesses to status variables coded with more than 8/16-bit, the variable might be updated by the Communication Controller between two accesses (non-atomic read accesses). The status vector may change faster than the Host can poll the status vector, depending on f_{CLC_ERAY} frequency.

Communication Controller Status Vector

CCSV

Communication Controller Status Vector **(0100_H)** **Application Reset Value: $0010\ 4000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**PSL**						**RCA**					**WSV**		
r		rh						rh					rh		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**CSI**	**CSAI**	**CSNI**	**0**		**SLM**		**HRQ**	**FSI**	**POCS**					
r	rh	rh	rh	r		rh		rh	rh	rh					

Field	Bits	Type	Description
POCS	5:0	rh	**Protocol Operation Control Status** 00_H ... 05_H, $0F_H$ Indicates the actual state of operation of the Communication Controller Protocol Operation Control 06_H … 14_H are reserved. 10_H ... 13_H Indicates the actual state of operation of the POC in the wakeup path 14_H … $1F_H$ are reserved. 20_H ... $2B_H$ Indicates the actual state of operation of the POC in the startup path $2C_H$ … $3F_H$ are reserved. 00_H “DEFAULT_CONFIG” state 01_H “READY” state 02_H “NORMAL_ACTIVE” state 03_H “NORMAL_PASSIVE” state 04_H “HALT” state 05_H “MONITOR_MODE” state $0F_H$ “CONFIG” state 10_H WAKEUP_STANDBY state 11_H “WAKEUP_LISTEN” state 12_H “WAKEUP_SEND” state 13_H “WAKEUP_DETECT” state 20_H “STARTUP_PREPARE” state 21_H “COLDSTART_LISTEN” state 22_H “COLDSTART_COLLISION_RESOLUTION state 23_H “COLDSTART_CONSISTENCY_CHECK” state 24_H “COLDSTART_GAP state 25_H “COLDSTART_JOIN” State 26_H “INTEGRATION_COLDSTART_CHECK” state 27_H “INTEGRATION_LISTEN” state 28_H “INTEGRATION_CONSISTENCY_CHECK” state 29_H “INITIALIZE_SCHEDULE” state $2A_H$ “ABORT_STARTUP” state $2B_H$ “STARTUP_SUCCESS” state
FSI	6	rh	**Freeze Status Indicator(vPOC!Freeze)** Indicates that the POC has entered the “HALT” state due to CHI command “FREEZE” or due to an error condition requiring an immediate POC halt. Reset by transition from “HALT” to “DEFAULT_CONFIG” state.
HRQ	7	rh	**Halt Request(vPOC!CHIHaltRequest)** Indicates that a request from the Host has been received to halt the POC at the end of the communication cycle. Reset by transition from “HALT” to “DEFAULT_CONFIG” state or when entering “READY” state.

Field	Bits	Type	Description
SLM	9:8	rh	**Slot Mode(vPOC!SlotMode)** Indicates the actual slot mode of the POC in states READY, WAKEUP, STARTUP, NORMAL_ACTIVE, and NORMAL_PASSIVE. Default is “SINGLE”. Changes to “ALL”, depending on configuration bit SUCC1.TSM. In “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state the CHI command “ALL_SLOTS” will change the slot mode from “SINGLE” over “ALL_PENDING” to “ALL”. Set to SINGLE in all other states. 00_B SINGLE 01_B Reserved 10_B ALL_PENDING 11_B ALL
CSNI	12	rh	**Coldstart Noise Indicator(vPOC!ColdstartNoise)** Indicates that the cold start procedure occurred under noisy conditions. Reset by CHI command “RESET_STATUS_INDICATORS” or by transition from “HALT” to “DEFAULT_CONFIG” state or from “READY” to “STARTUP” state.
CSAI	13	rh	**Coldstart Abort Indicator** Coldstart aborted. Reset by CHI command “RESET_STATUS_INDICATORS” or by transition from “HALT” to “DEFAULT_CONFIG” state or from “READY” to “STARTUP” state.
CSI	14	rh	**Cold Start Inhibit(vColdStartInhibit)** Indicates that the node is disabled from cold starting. The flag is set whenever the POC enters “READY” state due to CHI command “READY”. The flag has to be reset under control of the Host by CHI command “ALLOW_COLDSTART” (SUCC1.CMD = 1001_B). 0_B Cold starting of node enabled 1_B Cold starting of node disabled

Field	Bits	Type	Description
WSV	18:16	rh	**Wakeup Status(vPOC!WakeupStatus)** Indicates the status of the current wakeup attempt. Reset by CHI command “RESET_STATUS_INDICATORS” or by transition from “HALT” to “DEFAULT_CONFIG” state. 000_B UNDEFINED. Wakeup not yet executed by the Communication Controller. 001_B RECEIVED_HEADER. Set when the Communication Controller finishes wakeup due to the reception of a Frame Header without coding violation on either channel in “WAKEUP_LISTEN” state. 010_B RECEIVED_WUP. Set when the Communication Controller finishes wakeup due to the reception of a valid wakeup pattern on the configured wakeup channel in “WAKEUP_LISTEN” state. 011_B COLLISION_HEADER. Set when the Communication Controller stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid Header on either channel. 100_B COLLISION_WUP. Set when the Communication Controller stops wakeup due to a detected collision during wakeup pattern transmission by receiving a valid wakeup pattern on the configured wakeup channel. 101_B COLLISION_UNKNOWN. Set when the Communication Controller stops wakeup by leaving “WAKEUP_DETECT” state after expiration of the wakeup timer without receiving a valid wakeup pattern or a valid Frame Header. 110_B TRANSMITTED. Set when the Communication Controller has successfully completed the transmission of the wakeup pattern. 111_B Reserved
RCA	23:19	rh	**Remaining Coldstart Attempts(vRemainingColdstartAttempts)** Indicates the number of remaining coldstart attempts. The RUN command resets this counter to the maximum number of coldstart attempts as configured by SUCC1.CSA.
PSL	29:24	rh	**POC Status Log** Status of CCSV.POCS immediately before entering “HALT” state. Set when entering “HALT” state. Set to “HALT” when FREEZE command is applied during “HALT” state. Reset to 000000_B when leaving “HALT” state.
0	11:10, 15, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

Communication Controller Error Vector

Reset by transition from “HALT” to “DEFAULT_CONFIG” state or when entering “READY” state.

CCEV

Communication Controller Error Vector **(0104_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			PTAC					ERRM		0		CCFC			
r			rh					rh		r		rh			

Field	Bits	Type	Description
CCFC	3:0	rh	**Clock Correction Failed Counter(vClockCorrectionFailed)** The Clock Correction Failed Counter is incremented by one at the end of any odd communication cycle where either the missing offset correction error or missing rate correction error are active. The Clock Correction Failed Counter is reset to 0 at the end of an odd communication cycle if neither the offset correction failed nor the rate correction failed errors are active. The Clock Correction Failed Counter stops at 15.
ERRM	7:6	rh	**Error Mode(vPOC!ErrorMode)** Indicates the actual error mode of the POC. 00_B “ACTIVE” (green) 01_B “PASSIVE” (yellow) 10_B “COMM_HALT” (red) 11_B Reserved
PTAC	12:8	rh	**Passive to Active Count(vAllowPassiveToActive)** Indicates the number of consecutive even / odd cycle pairs that have passed with valid rate and offset correction terms, while the node is waiting to transit from “NORMAL_PASSIVE” state to “NORMAL_ACTIVE” state. The transition takes place when PTAC equals SUCC1.PTA-1.
0	5:4, 31:13	r	**Reserved** Returns 0 if read; should be written with 0.

Slot Counter Value

SCV

Slot Counter Value **(0110_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					SCCB										
r					rh										

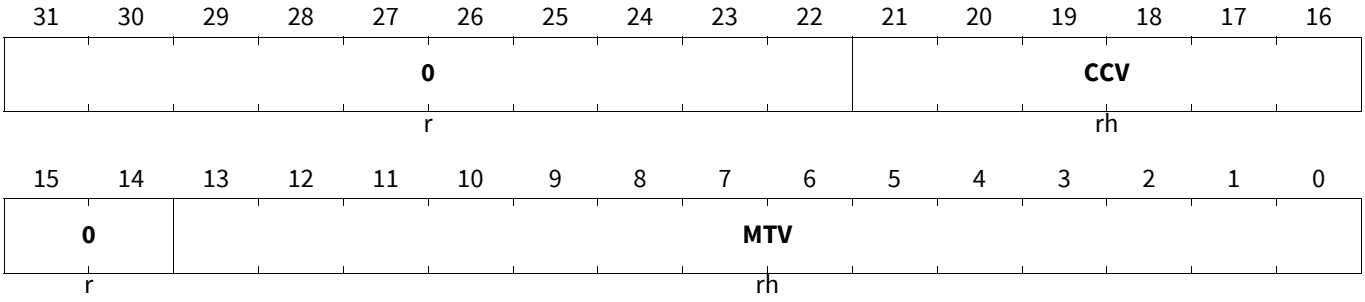
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					SCCA										
r					rh										

Field	Bits	Type	Description
SCCA	10:0	rh	**Slot Counter Channel A(vSlotCounter[A])** Current slot counter value on channel A. The value is incremented by the Communication Controller and reset at the start of a communication cycle. Valid values are 0 to 2047 (0_H to $7FF_H$).
SCCB	26:16	rh	**Slot Counter Channel B(vSlotCounter[B])** Current slot counter value on channel B. The value is incremented by the Communication Controller and reset at the start of a communication cycle. Valid values are 0 to 2047 (0_H to $7FF_H$).
0	15:11, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

Macrotick and Cycle Counter Value

MTCCV

Macrotick and Cycle Counter Value (0114_H) Application Reset Value: $0000\ 0000_H$

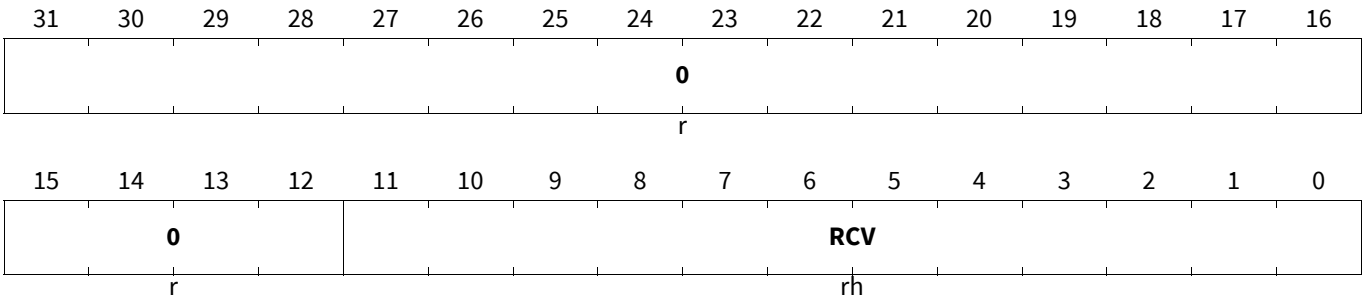


Field	Bits	Type	Description
MTV	13:0	rh	**Macrotick Value(vMacrotick)** Current Macrotick value. The value is incremented by the Communication Controller and reset at the start of a communication cycle. Valid values are 0 to 16000 (0_H to $3E80_H$).
CCV	21:16	rh	**Cycle Counter Value(vCycleCounter)** Current cycle counter value. The value is incremented by the Communication Controller at the start of a communication cycle. Valid values are 0 to 63 (0_H to $3F_H$).
0	15:14, 31:22	r	**Reserved** Returns 0 if read; should be written with 0.

Rate Correction Value

RCV

Rate Correction Value **(0118_H)** **Application Reset Value: $0000\ 0000_H$**



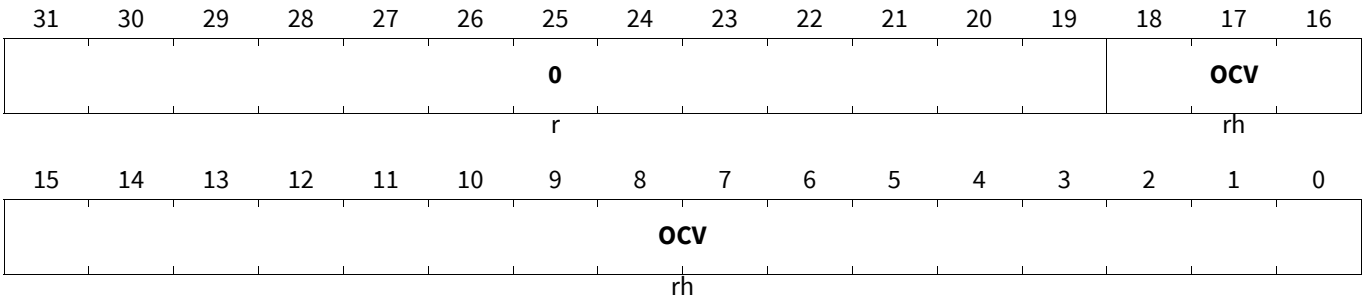
Field	Bits	Type	Description
RCV	11:0	rh	**Rate Correction Value(vRateCorrection)** Rate correction value (two's complement). Calculated internal rate correction value before limitation. If the RCV value exceeds the limits defined by GTUC10.MRC, flag SFS.RCLR is set to 1.
0	31:12	r	**Reserved** Returns 0 if read; should be written with 0.

Offset Correction Value

Note: The external rate / offset correction value is added to the limited rate / offset correction value.

OCV

Offset Correction Value **($011C_H$)** **Application Reset Value: $0000\ 0000_H$**



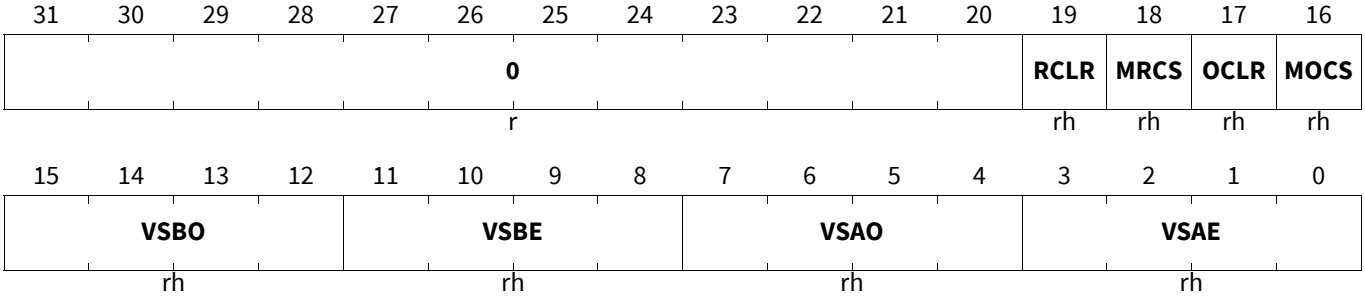
Field	Bits	Type	Description
OCV	18:0	rh	**Offset Correction Value(vOffsetCorrection)** Offset correction value (two's complement). Calculated internal offset correction value before limitation. If the OCV value exceeds the limits defined by GTUC10.MOC flag SFS.OCLR is set to 1.
0	31:19	r	**Reserved** Returns 0 if read; should be written with 0.

SYNC Frame Status

The maximum number of valid SYNC Frames in a communication cycle is 15.

SFS

SYNC Frame Status **(0120_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
VSAE	3:0	rh	**Valid SYNC Frames Channel A, even communication cycle** Holds the number of valid SYNC Frames received on channel A in the even communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the network idle time (NIT) of each even communication cycle. This bit field is only valid if the channel A is assigned to the Communication Controller by SUCC1.CCHA.
VSAO	7:4	rh	**Valid SYNC Frames Channel A, odd communication cycle** Holds the number of valid SYNC Frames received on channel A in the odd communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the network idle time (NIT) of each odd communication cycle. This bit field is only valid if the channel A is assigned to the Communication Controller by SUCC1.CCHA.
VSBE	11:8	rh	**Valid SYNC Frames Channel B, even communication cycle** Holds the number of valid SYNC Frames received on channel B in the even communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the network idle time (NIT) of each even communication cycle. This bit field is only valid if the channel B is assigned to the Communication Controller by SUCC1.CCHB.
VSBO	15:12	rh	**Valid SYNC Frames Channel B, odd communication cycle** Holds the number of valid SYNC Frames received on channel B in the odd communication cycle. If transmission of SYNC Frames is enabled by SUCC1.TXSY the value is incremented by one. The value is updated during the network idle time (NIT) of each odd communication cycle. This bit field is only valid if the channel B is assigned to the Communication Controller by SUCC1.CCHB.
MOCS	16	rh	**Missing Offset Correction Signal** The Missing Offset Correction flag signals to the Host, that no offset correction calculation can be performed because no SYNC Frames were received. The flag is updated by the Communication Controller at start of offset correction phase. 0_B Offset correction signal valid 1_B Missing offset correction signal

Field	Bits	Type	Description
OCLR	17	rh	**Offset Correction Limit Reached** The Offset Correction Limit Reached flag signals to the Host, that the offset correction value has exceeded its limit as defined by GTUC10.MOC. The flag is updated by the Communication Controller at start of offset correction phase. 0_B Offset correction below limit 1_B Offset correction limit reached
MRCS	18	rh	**Missing Rate Correction Signal** The Missing Rate Correction Flag signals to the Host, that no rate correction calculation can be performed because no pairs of even / odd SYNC Frames were received. The flag is updated by the Communication Controller at start of offset correction phase. 0_B Rate correction signal valid 1_B Missing rate correction signal
RCLR	19	rh	**Rate Correction Limit Reached** The Rate Correction Limit Reached flag signals to the Host, that the rate correction value has exceeded its limit.as defined by GTUC10.MRC. The flag is updated by the Communication Controller at start of offset correction phase. 0_B Rate correction below limit 1_B Rate correction limit reached
0	31:20	r	**Reserved** Returns 0 if read; should be written with 0.

Symbol Window and Network Idle Time Status

Symbol window related status information. Updated by the Communication Controller at the end of the symbol window for each channel. During startup the status data is not updated.

Note: MTSA and MTSB may be changed outside "DEFAULT_CONFIG" or "CONFIG" state when the write to SUC Configuration Register 1 (SUCC1) register is directly preceded by the unlock sequence as described in "Lock Register". This may be combined with CHI command SEND_MTS. If both bits MTSA and MTSB are set to 1 an MTS symbol will be transmitted on both channels when requested by writing SUCC1.CMD = 1000_B

SWNIT

Symbol Window and Network Idle Time Status (0124_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				SBNB	SENB	SBNA	SENA	MTSB	MTSA	TCSB	SBSB	SESB	TCSA	SBSA	SESA
r				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
SESA	0	rh	**Syntax Error in Symbol Window Channel A(vSS!SyntaxErrorA)** 0_B No syntax error detected 1_B Syntax error during symbol window detected on channel A
SBSA	1	rh	**Slot Boundary Violation in Symbol Window Channel A(vSS!BViolationA)** 0_B No slot boundary violation detected 1_B Slot boundary violation during symbol window detected on channel A
TCSA	2	rh	**Transmission Conflict in Symbol Window Channel A(vSS!TxConflictA)** 0_B No transmission conflict detected 1_B Transmission conflict in symbol window detected on channel A
SESB	3	rh	**Syntax Error in Symbol Window Channel B(vSS!SyntaxErrorB)** 0_B No syntax error detected 1_B Syntax error during symbol window detected on channel B
SBSB	4	rh	**Slot Boundary Violation in Symbol Window Channel B(vSS!BViolationB)** 0_B No slot boundary violation detected 1_B Slot boundary violation during symbol window detected on channel B
TCSB	5	rh	**Transmission Conflict in Symbol Window Channel B(vSS!TxConflictB)** 0_B No transmission conflict detected 1_B Transmission conflict in symbol window detected on channel B
MTSA	6	rh	**MTS Received on Channel A(vSS!ValidMTSA)** Media Access Test symbol received on channel A during the proceeding symbol window. Updated by the Communication Controller for each channel at the end of the symbol window. When this bit is set to 1, also interrupt flag SIR.MTSA is set to 1. 0_B No MTS symbol received on channel A 1_B MTS symbol received on channel A
MTSB	7	rh	**MTS Received on Channel B(vSS!ValidMTSB)** Media Access Test symbol received on channel B during the proceeding symbol window. Updated by the Communication Controller for each channel at the end of the symbol window. When this bit is set to 1, also interrupt flag SIR.MTSB is set to 1. 0_B No MTS symbol received on channel B 1_B MTS symbol received on channel B
SENA	8	rh	**Syntax Error during network idle time (NIT) Channel A(vSS!SyntaxErrorA)** Updated by the Communication Controller channel A at the end of the NIT. 0_B No syntax error detected 1_B Syntax error during network idle time (NIT) detected on channel A

Field	Bits	Type	Description
SBNA	9	rh	**Slot Boundary Violation during network idle time (NIT) Channel A(vSS!BViolationA)** Updated by the Communication Controller channel A at the end of the NIT. 0_B No slot boundary violation detected 1_B Slot boundary violation during network idle time (NIT) detected on channel A
SENB	10	rh	**Syntax Error during network idle time (NIT) Channel B(vSS!SyntaxErrorB)** Updated by the Communication Controller channel B at the end of the NIT. 0_B No syntax error detected 1_B Syntax error during network idle time (NIT) detected on channel B
SBNB	11	rh	**Slot Boundary Violation during network idle time (NIT) Channel B(vSS!BViolationB)** Updated by the Communication Controller channel B at the end of the NIT. 0_B No slot boundary violation detected 1_B Slot boundary violation during network idle time (NIT) detected on channel B
0	31:12	r	**Reserved** Returns 0 if read; should be written with 0.

Aggregated Channel Status

The aggregated channel status provides the Host with an accrued status of channel activity for all communication slots regardless of whether they are assigned for transmission or subscribed for reception. The aggregated channel status also includes status data from the symbol window and the network idle time. The status data is updated (set) after each slot and aggregated until it is reset by the Host. During startup the status data is not updated. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. An application reset will also clear the register.

Note: The set condition of flags CIA and CIB is also fulfilled if there is only one single Frame in the slot and the slot boundary at the end of the slot is reached during the Frames channel idle recognition phase. When one of the flags SEDB, CEDB, CIB, SBVB changes from 0 to 1, service request flag EIR.EDB is set to 1.When one of the flags SEDA, CEDA, CIA, SBVA changes from 0 to 1, service request flag EIR.EDA is set to 1. LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all rwh bits in this register.

ACS

Aggregated Channel Status **(0128_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			SBVB	CIB	CEDB	SEDB	VFRB	0			SBVA	CIA	CEDA	SEDA	VFRA
r			rwh	rwh	rwh	rwh	rwh	r			rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
VFRA	0	rwh	**Valid Frame Received on Channel A(vSS!ValidFrameA)** One or more valid Frames were received on channel A in any static or dynamic slot during the observation period. 0_B No valid Frame received 1_B Valid Frame(s) received on channel A
SEDA	1	rwh	**Syntax Error Detected on Channel A(vSS!SyntaxErrorA)** One or more syntax errors in static or dynamic slots, symbol window, and network idle time (NIT) were observed on channel A. 0_B No syntax error observed 1_B Syntax error(s) observed on channel A
CEDA	2	rwh	**Content Error Detected on Channel A(vSS!ContentErrorA)** One or more Frames with a content error were received on channel A in any static or dynamic slot during the observation period. 0_B No Frame with content error received 1_B Frame(s) with content error received on channel A
CIA	3	rwh	**Communication Indicator Channel A** One or more valid Frames were received on channel A in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid Frame AND had any combination of either syntax error OR content error OR slot boundary violation. 0_B No valid Frame(s) received in slots containing any additional communication 1_B Valid Frame(s) received on channel A in slots containing any additional communication
SBVA	4	rwh	**Slot Boundary Violation on Channel A(vSS!BViolationA)** One or more slot boundary violations were observed on channel A at any time during the observation period (static or dynamic slots, symbol window, and network idle time NIT). 0_B No slot boundary violation observed 1_B Slot boundary violation(s) observed on channel A

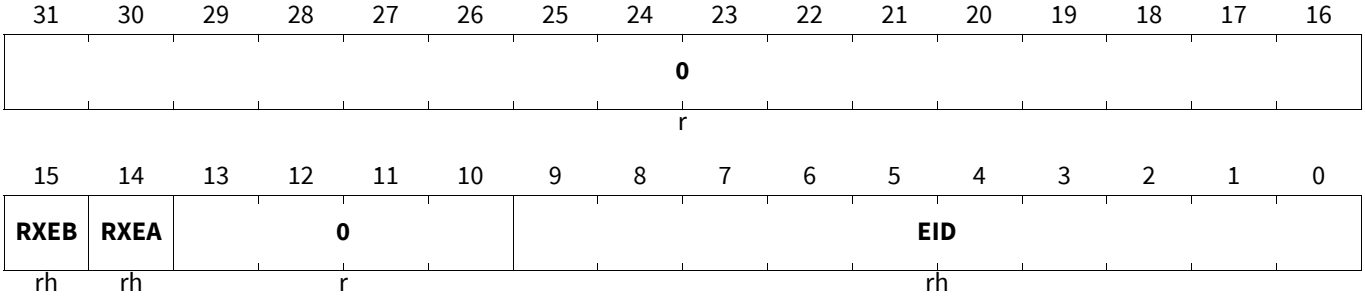
Field	Bits	Type	Description
VFRB	8	rwh	**Valid Frame Received on Channel B(vSS!ValidFrameB)** One or more valid Frames were received on channel B in any static or dynamic slot during the observation period. 0_B No valid Frame received 1_B Valid Frame(s) received on channel B
SEDB	9	rwh	**Syntax Error Detected on Channel B(vSS!SyntaxErrorB)** One or more syntax errors in static or dynamic slots, symbol window, and network idle time (NIT) were observed on channel B. 0_B No syntax error observed 1_B Syntax error(s) observed on channel B
CEDB	10	rwh	**Content Error Detected on Channel B(vSS!ContentErrorB)** One or more Frames with a content error were received on channel B in any static or dynamic slot during the observation period. 0_B No Frame with content error received 1_B Frame(s) with content error received on channel B
CIB	11	rwh	**Communication Indicator Channel B** One or more valid Frames were received on channel B in slots that also contained any additional communication during the observation period, i.e. one or more slots received a valid Frame AND had any combination of either syntax error OR content error OR slot boundary violation. 0_B No valid Frame(s) received in slots containing any additional communication 1_B Valid Frame(s) received on channel B in slots containing any additional communication
SBVB	12	rwh	**Slot Boundary Violation on Channel B(vSS!BViolationB)** One or more slot boundary violations were observed on channel B at any time during the observation period (static or dynamic slots, symbol window, and network idle time NIT). 0_B No slot boundary violation observed 1_B Slot boundary violation(s) observed on channel B
0	7:5, 31:13	r	**Reserved** Returns 0 if read; should be written with 0.

Even Sync ID Symbol Window n

Registers Even Sync ID nn (ESIDn, n=01-15) hold the Frame IDs of the SYNC Frames received in **even** communication cycles, sorted in ascending order, with register ESID01 holding the lowest received SYNC Frame ID. If the node itself transmits a SYNC Frame in an even communication cycle, register ESID01 holds the respective SYNC Frame ID as configured in Message Buffer 0 and the flags RXEA, RXEB are set. The value is updated during the network idle time (NIT) of each even communication cycle.

ESIDn (n=01-15)

Even Sync ID Symbol Window n **(0130_H+(n-1)*4)** **Application Reset Value: 0000 0000_H**



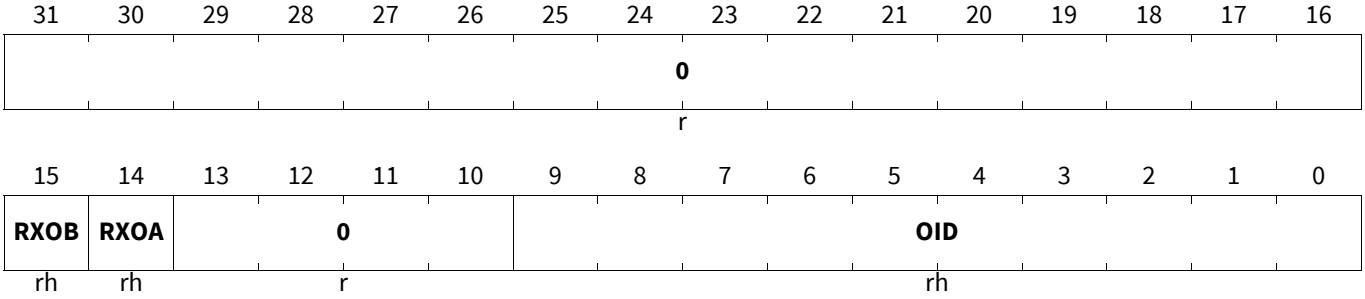
Field	Bits	Type	Description
EID	9:0	rh	**Even Sync ID(vsSyncIDListA,B even)** SYNC Frame ID even communication cycle.
RXEA	14	rh	**Received/Configured Even Sync ID on Channel A** Signals that a SYNC Frame corresponding to the stored even sync ID was received on channel A or that the node is configured to be a sync node with key slot = EID (ESID1 only). 0_B SYNC Frame not received on channel A / node configured to transmit SYNC Frames 1_B SYNC Frame received on channel A/ node not configured to transmit SYNC Frames
RXEB	15	rh	**Received/Configured Even Sync ID on Channel B** Signals that a SYNC Frame corresponding to the stored even sync ID was received on channel B or that the node is configured to be a sync node with key slot = EID (ESID1 only). 0_B SYNC Frame not received on channel B / node configured to transmit SYNC Frames 1_B SYNC Frame received on channel B / node not configured to transmit SYNC Frames
0	13:10, 31:16	r	**Reserved** Returns 0 if read; should be written with 0.

Odd Sync ID Symbol Window n

he Odd Sync ID nn (OSIDn, n=01-15) hold the Frame IDs of the SYNC Frames received in **odd** communication cycles, sorted in ascending order, with register OSID01 holding the lowest received SYNC Frame ID. If the node itself transmits a SYNC Frame in an odd communication cycle, register OSID01 holds the respective SYNC Frame ID as configured in Message Buffer 0 and flags RXOA, RXOB are set. The value is updated during the network idle time (NIT) of each odd communication cycle.

OSIDn (n=01-15)

Odd Sync ID Symbol Window n **(0170_H+(n-1)*4)** **Application Reset Value: 0000 0000_H**



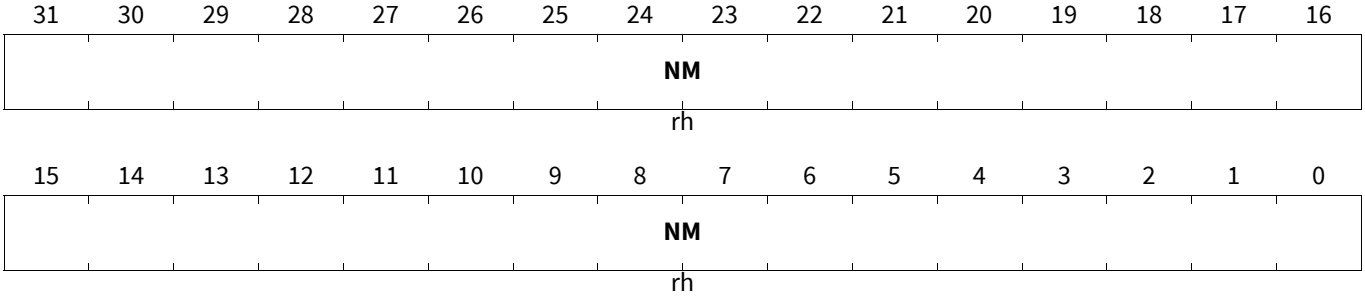
Field	Bits	Type	Description
OID	9:0	rh	**Odd Sync ID(vsSyncIDListA,B odd)** SYNC Frame ID even communication cycle.
RXOA	14	rh	**Received Odd Sync ID on Channel A** Signals that a SYNC Frame corresponding to the stored odd sync ID was received on channel A or that the node is configured to be a sync node with key slot = OID (OSID1 only). 0_B SYNC Frame not received on channel A/ node configured to transmit SYNC Frames 1_B SYNC Frame received on channel A/ node not configured to transmit SYNC Frames
RXOB	15	rh	**Received Odd Sync ID on Channel B** Signals that a SYNC Frame corresponding to the stored odd sync ID was received on channel B or that the node is configured to be a sync node with key slot = OID (OSID1 only) 0_B SYNC Frame not received on channel B/ node configured to transmit SYNC Frames 1_B SYNC Frame received on channel B/ node not configured to transmit SYNC Frames
0	13:10, 31:16	r	**Reserved** Returns 0 if read; should be written with 0.

Network Management Vector x

The three Network Management Vectors n (NMVx, x=1-3) registers hold the accrued Network Management (NM) vector (configurable 0 to 12 byte). The accrued Network Management (NM) vector is generated by the Communication Controller by bit-wise ORing each Network Management (NM) vector received (valid static Frames with PPI = 1) on each channel (see “Network Management”). The Communication Controller updates the Network Management (NM) vector at the end of each communication cycle as long as the Communication Controller is either in “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state. NMVx-bytes exceeding the configured Network Management (NM) vector length are not valid.

NMVx (x=1-3)

Network Management Vector x **($01B0_H$+(x-1)*4)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
NM	31:0	rh	**Network Management Vector**

Table below shows the assignment of the received payload's data byte to the Network Management vector.

Table 419 Assignment of data bytes to network management vector

Bit / Word	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NM1	Data3								Data2								Data1								Data0								
NM2	Data7								Data6								Data5								Data4								
NM3	Data11								Data10								Data9								Data8								

41.4.2.6 Message Buffer Control Registers

Message RAM Configuration

The Message RAM Configuration register defines the number of Message Buffers assigned to the static segment, dynamic segment, and FIFO. The register can be written during “DEFAULT_CONFIG” or “CONFIG” state only.

MRC

Message RAM Configuration (0300_H) Application Reset Value: $0180\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					SPLM	SEC		LCB							
r					rw	rw		rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FFB								FDB							
rw								rw							

Field	Bits	Type	Description
FDB	7:0	rw	**First Dynamic Buffer** May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 00_H No group of Message Buffers exclusively for the static segment configured 01_H …$7F_H$ Message Buffers 0 to FDB-1 reserved for static segment 80_H …FF_H No dynamic Message Buffers configured
FFB	15:8	rw	**First Buffer of FIFO** May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 00_H …$7E_H$ Message Buffers from FFB to LCB assigned to the FIFO $7F_H$ All Message Buffers assigned to the FIFO 80_H …FF_H No Message Buffers assigned to the FIFO
LCB	23:16	rw	**Last Configured Buffer** May be only modified in “DEFAULT_CONFIG” or “CONFIG” state. 01_H …$7F_H$ Number of Message Buffers is LCB + 1 80_H …FF_H No Message Buffer configured

Field	Bits	Type	Description
SEC	25:24	rw	**Secure Buffers** Not evaluated when the Communication Controller is in "DEFAULT_CONFIG" or "CONFIG" state. For temporary unlocking see "Host Handling of Errors". *Note: In nodes configured for SYNC Frame transmission or for single slot mode operation Message Buffer 0 (and if SPLM = 1, also Message Buffer 1) Reconfiguration of all Message Buffers is always locked* 00_B Reconfiguration of Message Buffers enabled with numbers < FFB enabled. 01_B Reconfiguration of Message Buffers with numbers < FDB and with numbers ≥ FFB locked and transmission of Message Buffers for static segment with numbers ≥ FDB disabled 10_B Reconfiguration of all Message Buffers locked 11_B Reconfiguration of all Message Buffers locked and transmission of Message Buffers for static segment with numbers ≥ FDB disabled
SPLM	26	rw	**SYNC Frame Payload Multiplex** This bit is only evaluated if the node is configured as sync node (SUCC1.TXSY = 1) or for single slot mode operation (SUCC1.TSM = 1). When this bit is set to 1 Message Buffers 0 and 1 are dedicated for SYNC Frame transmission with different payload data on channel A and B. When this bit is reset to 0, SYNC Frames are transmitted from Message Buffer 0 with the same payload data on both channels. Note that the channel filter configuration for Message Buffer 0 resp. Message Buffer 1 has to be chosen accordingly. 0_B Only Message Buffer 0 locked against reconfiguration 1_B Both Message Buffers 0 and 1 are locked against reconfiguration
0	31:27	r	**Reserved** Returns 0 if read; should be written with 0.

Note: In case the node is configured as sync node (SUCC1.TXSY = 1) or for single slot mode operation (SUCC1.TSM = 1), Message Buffer 0 resp. 1 is reserved for SYNC Frames or single slot Frames and have to be configured with the node-specific key slot ID. In case the node is neither configured as sync node nor for single slot operation Message Buffer 0 resp. 1 is treated like all other Message Buffers.

Table 420 Usage of the three Message Buffer Pointer

Message Buffer 0	⇓ Static Buffers		
Message Buffer 1			
…			
	⇓ Static + Dynamic	⇐ **FDB**	
			FIFO configured: **FFB** > **FDB**
	⇓ FIFO	⇐ **FFB**	No FIFO configured: **FFB** ≥ 128

Table 420 Usage of the three Message Buffer Pointer (cont'd)

Message Buffer N-1		**LCB ≥ FDB, LCB ≥ FFB**
Message Buffer N		⇐ **LCB**

The programmer has to ensure that the configuration defined by FDB, FFB, and LCB is valid. **The Communication Controller does not check for erroneous configurations!**

Note: The maximum number of Header Sections is 128. This means a maximum of 128 Message Buffer can be configured. The maximum length of a Data Section is 254 byte. The length of the Data Section may be configured differently for each Message Buffer. For details see "Message RAM". In case two or more Message Buffers are assigned to slot 1 by use of cycle filtering, all of them must be located either in the "Static Buffers" or at the beginning of the "Static + Dynamic Buffers" section. The payload length configured and the length of the Data Section need to be configured identically for all Message Buffers belonging to the FIFO via WRHS2.PLC and WRHS3.DP. When the Communication Controller is not in "DEFAULT_CONFIG" or "CONFIG" state reconfiguration of Message Buffers belonging to the FIFO is locked.

FIFO Rejection Filter

The FIFO Rejection Filter defines a user specified sequence of bits to which channel, Frame ID, and cycle count of the incoming Frames are compared. Together with the FIFO Rejection Filter Mask this register determines whether a message is rejected by the FIFO. The FRF register can be written during "DEFAULT_CONFIG" or "CONFIG" state only.

FRF

FIFO Rejection Filter **(0304_H)** **Application Reset Value: $0180\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							**RNF**	**RSS**	**CYF**						
r							rw	rw	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**FID**											**CH**	
r			rw											rw	

Field	Bits	Type	Description
CH	1:0	rw	**Channel Filter** May be modified in "DEFAULT_CONFIG" or "CONFIG" state only. 00_B receive on both channels[1)] 01_B receive only on channel B 10_B receive only on channel A 11_B no reception
FID	12:2	rw	**Frame ID Filter** Determines the Frame ID to be rejected by the FIFO. With the additional configuration of register FRFM, the corresponding Frame ID filter bits are ignored, which results in further rejected Frame IDs. When FRFM.MFID is zero, a Frame ID filter value of zero means that no Frame ID is rejected. 000_H …$7FF_H$ Frame ID filter values

Field	Bits	Type	Description
CYF	22:16	rw	**Cycle Counter Filter** The 7-bit cycle counter filter determines the cycle set to which Frame ID and channel rejection filter are applied. In cycles **not** belonging to the cycle set specified by CYF, all Frames are rejected. For details about the configuration of the cycle counter filter see “Cycle Counter Filtering”. May be modified in “DEFAULT_CONFIG” or “CONFIG” state only.
RSS	23	rw	**Reject in Static Segment** If this bit is set, the FIFO is used only be used in dynamic segment. May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B FIFO also used in static segment 1_B Reject messages for static segment
RNF	24	rw	**Reject NULL Frames** If this bit is set, received NULL Frames are not stored in the FIFO. May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. 0_B NULL Frames are stored in the FIFO 1_B Reject all NULL Frames
0	15:13, 31:25	r	**Reserved** Returns 0 if read; should be written with 0.

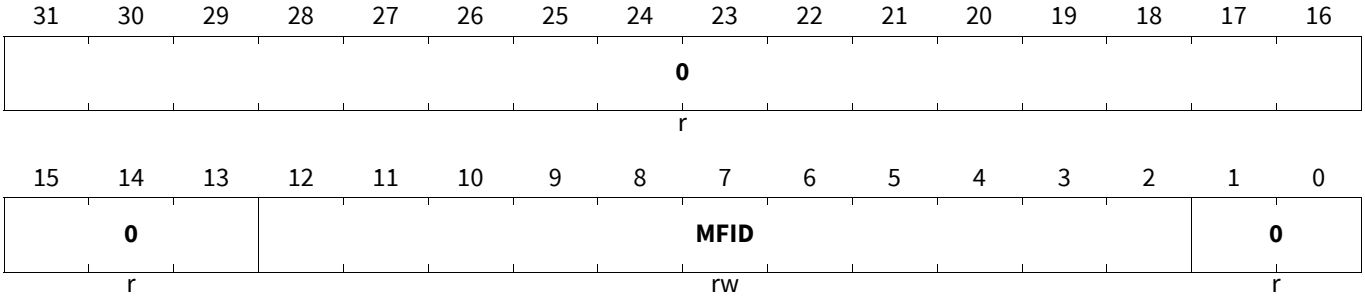
1) If reception on both channels is configured, also in static segment always both Frames (from channel A and B) are stored in the FIFO, even if they are identical.

FIFO Rejection Filter Mask

The FIFO Rejection Filter Mask specifies which of the corresponding Frame ID filter bits are relevant for rejection filtering. If a bit is set, it indicates that the corresponding bit in the FRF register will not be considered for rejection filtering. The FRFM register can be written during “DEFAULT_CONFIG” or “CONFIG” state only.

FRFM

FIFO Rejection Filter Mask **(0308_H)** **Application Reset Value: $0000\ 0000_H$**

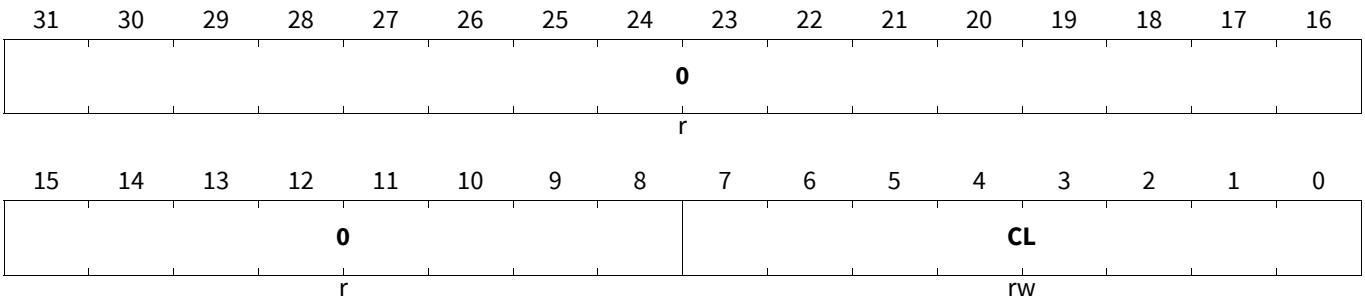


Field	Bits	Type	Description
MFID	12:2	rw	**Mask Frame ID Filter** May be modified in “DEFAULT_CONFIG” or “CONFIG” state only. When '0' written in a bit position, the corresponding Frame ID filter bit is used for rejection filtering. When '1' written in a bit position, the corresponding Frame ID filter bit is ignored. Valid values are from 0x000 - 0x3FF.
0	1:0, 31:13	r	**Reserved** Returns 0 if read; should be written with 0.

FIFO Critical Level

The Communication Controller accepts modifications of the register in “DEFAULT_CONFIG” or “CONFIG” state only.

FCL
FIFO Critical Level **($030C_H$)** **Application Reset Value: $0000\ 0080_H$**



Field	Bits	Type	Description
CL	7:0	rw	**Critical Level** When the receive FIFO fill level FSR.RFFL is equal or greater than the critical level configured by CL, the receive FIFO critical level flag FSR.RFCL is set. If CL is programmed to values > 128, bit FSR.RFCL is never set. When FSR.RFCL changes from 0 to 1 bit SIR.RFCL is set to 1, and if enabled, a service request is generated.
0	31:8	r	**Reserved** Returns 0 if read; should be written with 0.

41.4.2.7 Message Buffer Status Registers

Message Handler Status

The Message Handler Status register gives the Host access to the current state of the Message Handler. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag. An application reset will also clear the register. If one of the flags MHDS.EIBF, MHDS.EOBF, MHDS.EMR, MHDS.ETBF1, MHDS.ETBF2 changes from 0 to 1 EIR.EERR is set.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register.

MHDS

Message Handler Status **(0310_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				**MBU**				**0**	**MBT**						
r				rh				r	rh						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**FMB**				**CRAM**	**MFMB**	**FMBD**	**ETBF2**	**ETBF1**	**EMR**	**EOBF**	**EIBF**
r				rh				rh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
EIBF	0	rwh	**ECC Error Input Buffer RAM 1,2** 0_B No error 1_B Error occurred when reading Input Buffer RAM 1 or Input Buffer RAM 2
EOBF	1	rwh	**ECC Error Output Buffer RAM 1,2** 0_B No error 1_B Error occurred when reading Output Buffer RAM 1 or Output Buffer RAM 2
EMR	2	rwh	**ECC Error Message RAM** 0_B No error 1_B Error occurred when reading the Message RAM
ETBF1	3	rwh	**ECC Error Transient Buffer RAM A** 0_B No error 1_B Error occurred when reading Transient Buffer RAM A
ETBF2	4	rwh	**ECC Error Transient Buffer RAM B** 0_B No error 1_B Error occurred when reading Transient Buffer RAM B
FMBD	5	rwh	**Faulty Message Buffer Detected** 0_B No faulty Message Buffer 1_B Message Buffer referenced by MHDS.FMB holds faulty data due to a ECC error

Field	Bits	Type	Description
MFMB	6	rwh	**Multiple Faulty Message Buffers detected** 0_B No additional faulty Message Buffer 1_B Another faulty Message Buffer was detected while flag MHDS.FMBD is set
CRAM	7	rh	**Clear all internal RAM's** Signals that execution of the CHI command CLEAR_RAMS is ongoing (all bits of all internal RAM blocks are written to 0). The bit is set by CHI command CLEAR_RAMS. 0_B No execution of the CHI command CLEAR_RAMS 1_B Execution of the CHI command CLEAR_RAMS ongoing
FMB	14:8	rh	**Faulty Message Buffer** ECC error occurred when reading from the Message Buffer or when transferring data from Input Buffer or Transient Buffer A or Transient Buffer B to the Message Buffer referenced by MHDS.FMB. Value only valid when one of the flags MHDS.EIBF, MHDS.EMR, MHDS.ETBF1, MHDS.ETBF2, and flag MHDS.FMBD is set. Updated only after the Host has reset flag MHDS.FMBD.
MBT	22:16	rh	**Message Buffer Transmitted** Number of last successfully transmitted Message Buffer. If the Message Buffer is configured for single-shot mode, the respective TXR flag in the Transmission Request Registers TXRQ1 to TXRQ4 was reset. MBT is reset when the Communication Controller leaves "CONFIG" state or enters "STARTUP" state.
MBU	30:24	rh	**Message Buffer Updated** Number of Message Buffer that was updated last. For this Message Buffer the respective NDn (n = 0-31) to NDn (n = 96-127)and / or MBCn (n = 0-31) to MBCn (n = 96-127) flag in the New Data Registers NDAT1 to NDAT4 and the Message Buffer Status Changed MBSC1 to MBSC4 registers are also set. MBU is reset when the Communication Controller leaves "CONFIG" state or enters "STARTUP" state.
0	15, 23, 31	r	**Reserved** Returns 0 if read; should be written with 0.

Last Dynamic Transmit Slot

The Last Dynamic Transmit Slot Register stores the Slot Counter value at the time of the last Frame transmission in the dynamic segment. This register is reset when the Communication Controller leaves "CONFIG" state or enters "STARTUP" state.

LDTS

Last Dynamic Transmit Slot **(0314_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					**LDTB**										
r					rh										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					**LDTA**										
r					rh										

Field	Bits	Type	Description
LDTA	10:0	rh	**Last Dynamic Transmission Channel A** Value of (vSlotCounter[A]) at the time of the last Frame transmission on channel A in the dynamic segment of this node. It is updated at the end of the dynamic segment and is reset to zero if no Frame was transmitted during the dynamic segment.
LDTB	26:16	rh	**Last Dynamic Transmission Channel B** Value of (vSlotCounter[B]) at the time of the last Frame transmission on channel B in the dynamic segment of this node. It is updated at the end of the dynamic segment and is reset to zero if no Frame was transmitted during the dynamic segment.
0	15:11, 31:27	r	**Reserved** Returns 0 if read; should be written with 0.

FIFO Status Register

The register is reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state.

FSR

FIFO Status Register **(0318_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFFL								**0**					**RFO**	**RFCL**	**RFNE**
rh								r					rh	rh	rh

Field	Bits	Type	Description
RFNE	0	rh	**Receive FIFO Not Empty** This flag is set by the Communication Controller when a received valid Frame (data or NULL Frame depending on rejection mask) was stored in the FIFO. In addition, service request flag SIR.RFNE is set. The bit is reset after the Host has read all message from the FIFO. 0_B Receive FIFO is empty 1_B Receive FIFO is not empty
RFCL	1	rh	**Receive FIFO Critical Level** This flag is set when the receive FIFO fill level RFFL is equal or greater than the critical level as configured by FCL.CL. The flag is cleared by the Communication Controller as soon as RFFL drops below FCL.CL. When RFCL changes from 0 to 1 bit SIR.RFCL is set to 1, and if enabled, an service request is generated. 0_B Receive FIFO below critical level 1_B Receive FIFO critical level reached
RFO	2	rh	**Receive FIFO Overrun** The flag is set by the Communication Controller when a receive FIFO overrun is detected. When a receive FIFO overrun occurs, the oldest message is overwritten with the actual received message. In addition, service request flag EIR.RFO is set.The flag is cleared by the next FIFO read access issued by the Host. 0_B No receive FIFO overrun detected 1_B A receive FIFO overrun has been detected
RFFL	15:8	rh	**Receive FIFO Fill Level** Number of FIFO buffers filled up with new data not yet read by the Host. Maximum value is 128.
0	7:3, 31:16	r	**Reserved** Returns 0 if read; should be written with 0.

Message Handler Constraints Flags

Some constraints exist for the Message Handler regarding f_{CLC_ERAY} frequency, Message RAM configuration, and FlexRay™ bus traffic. To simplify software development, constraints violations are reported by setting flags in the MHDF. The register is reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state. A flag is cleared by setting the corresponding bit position. Clearing has no effect on the flag. Software write of '1' when a flag is already cleared has no effect. If any flag in MHDFL is set, interrupt flag EIR.MHF is set.

Note: LDMST or SWAPMSK.W instructions should be used only with bit mask enabled for all ´rwh´ bits in this register.

MHDF

Message Handler Constraints Flags **(031C$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							WAHP	TNSB	TNSA	TBFB	TBFA	FNFB	FNFA	SNUB	SNUA
r							rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SNUA	0	rwh	**Status Not Updated Channel A** This flag is set by the Communication Controller when the Message Handler, due to overload condition, was not able to update a Message Buffer's status MBS with respect to channel A. 0_B No overload condition occurred when updating MBS for channel A 1_B MBS for channel A not updated
SNUB	1	rwh	**Status Not Updated Channel B** This flag is set by the Communication Controller when the Message Handler, due to overload condition, was not able to update a Message Buffer's status MBS with respect to channel B. 0_B No overload condition occurred when updating MBS for channel B 1_B MBS for channel B not updated
FNFA	2	rwh	**Find Sequence Not Finished Channel A** This flag is set by the Communication Controller when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching Message Buffer) with respect to channel A. 0_B No find sequence not finished for channel A 1_B Find sequence not finished for channel A
FNFB	3	rwh	**Find Sequence Not Finished Channel B** This flag is set by the Communication Controller when the Message Handler, due to overload condition, was not able to finish a find sequence (scan of Message RAM for matching Message Buffer) with respect to channel B. 0_B No find sequence not finished for channel B 1_B Find sequence not finished for channel B
TBFA	4	rwh	**Transient Buffer Access Failure A** This flag is set by the Communication Controller when a read or write access to Transient Buffer A requested by PRT A could not complete within the available time. 0_B No TBF A access failure 1_B TBF A access failure

Field	Bits	Type	Description
TBFB	5	rwh	**Transient Buffer Access Failure B** This flag is set by the Communication Controller when a read or write access to Transient Buffer B requested by PRT B could not complete within the available time. 0_B No Transient Buffer B access failure 1_B Transient Buffer B access failure
TNSA	6	rwh	**Transmission Not Started Channel A** This flag is set by the CC when the Message Handler was not ready to start a scheduled transmission on channel A at the action point of the configured slot. 0_B No transmission not started on channel A 1_B Transmission not started on channel A
TNSB	7	rwh	**Transmission Not Started Channel B** This flag is set by the CC when the Message Handler was not ready to start a scheduled transmission on channel B at the action point of the configured slot. 0_B No transmission not started on channel B 1_B Transmission not started on channel B
WAHP	8	rwh	**Write Attempt to Header Partition** Outside “DEFAULT_CONFIG” and “CONFIG” state this flag is set by the Communication Controller when the message handler tries to write message data into the Header Partition of the Message RAM due to faulty configuration of a Message Buffer. The write attempt is not executed, to protect the Header Partition from unintended write accesses. 0_B No write attempt to Header Partition 1_B Write attempt to Header Partition
0	31:9	r	**Reserved** Returns 0 if read; should be written with 0.

Transmission Request Register 1

This register reflect the state of the TXR flags of the configured Message Buffers 0 to 31. The flags are evaluated for transmit buffers only. If the number of configured Message Buffers is less than 31, the remaining TXRn flags have no meaning and are read as 0.

TXRQ1
Transmission Request Register 1 **(0320_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXR31	**TXR30**	**TXR29**	**TXR28**	**TXR27**	**TXR26**	**TXR25**	**TXR24**	**TXR23**	**TXR22**	**TXR21**	**TXR20**	**TXR19**	**TXR18**	**TXR17**	**TXR16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXR15	**TXR14**	**TXR13**	**TXR12**	**TXR11**	**TXR10**	**TXR9**	**TXR8**	**TXR7**	**TXR6**	**TXR5**	**TXR4**	**TXR3**	**TXR2**	**TXR1**	**TXR0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TXRn (n=0-31)	n	rh	**Transmission Request n (n = 0-31)** If the flag is set, the respective Message Buffer 0 to 31 is ready for transmission respectively transmission of this Message Buffer is in progress. In single-shot mode the flags are reset after transmission has completed.

Transmission Request Register 2

This register reflect the state of the TXR flags of the configured Message Buffers 31 to 63. The flags are evaluated for transmit buffers only. If the number of configured Message Buffers is less than 63, the remaining TXRn flags have no meaning and are read as 0.

TXRQ2
Transmission Request Register 2 **(0324_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXR63	**TXR62**	**TXR61**	**TXR60**	**TXR59**	**TXR58**	**TXR57**	**TXR56**	**TXR55**	**TXR54**	**TXR53**	**TXR52**	**TXR51**	**TXR50**	**TXR49**	**TXR48**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXR47	**TXR46**	**TXR45**	**TXR44**	**TXR43**	**TXR42**	**TXR41**	**TXR40**	**TXR39**	**TXR38**	**TXR37**	**TXR36**	**TXR35**	**TXR34**	**TXR33**	**TXR32**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TXRn (n=32-63)	n-32	rh	**Transmission Request n (n = 32-63)** If the flag is set, the respective Message Buffer 32 to 63 is ready for transmission respectively transmission of this Message Buffer is in progress. In single-shot mode the flags are reset after transmission has completed.

Transmission Request Register 3

This register reflect the state of the TXR flags of the configured Message Buffers 64 to 95. The flags are evaluated for transmit buffers only. If the number of configured Message Buffers is less than 95, the remaining TXRn flags have no meaning and are read as 0.

TXRQ3
Transmission Request Register 3 **(0328_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXR95	**TXR94**	**TXR93**	**TXR92**	**TXR91**	**TXR90**	**TXR89**	**TXR88**	**TXR87**	**TXR86**	**TXR85**	**TXR84**	**TXR83**	**TXR82**	**TXR81**	**TXR80**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXR79	**TXR78**	**TXR77**	**TXR76**	**TXR75**	**TXR74**	**TXR73**	**TXR72**	**TXR71**	**TXR70**	**TXR69**	**TXR68**	**TXR67**	**TXR66**	**TXR65**	**TXR64**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TXRn (n=64-95)	n-64	rh	**Transmission Request n (n = 64-95)** If the flag is set, the respective Message Buffer 64 to 95 is ready for transmission respectively transmission of this Message Buffer is in progress. In single-shot mode the flags are reset after transmission has completed.

Transmission Request Register 4

This register reflect the state of the TXR flags of the configured Message Buffers 96 to 127. The flags are evaluated for transmit buffers only. If the number of configured Message Buffers is less than 127, the remaining TXRn flags have no meaning and are read as 0.

TXRQ4
Transmission Request Register 4 **($032C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXR127	**TXR126**	**TXR125**	**TXR124**	**TXR123**	**TXR122**	**TXR121**	**TXR120**	**TXR119**	**TXR118**	**TXR117**	**TXR116**	**TXR115**	**TXR114**	**TXR113**	**TXR112**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXR111	**TXR110**	**TXR109**	**TXR108**	**TXR107**	**TXR106**	**TXR105**	**TXR104**	**TXR103**	**TXR102**	**TXR101**	**TXR100**	**TXR99**	**TXR98**	**TXR97**	**TXR96**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TXRn (n=96-127)	n-96	rh	**Transmission Request n (n = 96-127)** If the flag is set, the respective Message Buffer 96 to 127 is ready for transmission respectively transmission of this Message Buffer is in progress. In single-shot mode the flags are reset after transmission has completed.

New Data Register 1

This register reflect the state of the ND flags of all configured Message Buffers 0 to 31. ND flags assigned to transmit buffers are meaningless. If the number of configured Message Buffers is less than 31, the remaining NDn flags have no meaning. The registers are reset when the Communication Controller leaves “CONFIG” state or enters “STARTUP” state.

NDAT1

New Data Register 1 **(0330_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND31	**ND30**	**ND29**	**ND28**	**ND27**	**ND26**	**ND25**	**ND24**	**ND23**	**ND22**	**ND21**	**ND20**	**ND19**	**ND18**	**ND17**	**ND16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND15	**ND14**	**ND13**	**ND12**	**ND11**	**ND10**	**ND9**	**ND8**	**ND7**	**ND6**	**ND5**	**ND4**	**ND3**	**ND2**	**ND1**	**ND0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NDn (n=0-31)	n	rh	**New Data n (n = 0-31)** The flags are set when a valid received Data Frame matches the Message Buffer's filter configuration, independent of the payload length received or the payload length configured for that Message Buffer. The flags are not set after reception of NULL Frames except for Message Buffers belonging to the receive FIFO. An ND flag is reset when the Header Section of the corresponding Message Buffer is reconfigured or when the Data Section has been transferred to the Output Buffer.

New Data Register 2

This register reflect the state of the ND flags of all configured Message Buffers 32 to 63. ND flags assigned to transmit buffers are meaningless. If the number of configured Message Buffers is less than 63, the remaining NDn flags have no meaning. The registers are reset when the Communication Controller leaves "CONFIG" state or enters "STARTUP" state.

NDAT2

New Data Register 2 **(0334_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND63	**ND62**	**ND61**	**ND60**	**ND59**	**ND58**	**ND57**	**ND56**	**ND55**	**ND54**	**ND53**	**ND52**	**ND51**	**ND50**	**ND49**	**ND48**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND47	**ND46**	**ND45**	**ND44**	**ND43**	**ND42**	**ND41**	**ND40**	**ND39**	**ND38**	**ND37**	**ND36**	**ND35**	**ND34**	**ND33**	**ND32**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NDn (n=32-63)	n-32	rh	**New Data n (n = 32-63)** The flags are set when a valid received Data Frame matches the Message Buffer's filter configuration, independent of the payload length received or the payload length configured for that Message Buffer. The flags are not set after reception of NULL Frames except for Message Buffers belonging to the receive FIFO. An ND flag is reset when the Header Section of the corresponding Message Buffer is reconfigured or when the Data Section has been transferred to the Output Buffer.

New Data Register 3

This register reflect the state of the ND flags of all configured Message Buffers 64 to 95. ND flags assigned to transmit buffers are meaningless. If the number of configured Message Buffers is less than 95, the remaining NDn flags have no meaning. The registers are reset when the Communication Controller leaves "CONFIG" state or enters "STARTUP" state.

NDAT3
New Data Register 3 **(0338_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND95	**ND94**	**ND93**	**ND92**	**ND91**	**ND90**	**ND89**	**ND88**	**ND87**	**ND86**	**ND85**	**ND84**	**ND83**	**ND82**	**ND81**	**ND80**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND79	**ND78**	**ND77**	**ND76**	**ND75**	**ND74**	**ND73**	**ND72**	**ND71**	**ND70**	**ND69**	**ND68**	**ND67**	**ND66**	**ND65**	**ND64**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NDn (n=64-95)	n-64	rh	**New Data n (n = 64-95)** The flags are set when a valid received Data Frame matches the Message Buffer's filter configuration, independent of the payload length received or the payload length configured for that Message Buffer. The flags are not set after reception of NULL Frames except for Message Buffers belonging to the receive FIFO. An ND flag is reset when the Header Section of the corresponding Message Buffer is reconfigured or when the Data Section has been transferred to the Output Buffer.

New Data Register 4

This register reflect the state of the ND flags of all configured Message Buffers 96 to 127. ND flags assigned to transmit buffers are meaningless. If the number of configured Message Buffers is less than 127, the remaining NDn flags have no meaning. The registers are reset when the Communication Controller leaves "CONFIG" state or enters "STARTUP" state.

NDAT4

New Data Register 4 **(033C$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND127	**ND126**	**ND125**	**ND124**	**ND123**	**ND122**	**ND121**	**ND120**	**ND119**	**ND118**	**ND117**	**ND116**	**ND115**	**ND114**	**ND113**	**ND112**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND111	**ND110**	**ND109**	**ND108**	**ND107**	**ND106**	**ND105**	**ND104**	**ND103**	**ND102**	**ND101**	**ND100**	**ND99**	**ND98**	**ND97**	**ND96**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NDn (n=96-127)	n-96	rh	**New Data n (n = 96-127)** The flags are set when a valid received Data Frame matches the Message Buffer's filter configuration, independent of the payload length received or the payload length configured for that Message Buffer. The flags are not set after reception of NULL Frames except for Message Buffers belonging to the receive FIFO. An ND flag is reset when the Header Section of the corresponding Message Buffer is reconfigured or when the Data Section has been transferred to the Output Buffer.

Message Buffer Status Changed 1

This register reflect the state of the MBC flags of all configured Message Buffers. If the number of configured Message Buffers is less than 31, the remaining MBCn flags have no meaning. The register is reset when the communication controller leaves "CONFIG" state or enters "STARTUP" state.

MBSC1

Message Buffer Status Changed 1 **(0340$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MBC31	**MBC30**	**MBC29**	**MBC28**	**MBC27**	**MBC26**	**MBC25**	**MBC24**	**MBC23**	**MBC22**	**MBC21**	**MBC20**	**MBC19**	**MBC18**	**MBC17**	**MBC16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBC15	**MBC14**	**MBC13**	**MBC12**	**MBC11**	**MBC10**	**MBC9**	**MBC8**	**MBC7**	**MBC6**	**MBC5**	**MBC4**	**MBC3**	**MBC2**	**MBC1**	**MBC0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
MBCn (n=0-31)	n	rh	**Message Buffer Status Changed n (n = 0-31)** An MBC flags is set whenever the Message Handler changes on of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the Header Section (see “Message Buffer Status”) of the respective Message Buffer 0 to Message Buffer 31. The flags are reset when the Header Section of the Message Buffer is reconfigured or when it has been transferred to the Output Buffer.

Message Buffer Status Changed 2

This register reflect the state of the MBC flags of all configured Message Buffers. If the number of configured Message Buffers is less than 63, the remaining MBCn flags have no meaning. The register is reset when the communication controller leaves “CONFIG” state or enters “STARTUP” state.

MBSC2

Message Buffer Status Changed 2 **(0344_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MBC63	**MBC62**	**MBC61**	**MBC60**	**MBC59**	**MBC58**	**MBC57**	**MBC56**	**MBC55**	**MBC54**	**MBC53**	**MBC52**	**MBC51**	**MBC50**	**MBC49**	**MBC48**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBC47	**MBC46**	**MBC45**	**MBC44**	**MBC43**	**MBC42**	**MBC41**	**MBC40**	**MBC39**	**MBC38**	**MBC37**	**MBC36**	**MBC35**	**MBC34**	**MBC33**	**MBC32**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
MBCn (n=32-63)	n-32	rh	**Message Buffer Status Changed n (n = 32-63)** An MBC flags is set whenever the Message Handler changes on of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the Header Section (see “Message Buffer Status”) of the respective Message Buffer 32 to Message Buffer 63. The flags are reset when the Header Section of the Message Buffer is reconfigured or when it has been transferred to the Output Buffer.

Message Buffer Status Changed 3

This register reflect the state of the MBC flags of all configured Message Buffers. If the number of configured Message Buffers is less than 95, the remaining MBCn flags have no meaning. The register is reset when the communication controller leaves “CONFIG” state or enters “STARTUP” state.

MBSC3

Message Buffer Status Changed 3 **(0348_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MBC95	MBC94	MBC93	MBC92	MBC91	MBC90	MBC89	MBC88	MBC87	MBC86	MBC85	MBC84	MBC83	MBC82	MBC81	MBC80
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBC79	MBC78	MBC77	MBC76	MBC75	MBC74	MBC73	MBC72	MBC71	MBC70	MBC69	MBC68	MBC67	MBC66	MBC65	MBC64
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
MBCn (n=64-95)	n-64	rh	**Message Buffer Status Changed n (n = 64-95)** An MBC flags is set whenever the Message Handler changes on of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the Header Section (see “Message Buffer Status”) of the respective Message Buffer 64 to Message Buffer 95. The flags are reset when the Header Section of the Message Buffer is reconfigured or when it has been transferred to the Output Buffer.

Message Buffer Status Changed 4

This register reflect the state of the MBC flags of all configured Message Buffers. If the number of configured Message Buffers is less than 127, the remaining MBCn flags have no meaning. The register is reset when the communication controller leaves “CONFIG” state or enters “STARTUP” state.

MBSC4

Message Buffer Status Changed 4 **($034C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MBC127	MBC126	MBC125	MBC124	MBC123	MBC122	MBC121	MBC120	MBC119	MBC118	MBC117	MBC116	MBC115	MBC114	MBC113	MBC112
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MBC111	MBC110	MBC109	MBC108	MBC107	MBC106	MBC105	MBC104	MBC103	MBC102	MBC101	MBC100	MBC99	MBC98	MBC97	MBC96
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

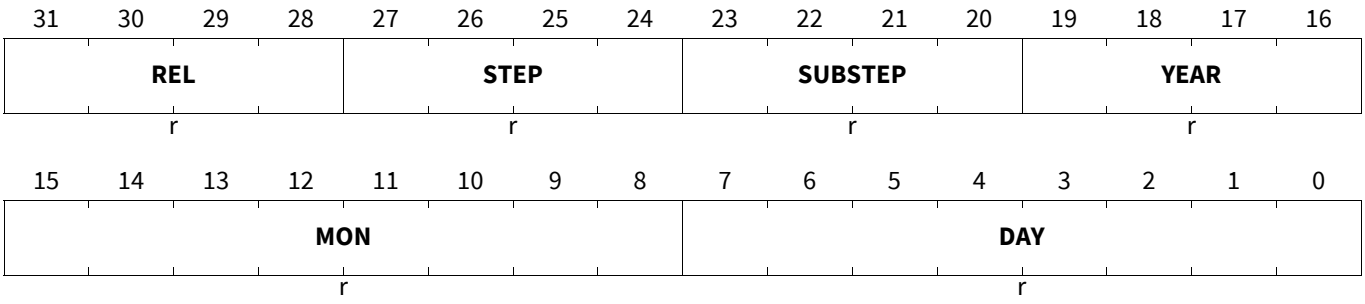
Field	Bits	Type	Description
MBCn (n=96-127)	n-96	rh	**Message Buffer Status Changed n (n = 96-127)** An MBC flags is set whenever the Message Handler changes on of the status flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB in the Header Section (see “Message Buffer Status”) of the respective Message Buffer 96 to Message Buffer 127. The flags are reset when the Header Section of the Message Buffer is reconfigured or when it has been transferred to the Output Buffer.

41.4.2.8 Identification Registers

Core Release Register

This register contains bit fields about the ERAY module identification. It is read only.

CREL
Core Release Register **($03F0_H$)** **Application Reset Value: $XXXX\ XXXX_H$**



Field	Bits	Type	Description
DAY	7:0	r	**Design Time Stamp, Day** Two digits, BCD-coded.
MON	15:8	r	**Design Time Stamp, Month** Two digits, BCD-coded.
YEAR	19:16	r	**Design Time Stamp, Year** One digit, BCD-coded.
SUBSTEP	23:20	r	**Sub-Step of Core Release** One digits, BCD-coded. 0_H Alpha, pre-Beta, pre-Beta-update, pre-Beta2, pre-Beta2-update, Beta, Beta2, Revision 1.0.0 1_H Beta_ct, Beta-ct-fix1, Revision 1.0.1 2_H Revision1.0RC1,Beta-ct-fix2, REVISION 1.0RC1
STEP	27:24	r	**Step of Core Release** One digits, BCD-coded. 0_H Revision 1.0.0 1_H Alpha 2_H pre-Beta 3_H pre-Beta-update 4_H pre-Beta2 5_H pre-Beta2-update 6_H Beta 7_H Beta2
REL	31:28	r	**Core Release** One digit, BCD-coded. 0_H alpha…beta2ct 1_H Revision 1.0

Table 421 Coding of releases

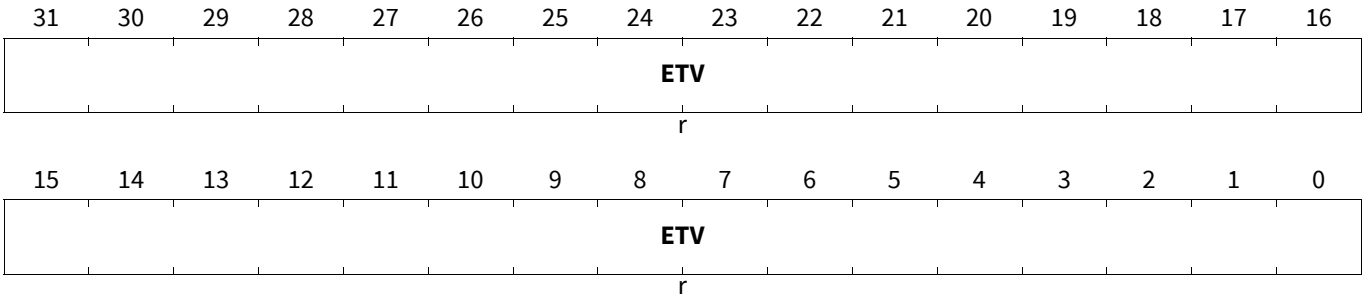
Release	Step	Sub-Step	Name	Release Date
0	1	0	Alpha	
0	2	0	pre-Beta	
0	3	0	pre-Beta-update	
0	4	0	pre-Beta2	
0	5	0	pre-Beta2-update	
0	6	0	Beta	
0	6	1	Beta-ct-fix1	14.10.2005
0	6	2	Beta-ct-fix2	14.12.2005
0	7	0	Beta2	03.02.2006
0	7	1	Beta2ct	24.03.2006
0	7	2	Revision 1.0RC1	07.04.2006
1	0	0	Release 1.0.0	19.05.2006
1	0	1	Release 1.0.1	2006
1	0	2	Release 1.0.2	31.10.2007

Endian Register

This register may be used to check, if the data of the E-Ray is handled by a host with the correct endian format. It is read only.

ENDN

Endian Register **(03F4$_H$)** **Application Reset Value: 8765 4321$_H$**



Field	Bits	Type	Description
ETV	31:0	r	**Endianness Test Value** The endianness test value.

41.4.2.9 Input Buffer

Double buffer structure consisting of Input Buffer Host and Input Buffer Shadow. While the Host can write to Input Buffer Host, the transfer to the Message RAM is done from Input Buffer Shadow. The Input Buffer holds the Header and Data Sections to be transferred to the selected Message Buffer in the Message RAM. It is used to configure the Message Buffers in the Message RAM and to update the Data Sections of transmit buffers.

When updating the Header Section of a Message Buffer in the Message RAM from the Input Buffer, the Message Buffer Status as described in “Message Buffer Status” is automatically reset to zero.

The Header Sections of Message Buffers belonging to the receive FIFO can only be (re)configured when the Communication Controller is in “DEFAULT_CONFIG” or “CONFIG” state. For those Message Buffers only the payload length configured and the data pointer need to be configured via WRHS2.PLC and WRHS2.DP. All information required for acceptance filtering is taken from the FIFO rejection filter and the FIFO rejection filter mask.

The data transfer between Input Buffer (IBF) and Message RAM is described in detail in “Data Transfer from Input Buffer to Message RAM”.

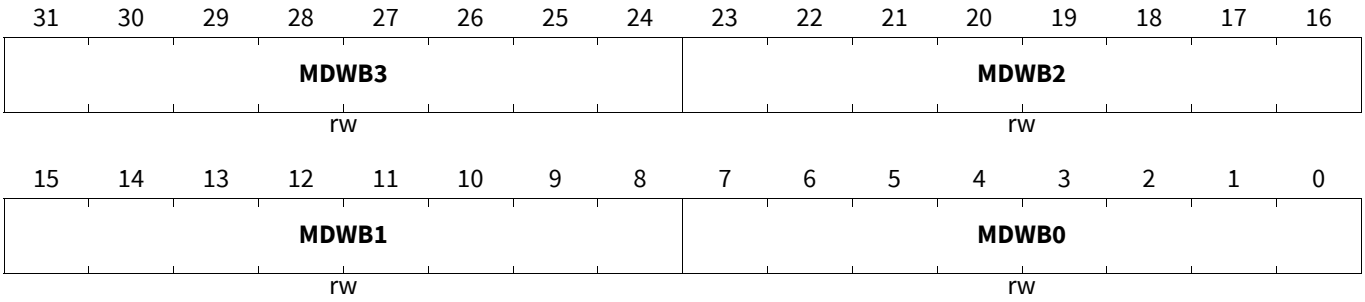
Write Data Section n

The Write Data Section (WRDSn, n = 01-64) holds the data words to be transferred to the Data Section of the addressed Message Buffer. The data words (DW_n) are written to the Message RAM in transmission order from DW1 (byte0, byte1) to DW_{PL} (PL = number of data words as defined by the payload length configured by WRHS2.PLC).

Notes

1. *16-bit Word 127 is located on WRDS64.MDW. In this case WRDS64.MDW is unused (no valid data). The Input Buffer RAMs are initialized to zero by CHI command CLEAR_RAMS.*
2. *When writing to the WRDSnn (nn = 01-64), each 32-bit word has to be filled up by one 32-bit access OR two consecutive 16-bit accesses OR four consecutive 8-bit accesses before the transfer from the Input Buffer to the Message RAM is started by writing the number of the target Message Buffer in the Message RAM to the Input Buffer Command Request register. If a 32-bit word of the Input Buffer has been filled with less then two consecutive 16-bit accesses OR four consecutive 8-bit accesses (less then 32-bit), random data is transferred into the Input buffer for every not written 16-bit or 8-bit of a 32-bit word.*

WRDSn (n=01-64)
Write Data Section n **(0400_H+(n-1)*4)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
MDWB0	7:0	rw	**32-Bit Word nn, Byte 0**
MDWB1	15:8	rw	**32-Bit Word nn, Byte 1**
MDWB2	23:16	rw	**32-Bit Word nn, Byte 2**
MDWB3	31:24	rw	**32-Bit Word nn, Byte 3**

Write Header Section 1

WRHS1

Write Header Section 1 **(0500_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		MBI	TXM	PPIT	CFG	CHB	CHA	0	CYC						
r		rw	rw	rw	rw	rw	rw	r	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					FID										
r					rw										

Field	Bits	Type	Description
FID	10:0	rw	**Frame ID** Frame ID of the selected Message Buffer. The Frame ID defines the slot number for transmission / reception of the respective message. Message Buffers with Frame ID = 0 are considered as not valid.
CYC	22:16	rw	**Cycle Code** The 7-bit cycle code determines the cycle set used for cycle counter filtering. For details about the configuration of the cycle code see Section "Cycle Counter Filtering".
CHA	24	rw	**Channel Filter Control A** The channel filtering field A associated with the buffer serves of channel A as a filter for receive buffers, and as a control field for transmit buffers
CHB	25	rw	**Channel Filter Control B** The channel filtering field B associated with the buffer serves of channel B as a filter for receive buffers, and as a control field for transmit buffers
CFG	26	rw	**Message Buffer Direction Configuration Bit** This bit is used to configure the corresponding buffer as a transmit buffer or as a receive buffer. For Message Buffers belonging to the receive FIFO the bit is not evaluated. 0_B The corresponding buffer is configured as Receive Buffer 1_B The corresponding buffer is configured as Transmit Buffer
PPIT	27	rw	**Payload Preamble Indicator Transmit** This bit is used to control the state of the Payload Preamble Indicator in transmit Frames. If the bit is set in a static Message Buffer, the respective Message Buffer holds Network Management information. If the bit is set in a dynamic Message Buffer the first two byte of the Payload Segment may be used for message ID filtering by the receiver. Message ID filtering of received FlexRay™ Frames is not supported by the E-Ray module, but can be done by the Host. 0_B Payload Preamble Indicator not set 1_B Payload Preamble Indicator set

Field	Bits	Type	Description
TXM	28	rw	**Transmission Mode** This bit is used to select the transmission mode (see “Transmit Buffers”). 0_B Continuous mode 1_B Single-shot mode
MBI	29	rw	**Message Buffer Service Request** This bit enables the receive / transmit service request for the corresponding Message Buffer. After a dedicated receive buffer has been updated by the Message Handler, flag SIR.RXI and /or SIR.MBSI in the Status Service Request register are set. After a transmission has completed flag SIR.TXI is set. 0_B The corresponding Message Buffer service request is disabled 1_B The corresponding Message Buffer service request is enabled
0	15:11, 23, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

Note: The Input Buffer RAMs are initialized to zero by CHI command CLEAR_RAMS. Note that only the currently active IBF bank is cleared. To clear the 2nd bank as well, CUST1.IBF1PAG and CUST1.IBF2PAG need to be set and command CLEAR_RAMS need to be issued again. This is required in particular after an application reset. If the 2nd bank of IBF is left unused, this procedure is not required.

Table 422 Channel Filter Control Bits

CHA	CHB	Transmit Buffertransmit Frame on	Receive Bufferstore Frame received from
1[1)]	1[1)]	Both Channels (static segment only)	Channel A or B (store first semantically valid Frame, static segment only)
1	0	Channel A	Channel A
0	1	Channel B	Channel B
0	0	No Transmission	Ignore Frame

Note: If a Message Buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no Frames are transmitted resp. received Frames are ignored (same function as CHA = CHB = 0)

Write Header Section 2

WRHS2
Write Header Section 2 **(0504_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0									PLC						
r									rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					CRC										
r					rw										

Field	Bits	Type	Description
CRC	10:0	rw	**Header CRC(vRF!Header!HeaderCRC)** Receive Buffer: Configuration not required Transmit Buffer: Header CRC calculated and configured by the Host. For calculation of the Header CRC the payload length of the Frame send on the bus has to be considered. In static segment the payload length of all Frames is configured by MHDC.SFDL.
PLC	22:16	rw	**Payload Length Configured** Length of Data Section (number of 2-byte words) as configured by the Host. During static segment the static Frame payload length as configured by MHDC.SFDL in the MHD Configuration Register defines the payload length for all static Frames. If the payload length configured by PLC is shorter than this value padding byte are inserted to ensure that Frames have proper physical length. The padding pattern is logical zero.
0	15:11, 31:23	r	**Reserved** Returns 0 if read; should be written with 0.

Write Header Section 3

WRHS3
Write Header Section 3 **(0508$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					DP										
r					rw										

Field	Bits	Type	Description
DP	10:0	rw	**Data Pointer** Pointer to the first 32-bit word of the Data Section of the addressed Message Buffer in the Message RAM.
0	31:11	r	**Reserved** Returns 0 if read; should be written with 0.

Input Buffer Command Mask

Configures how the Message Buffer in the Message RAM selected by the Input Buffer Command Request register IBCR is updated. If IBF Host and IBF Shadow are swapped, also masked bits IBCM.LHSH, IBCM.LDSH, and IBCM.STXRH are swapped with bits IBCM.LHSS, IBCM.LDSS, and IBCM.STXRS to keep them attached to the respective Input Buffer transfer.

IBCM

Input Buffer Command Mask **(0510_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													STXRS	LDSS	LHSS
r													rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													STXRH	LDSH	LHSH
r													rwh	rwh	rwh

Field	Bits	Type	Description
LHSH	0	rwh	**Load Header Section Host** 0_B Header Section is not updated 1_B Header Section selected for transfer from Input Buffer to the Message RAM
LDSH	1	rwh	**Load Data Section Host** 0_B Data Section is not updated 1_B Data Section selected for transfer from Input Buffer to the Message RAM
STXRH	2	rwh	**Set Transmission Request Host** If this bit is set to 1, the Transmission Request flag TXRQ1.TXRn (n = 0-31) to TXRQ4.TXRn (n = 96-127) for the selected Message Buffer is set in the Transmission Request Registers to release the Message Buffer for transmission. In single-shot mode the flag is cleared by the Communication Controller after transmission has completed. TXRQ1.TXRn (n = 0-31) to TXRQ4.TXRn (n = 96-127) are evaluated for transmit buffer only. 0_B Reset Transmission Request flag 1_B Set Transmission Request flag, transmit buffer released for transmission
LHSS	16	rh	**Load Header Section Shadow** 0_B Header Section is not updated 1_B Header Section selected for transfer from Input Buffer to the Message RAM (transfer is ongoing of finalized)
LDSS	17	rh	**Load Data Section Shadow** 0_B Data Section is not updated 1_B Data Section selected for transfer from Input Buffer to the Message RAM (transfer is ongoing of finalized)

Field	Bits	Type	Description
STXRS	18	rh	**Transmission Request Shadow** If this bit is set to 1, the Transmission Request flag TXRQ1.TXRn (n = 0-31) to TXRQ4.TXRn (n = 96-127) for the selected Message Buffer is set in the Transmission Request Registers to release the Message Buffer for transmission. In single-shot mode the flag is cleared by the Communication Controller after transmission has completed. TXRQ1.TXRn (n = 0-31) to TXRQ4.TXRn (n = 96-127) are evaluated for transmit buffer only. 0_B Reset Transmission Request flag 1_B Set Transmission Request flag, transmit buffer released for transmission (operation is ongoing of finalized)
0	15:3, 31:19	r	**Reserved** Returns 0 if read; should be written with 0.

Input Buffer Command Request

When the Host writes the number of the target Message Buffer in the Message RAM to IBRH in the Input Buffer Command Request register, IBF Host and IBF Shadow are swapped. In addition the Message Buffer numbers stored under IBRH and IBRS are also swapped (see also “Data Transfer from Input Buffer to Message RAM”).

With this write operation the IBSYS bit in the Input Buffer Command Request register is set to 1. The Message Handler then starts to transfer the contents of IBF Shadow to the Message Buffer in the Message RAM selected by IBRS.

While the Message Handler transfers the data from IBF Shadow to the target Message Buffer in the Message RAM, the Host may write the next message into the IBF Host. After the transfer between IBF Shadow and the Message RAM has completed, the IBSYS bit is set back to 0 and the next transfer to the Message RAM may be started by the Host by writing the respective target Message Buffer number to IBRH.

If a write access to IBRH occurs while IBSYS is 1, IBSYH is set to 1. After completion of the ongoing data transfer from IBF Shadow to the Message RAM, IBF Host and IBF Shadow are swapped, IBSYH is reset to 0. IBSYS remains set to 1, and the next transfer to the Message RAM is started. In addition the Message Buffer numbers stored under IBRH and IBRS are also swapped. Any write access to an Input Buffer register while both IBSYS and IBSYH are set will cause the error flag EIR.IIBA to be set. In this case the Input Buffer will not be changed.

IBCR

Input Buffer Command Request (0514_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBSYS	**0**								**IBRS**						
rh	r								rh						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBSYH	**0**								**IBRH**						
rh	r								rwh						

Field	Bits	Type	Description
IBRH	6:0	rwh	**Input Buffer Request Host** Selects the target Message Buffer in the Message RAM for data transfer from Input Buffer. Valid values are 00_H to $7F_H$ (0…127).

Field	Bits	Type	Description
IBSYH	15	rh	**Input Buffer Busy Host** Set to 1 by writing IBRH while IBSYS is still 1. After the ongoing transfer between IBF Shadow and the Message RAM has completed, the IBSYH is set back to 0. 0_B No request pending 1_B Request while transfer between IBF Shadow and Message RAM in progress
IBRS	22:16	rh	**Input Buffer Request Shadow** Number of the target Message Buffer actually updated/lately updated. Valid values are 00_H to $7F_H$ (0…127).
IBSYS	31	rh	**Input Buffer Busy Shadow** Set to 1 after writing IBRH. When the transfer between IBF Shadow and the Message RAM has completed, IBSYS is set back to 0. 0_B Transfer between IBF Shadow and Message RAM completed 1_B Transfer between IBF Shadow and Message RAM in progress
0	14:7, 30:23	r	**Reserved** Returns 0 if read; should be written with 0.

41.4.2.10 Output Buffer

Double buffer structure consisting of Output Buffer Host and Output Buffer Shadow. Used to read out Message Buffers from the Message RAM. While the Host can read from Output Buffer Host, the Message Handler transfers the selected Message Buffer from Message RAM to the respective Output Buffer Shadow. The data transfer between Message RAM and Output Buffer (OBF) is described in “Data Transfer from Message RAM to Output Buffer”.

Read Data Section n

The Read Data Section nn (RDDSn, n = 01-64) holds the data words read from the Data Section of the addressed Message Buffer. The data words are read from the Message RAM in reception order from DW1 (byte0, byte1) to DW_{PL} (PL = number of data words as defined by the Payload Length).

Note: DW127 is located on RDDS64.MDW. In this case RDDS64.MDW is unused (no valid data). The Output Buffer RAMs are initialized to zero by CHI command CLEAR_RAMS.

RDDSn (n=01-64)
Read Data Section n **(0600_H+(n-1)*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MDRB3								MDRB2							
rh								rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MDRB1								MDRB0							
rh								rh							

Field	Bits	Type	Description
MDRB0	7:0	rh	**32-Bit Word nn, Byte 0**
MDRB1	15:8	rh	**32-Bit Word nn, Byte 1**
MDRB2	23:16	rh	**32-Bit Word nn, Byte 2**
MDRB3	31:24	rh	**32-Bit Word nn, Byte 3**

Read Header Section 1

Values as configured by the Host via WRHS1 Register:

RDHS1
Read Header Section 1 **(0700_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		MBI	TXM	PPIT	CFG	CHB	CHA	0	CYC						
r		rh	rh	rh	rh	rh	rh	r	rh						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					FID										
r					rh										

Field	Bits	Type	Description
FID	10:0	rh	**Frame ID**
CYC	22:16	rh	**Cycle Code**
CHA	24	rh	**Channel Filter Control A**
CHB	25	rh	**Channel Filter Control B**
CFG	26	rh	**Message Buffer Direction Configuration Bit**
PPIT	27	rh	**Payload Preamble Indicator Transmit**
TXM	28	rh	**Transmission Mode**
MBI	29	rh	**Message Buffer Service Request**
0	15:11, 23, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

Note: In case that the Message Buffer read from the Message RAM belongs to the receive FIFO, FID holds the received Frame ID, while CYC, CHA, CHB, CFG, PPIT, TXM, and MBI are reset to zero.

Table 423 Channel Filter Control Bits

CHA	CHB	Transmit Buffertransmit Frame on	Receive Bufferstore Frame received from
1[1)]	1[1)]	Both Channels (static segment only)	Channel A or B (store first semantically valid Frame, static segment only)
1	0	Channel A	Channel A
0	1	Channel B	Channel B
0	0	No Transmission	Ignore Frame

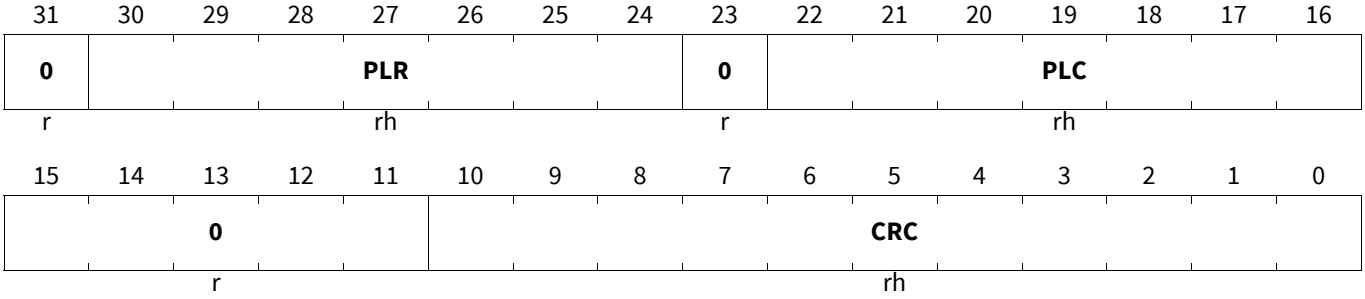
Note: If a Message Buffer is configured for the dynamic segment and both bits of the channel filtering field are set to 1, no Frames are transmitted resp. received Frames are ignored (same function as CHA = CHB = 0)

Read Header Section 2

Note: The Message RAM is organized in 4-byte words. When received data is stored into a Message Buffer's Data Section, the number of 2-byte data words written into the Message Buffer is PLC rounded to the next even value. PLC should be configured identical for all Message Buffers belonging to the receive FIFO. Header 2 is updated from Data Frames only.

RDHS2

Read Header Section 2 **(0704_H)** **Application Reset Value: $0000\ 0000_H$**



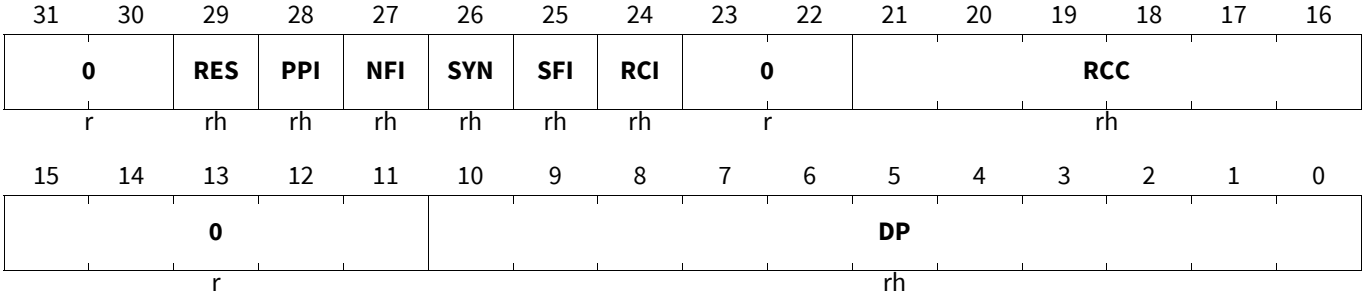
Field	Bits	Type	Description
CRC	10:0	rh	**Header CRC(vRF!Header!HeaderCRC)** Receive Buffer: Configuration not required. Header CRC updated from receive Data Frames. Transmit Buffer: Header CRC calculated and configured by the Host
PLC	22:16	rh	**Payload Length Configured** Length of Data Section (number of 2-byte words) as configured by the Host.
PLR	30:24	rh	**Payload Length Received(vRF!Header!Length)** Payload length value updated from received Data Frame (exception: if Message Buffer belongs to the receive FIFO PLR is also updated from received NULL Frames). When a message is stored into a Message Buffer the following behavior with respect to payload length received and payload length configured is implemented: • **PLR > PLC:** The payload data stored in the Message Buffer is truncated to the payload length configured for even PLC or else truncated to PLC + 1. • **PLR ≤ PLC:** The received payload data is stored into the Message Buffers Data Section. The remaining data bytes of the Data Section as configured by PLC are filled with undefined data. • **PLR = 0:** The Message Buffer's Data Section is filled with undefined data. • **PLC = 0:** Message Buffer has no Data Section configured. No data is stored into the Message Buffer's Data Section.
0	15:11, 23, 31	r	**Reserved** Returns 0 if read; should be written with 0.

Read Header Section 3

Note: Header 3 is updated from Data Frames only.

RDHS3

Read Header Section 3 **(0708_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DP	10:0	rh	**Data Pointer** Pointer to the first 32-bit word of the Data Section of the addressed Message Buffer in the Message RAM.
RCC	21:16	rh	**Receive Cycle Count(vRF!Header!CycleCount)** Cycle counter value updated from received Data Frame.
RCI	24	rh	**Received on Channel Indicator(vSS!Channel)** Indicates the channel from which the received Data Frame was taken to update the respective receive buffer. 0_B Frame received on channel B 1_B Frame received on channel A
SFI	25	rh	**Startup Frame Indicator(vRF!Header!SuFIndicator)** A Startup Frame is marked by the Startup Frame indicator. 0_B The received Frame is not a startup Frame 1_B The received Frame is a startup Frame
SYN	26	rh	**SYNC Frame Indicator(vRF!Header!SyFIndicator)** A SYNC Frame is marked by the SYNC Frame indicator. 0_B The received Frame is not a SYNC Frame 1_B The received Frame is a SYNC Frame
NFI	27	rh	**NULL Frame Indicator(vRF!Header!NFIndicator)** Is set to 1 after storage of the first received Data Frame. 0_B Up to now no Data Frame has been stored into the respective Message Buffer 1_B At least one Data Frame has been stored into the respective Message Buffer
PPI	28	rh	**Payload Preamble Indicator(vRF!Header!PPIndicator)** The payload preamble indicator defines whether a Network Management vector or message ID is contained within the Payload Segment of the received Frame. 0_B The Payload Segment of the received Frame does not contain a Network Management vector nor a message ID 1_B Static segment: Network Management vector in the first part of the payload Dynamic segment: Message ID in the first part of the payload

Field	Bits	Type	Description
RES	29	rh	**Reserved Bit(vRF!Header!Reserved)** Reflects the state of the received reserved bit. The reserved bit is transmitted as 0.
0	15:11, 23:22, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

Message Buffer Status

The Message Buffer status is updated by the Communication Controller with respect to the assigned channel(s) latest at the end of the slot following the slot assigned to the Message Buffer. The flags are updated only when the Communication Controller is in “NORMAL_ACTIVE” or “NORMAL_PASSIVE” state. If only one channel (A or B) is assigned to a Message Buffer, the channel-specific status flags of the other channel are written to zero. If both channels are assigned to a Message Buffer, the channel-specific status flags of both channels are updated. The Message Buffer status is updated only when the slot counter reached the configured Frame ID and when the cycle counter filter matched. When the Host updates a Message Buffer via Input Buffer, all MBS flags are reset to zero independent of which IBCM bits are set or not. For details about receive / transmit filtering see “Filtering and Masking”, “Transmit Process”, and “Receive Process”.

Whenever the Message Handler changes one of the flags VFRA, VFRB, SEOA, SEOB, CEOA, CEOB, SVOA, SVOB, TCIA, TCIB, ESA, ESB, MLST, FTA, FTB the respective Message Buffer’s MBC flag in registers MBSC1 to MBSC4 is set

MBS

Message Buffer Status **($070C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**RESS**	**PPIS**	**NFIS**	**SYNS**	**SFIS**	**RCIS**	**0**		**CCS**					
r		rh	rh	rh	rh	rh	rh	r		rh					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTB	**FTA**	**0**	**MLST**	**ESB**	**ESA**	**TCIB**	**TCIA**	**SVOB**	**SVOA**	**CEOB**	**CEOA**	**SEOB**	**SEOA**	**VFRB**	**VFRA**
rh	rh	r	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
VFRA	0	rh	**Valid Frame Received on Channel A(vSS!ValidFrameA)** A valid Frame indication is set if a valid Frame was received on channel A. 0_B No valid Frame received on channel A 1_B Valid Frame received on channel A
VFRB	1	rh	**Valid Frame Received on Channel B(vSS!ValidFrameB)** A valid Frame indication is set if a valid Frame was received on channel B. 0_B No valid Frame received on channel B 1_B Valid Frame received on channel B
SEOA	2	rh	**Syntax Error Observed on Channel A(vSS!SyntaxErrorA)** A syntax error was observed in the assigned slot on channel A. 0_B No syntax error observed on channel A 1_B Syntax error observed on channel A

Field	Bits	Type	Description
SEOB	3	rh	**Syntax Error Observed on Channel B(vSS!SyntaxErrorB)** A syntax error was observed in the assigned slot on channel B. 0_B No syntax error observed on channel B 1_B Syntax error observed on channel B
CEOA	4	rh	**Content Error Observed on Channel A(vSS!ContentErrorA)** A content error was observed in the assigned slot on channel A. 0_B No content error observed on channel A 1_B Content error observed on channel A
CEOB	5	rh	**Content Error Observed on Channel B(vSS!ContentErrorB)** A content error was observed in the assigned slot on channel B. 0_B No content error observed on channel B 1_B Content error observed on channel B
SVOA	6	rh	**Slot Boundary Violation Observed on Channel A(vSS!BViolationA)** A slot boundary violation (channel active at the start or at the end of the assigned slot) was observed on channel A. 0_B No slot boundary violation observed on channel A 1_B Slot boundary violation observed on channel A
SVOB	7	rh	**Slot Boundary Violation Observed on Channel B(vSS!BViolationB)** A slot boundary violation (channel active at the start or at the end of the assigned slot) was observed on channel B. 0_B No slot boundary violation observed on channel B 1_B Slot boundary violation observed on channel B
TCIA	8	rh	**Transmission Conflict Indication Channel A(vSS!TxConflictA)** A transmission conflict indication is set if a transmission conflict has occurred on channel A. 0_B No transmission conflict occurred on channel A 1_B Transmission conflict occurred on channel A
TCIB	9	rh	**Transmission Conflict Indication Channel B(vSS!TxConflictB)** A transmission conflict indication is set if a transmission conflict has occurred on channel B. 0_B No transmission conflict occurred on channel B 1_B Transmission conflict occurred on channel B
ESA	10	rh	**Empty Slot Channel A** In an empty slot there is no activity detected on the bus. The condition is checked in static and dynamic slots. 0_B Bus activity detected in the assigned slot on channel A 1_B No bus activity detected in the assigned slot on channel A
ESB	11	rh	**Empty Slot Channel B** In an empty slot there is no activity detected on the bus. The condition is checked in static and dynamic slots. 0_B Bus activity detected in the assigned slot on channel B 1_B No bus activity detected in the assigned slot on channel B

Field	Bits	Type	Description
MLST	12	rh	**Message Lost** The flag is set in case the Host did not read the message before the Message Buffer was updated from a received Data Frame. Not affected by reception of NULL Frames except for Message Buffers belonging to the receive FIFO. The flag is reset by a Host write to the Message Buffer via IBF or when a new message is stored into the Message Buffer after the Message Buffers ND flag was reset by reading out the Message Buffer via OBF. 0_B No message lost 1_B Unprocessed message was overwritten
FTA	14	rh	**Frame Transmitted on Channel A** Indicates that this node has transmitted a Data Frame in the assigned slot on channel A. *Note: The FlexRay™ protocol specification requires that FTA can only be reset by the Host. Therefore the Cycle Count Status CCS for these bits is only valid for the cycle where the bits are set to 1* 0_B No transmission transmitted on channel A 1_B Data Frame transmitted on channel A in cycle defined by CCS bit field
FTB	15	rh	**Frame Transmitted on Channel B** Indicates that this node has transmitted a Data Frame in the assigned slot on channel B. *Note: The FlexRay™ protocol specification requires that FTB can only be reset by the Host. Therefore the Cycle Count Status CCS for these bits is only valid for the cycle where the bits are set to 1* 0_B No transmission transmitted on channel B 1_B Data Frame transmitted on channel B in cycle defined by CCS bit field
CCS	21:16	rh	**Cycle Count Status** Cycle Count when status (MBS register) has been updated.
RCIS	24	rh	**Received on Channel Indicator Status(vSS!Channel)** Indicates the channel on which the Frame was received. *Note: For receive buffers (CFG = 0) the RCIS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.* 0_B Frame received on channel B 1_B Frame received on channel A

Field	Bits	Type	Description
SFIS	25	rh	**Startup Frame Indicator Status(vRF!Header!SuFIndicator)** A Startup Frame is marked by the Startup Frame indicator. *Note: For receive buffers (CFG = 0) the SFIS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.* 0_B No Startup Frame received 1_B The received Frame is a startup Frame
SYNS	26	rh	**SYNC Frame Indicator Status(vRF!Header!SyFIndicator)** A SYNC Frame is marked by the SYNC Frame indicator. *Note: For receive buffers (CFG = 0) the SYNS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.* 0_B No SYNC Frame received 1_B The received Frame is a SYNC Frame
NFIS	27	rh	**NULL Frame Indicator Status(vRF!Header!NFIndicator)** If reset to 0 the Payload Segment of the received Frame contains no usable data. *Note: For receive buffers (CFG = 0) the NFIS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.* 0_B Received Frame is a NULL Frame 1_B Received Frame is not a NULL Frame
PPIS	28	rh	**Payload Preamble Indictor Status(vRF!Header!PPIndicator)** The payload preamble indicator defines whether a Network Management vector or message ID is contained within the Payload Segment of the received Frame. *Note: For receive buffers (CFG = 0) the PPIS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.* 0_B Static Segment: The Payload Segment of the received Frame does not contain a Network Management vector or a message ID Dynamic Segment: The Payload Segment of the received Frame does not contain a Network Management vector or a message ID 1_B Static Segment: Network Management vector at the beginning of the payload Dynamic Segment: Message ID at the beginning of the payload

Field	Bits	Type	Description
RESS	29	rh	**Reserved Bit Status(vRF!Header!Reserved)** Reflects the state of the received reserved bit. The reserved bit is transmitted as 0. *Note: For receive buffers (CFG = 0) the RESS is updated from both valid data and NULL Frames. If no valid Frame was received, the previous value is maintained. For transmit buffers the flags have no meaning and should be ignored.*
0	13, 23:22, 31:30	r	**Reserved** Returns 0 if read; should be written with 0.

Output Buffer Command Mask

Configures how the Output Buffer is updated from the Message Buffer in the Message RAM selected by the Output Buffer Command Request register. If OBF Host and OBF Shadow are swapped, also mask bits OBCM.RDSH and OBCM.RHSH are swapped with bits OBCM.RDSS and OBCM.RHSS to keep them attached to the respective Output Buffer transfer.

Note: After the transfer of the Header Section from the Message RAM to OBF Shadow has completed, the Message Buffer status changed flag MBCn (n = 0-31) to MBCn (n = 96-127) of the selected Message Buffer in the Message Buffer Changed MBSC1 to MBSC4 registers is cleared. After the transfer of the Data Section from the Message RAM to OBF Shadow has completed, the New Data flag NDn (n = 0-31) to NDn (n = 96-127) of the selected Message Buffer in the New Data NDAT1 to NDAT4 registers is cleared.

OBCM

Output Buffer Command Mask (0710_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0														RDSH	RHSH
r														rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														RDSS	RHSS
r														rwh	rwh

Field	Bits	Type	Description
RHSS	0	rwh	**Read Header Section Shadow** 0_B Header Section is not read 1_B Header Section selected for transfer from Message RAM to Output Buffer
RDSS	1	rwh	**Read Data Section Shadow** 0_B Data Section is not read 1_B Data Section selected for transfer from Message RAM to Output Buffer

Field	Bits	Type	Description
RHSH	16	rh	**Read Header Section Host** 0_B Header Section is not read 1_B Header Section selected for transfer from Message RAM to Output Buffer
RDSH	17	rh	**Read Data Section Host** 0_B Data Section is not read 1_B Data Section selected for transfer from Message RAM to Output Buffer
0	15:2, 31:18	r	**Reserved** Returns 0 if read; should be written with 0.

Output Buffer Command Request

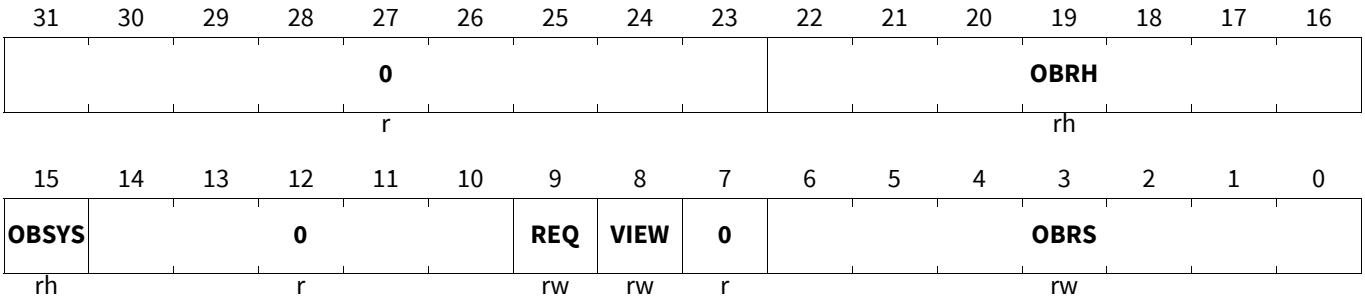
The Message Buffer selected by OBCR.OBRS is transferred from the Message RAM to the Output Buffer as soon as the Host has set OBCR.REQ. Bit OBCR.REQ can only be set while OBCR.OBSYS is 0 (see also “Data Transfer from Message RAM to Output Buffer”).

After setting OBCR.REQ, OBCR.OBSYS is automatically set, and the transfer of the Message Buffer selected by OBCR.OBRS from the Message RAM to Output Buffer Shadow is started. When the transfer between the Message RAM and OBF Shadow has completed, this is signalled by clearing OBCR.OBSYS. By setting OBCR.VIEW while OBCR.OBSYS is 0, OBF Host and OBF Shadow are swapped. When Output Buffer Host and Output Buffer Shadow are swapped, also mask bits OBCM.RDSH and OBCM.RHSH are swapped with bits OBCM.RDSS and OBCM.RHSS to keep them attached to the respective Output Buffer transfer. Now the Host can read the transferred Message Buffer from OBF Host. In parallel the Message Handler may transfer the next message from the Message RAM to OBF Shadow if OBCR.VIEW and OBCR.REQ are set at the same time.

Any write access to an Output Buffer register while OBCR.OBSYS is set will cause the error flag EIR.IOBA to be set. In this case the Output Buffer will not be changed.

OBCR

Output Buffer Command Request **(0714_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
OBRS	6:0	rw	**Output Buffer Request Shadow** Number of source Message Buffer to be transferred from the Message RAM to OBF Shadow. Valid values are 00_H to $7F_H$ (0 to 127). If the number of the first Message Buffer of the receive FIFO is written to this register the Message Handler transfers the Message Buffer addressed by the GET Index Register (GIDX, “FIFO Function”) to OBF Shadow.

Field	Bits	Type	Description
VIEW	8	rw	**View Shadow Buffer** Toggles between OBF Shadow and OBF Host. Only writeable while OBCR.OBSYS = 0. 0_B No action 1_B Swap OBF Shadow and OBF Host
REQ	9	rw	**Request Message RAM Transfer** Requests transfer of Message Buffer addressed by OBCR.OBRS from Message RAM to OBF Shadow. Only writeable while OBCR.OBSYS = 0. 0_B No request 1_B Transfer to OBF Shadow requested
OBSYS	15	rh	**Output Buffer Busy Shadow** Set to 1 after setting bit OBCR.REQ. When the transfer between the Message RAM and OBF Shadow has completed, OBCR.OBSYS is cleared again. 0_B No transfer in progress 1_B Transfer between Message RAM and OBF Shadow in progress
OBRH	22:16	rh	**Output Buffer Request Host** Number of Message Buffer currently accessible by the Host via RDHS1 to RDHS3, MBS, and RDDSnn (nn = 01-64). By setting OBCR.VIEW OBF Shadow and OBF Host are swapped and the transferred Message Buffer is accessible by the Host. Valid values are 00_H to $7F_H$ (01 to 127).
0	7, 14:10, 31:23	r	**Reserved** Returns 0 if read; should be written with 0.

41.5 Destructive Debug

On destructive debug entry ie., when DMU_SP_PROCONHSMCFG.DESTDBG = 11b and DMU_HF_PROCONDBG.EDM = 11b, the ERAY Transmit Data and transmit enable for Channel A and Channel B (TXDA, TXDB, TXENA and TXENB) are tied to logic ‘1b’. This blocks any further ERAY transmission when AURIX™ TC3xx Platform has entered the Destructive Debug state.

41.6 IO Interfaces

Table 424 List of ERAY Interface Signals

Interface Signals	I/O	Description
INT0_INT	out	**E-RAY Service Request 0**
INT1_INT	out	**E-RAY Service Request 1**
TINT0_INT	out	**E-RAY Timer Interrupt 0 Service Request**
TINT1_INT	out	**E-RAY Timer Interrupt 1 Service Request**
NDAT0_INT	out	**E-RAY New Data 0 Service Request**
NDAT1_INT	out	**E-RAY New Data 1 Service Request**
MBSC0_INT	out	**E-RAY Message Buffer Status Changed 0 Service Request**
MBSC1_INT	out	**E-RAY Message Buffer Status Changed 1 Service Request**

Table 424 List of ERAY Interface Signals (cont'd)

Interface Signals	I/O	Description
OBUSY	out	**E-RAY Output Buffer Busy Service Request**
IBUSY_INT	out	**E-RAY Input Buffer Busy Service Request**
sleep_n	in	**turn-off request from processor**
STPWT(3:0)	in	**StoP Watch Trigger signal**
TXDA	out	**Transmit Channel A**
TXDB	out	**Transmit Channel B**
TXENA	out	**Transmit Enable Channel A**
TXENB	out	**Transmit Enable Channel B**
RXDA0	in	**Receive Channel A0**
RXDA1	in	**Receive Channel A1**
RXDA2	in	**Receive Channel A2**
RXDA3	in	**Receive Channel A3**
RXDB0	in	**Receive Channel B0**
RXDB1	in	**Receive Channel B1**
RXDB2	in	**Receive Channel B2**
RXDB3	in	**Receive Channel B3**
INT0	out	**Interrupt 0 (high-active)**
INT1	out	**Interrupt 1 (high-active)**
TINT0	out	**Timer Interrupt 0 (high-active)**
TINT1	out	**Timer Interrupt 1 (high-active)**
NDAT0	out	**New Dat Interrupt 0 (high-active)**
NDAT1	out	**New Dat Interrupt 1 (high-active)**
MBSC0	out	**MBSC Interrupt 0 (high-active) (Message Buffer 0 Status changed)**
MBSC1	out	**MBSC Interrupt 1 (high-active) (Message Buffer 0 Status changed)**
OBUSY	out	**2 cycle pulse derived from falling edge of obusy**
IBUSY	out	**2 cycle pulse derived from falling edge of ibusy**
OBUSYNP	out	**obusy no pulse**
IBUSYNP	out	**ibusy no pulse**
MT	out	**Macrotick-clock from CC (synchronous to fpi clock)**
TXDREFEN	in	**Reference Stream Enable**

41.7 Revision History

Table 425 Revision History

Reference	Changes to Previous Version	Comment
V3.2.9		
Page 201	Older versions removed from revision history	–
V3.2.10		
Page 1, **Page 61**	f_{PLL_ERAY} corrected to f_{ERAY} in figures “General Block Diagram of the E-Ray Interface” and “Detailed Block Diagram of the E-Ray Interface”.	
V3.2.11		
Page 60	Wrong unit “ms” corrected to “µs”.	
Page 3	In E-Ray Block Diagram “TDAA“ corrected to “TXDA” and blanks removed.	
Page 15	Register bit field SUCC1.CMD value 0011_H corrected to 1011_H.	
Page 20	Sentence about SUCC1.TXST and SUCC1.TXSY rephrased.	
Page 24	NORMAL_OPERATION replaced by NORMAL_ACTIVE.	
Page 26	DEFAULT_CONFIG replaced by CONFIG and CONFIG replaced by DEFAULT_CONFIG.	
Page 26	Sentence redundancy removed (2 times).	
Page 29	Typos in DEFAULT_CONFIG removed.	
Page 31	Transmission Request 1,2 registers corrected to 1 to 4.	
Page 34	Bit RFF corrected to RFCL.	
Page 58	For SIR register RFF/Receive FIFO full corrected to RFCL/Receive FIFO Critical Level.	
Page 36	WRHS2 replaced by WRHS3 (duplicity), at “CUST1.IBF1PAG and” space added.	
Page 38	“rw” corrected to “rwh” for pos 2, 1 and 0.	
Page 38	“21…16” corrected to “22…16” for bit IBRS.	
Page 38	“5…0” corrected to ”6…0” for bit IBRH.	
Page 38	WRHS2 replaced by WRHS3 (duplicity).	
Page 39	“rw” corrected to “rwh” for bits RDSS and RHSS.	
Page 39	“rwh” corrected to “rw” for bits VIEW and OBRS.	
Page 49	“2048 x 32” replaced by “2048 32-bit words”.	
Page 54	In example “MB1” replaced by “MBn” (8 times), zeros removed (4 times), “2” replaced by “1” (1 time) and iterator “n” replaced by “m” (4 times).	
Page 59	“TINTxSR” replaced by “INTxSR”.	
Page 59	“new data” corrected to “message buffer status changed”.	
Page 106	In register EILS short description for bit field TABBL corrected to “Transmission Across Boundary Channel B Service Request Line”.	
Page 125	In register bit field SUCC1.TXST unnecessary foot note numbers removed.	
Page 148	In register bit field CCEV.PTAC the description “SUCC1.PTA” corrected to “SUCC1.PTA-1”.	

Table 425 Revision History (cont'd)

Reference	Changes to Previous Version	Comment
Page 161	Table "Usage of the three Message Buffer Pointer" in register MRC updated for clarification.	
Page 125	In explanation "COMMAND_NOT_ACCEPTED" in register bit field SUCC1.CMD value written in upper case.	
Page 185	In register bit fields IBCM.STXRH and IBCM.STXRS the description "TXRQ4.TXRn (n = 0-31)" corrected to "TXRQ4.TXRn (n = 96-127)" (4 times).	
Page 193	In register bit field MBS.SYNS the description "Startup Frame" corrected to "SYNC Frame" (2 times).	

42 Peripheral Sensor Interface (PSI5)

The PSI5 module communicates with the external world via one I/O line for each channel. The PSI5RXx lines are the receive data input signals. If the optional unidirectional mode is used, the signals PSI5TXx are transmitted on separate GPIO ports. Receive and transmit path are always routed to two different ports.

Figure 634 shows a global view of the PSI5 interface.

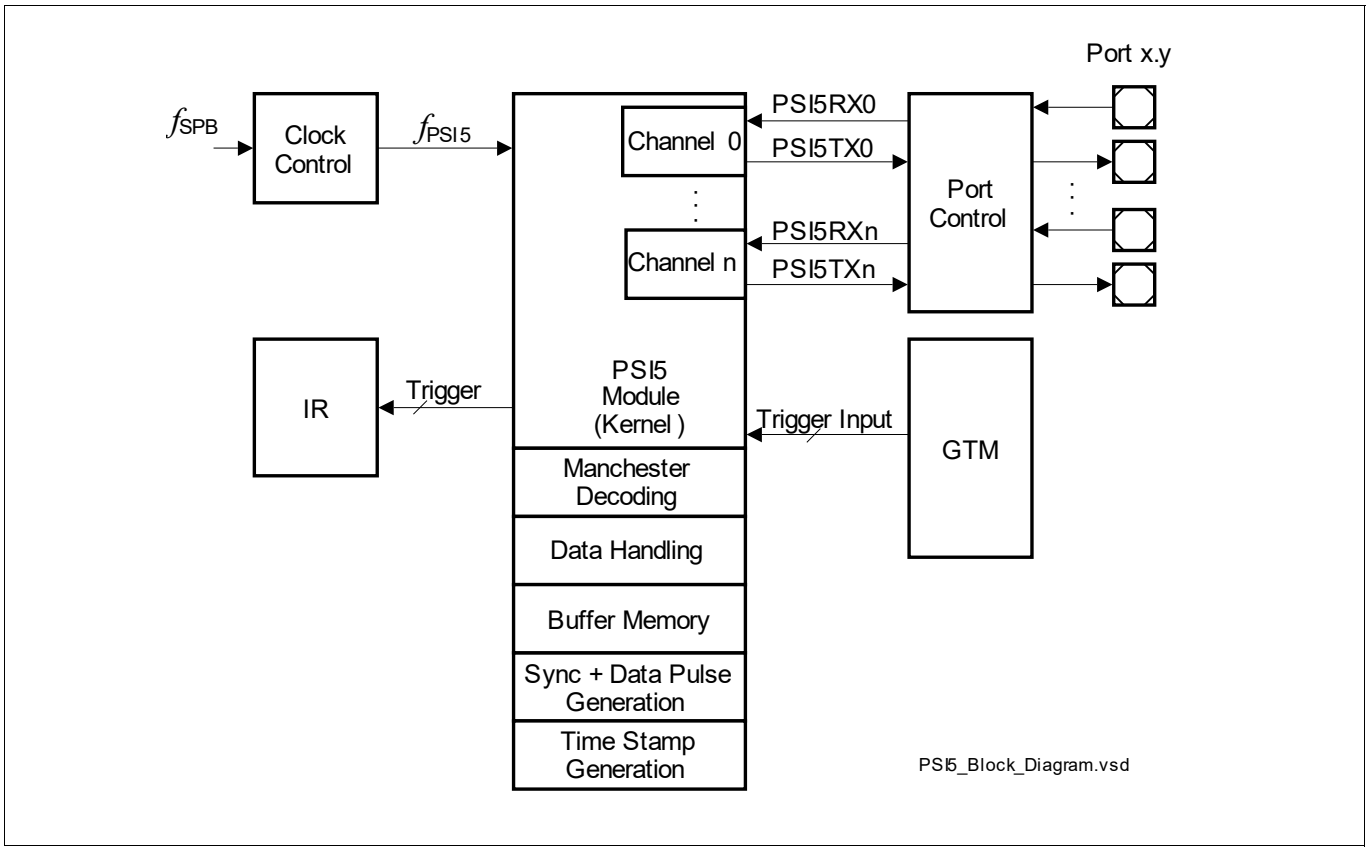


Figure 634 General Block Diagram of the PSI5 Interface

42.1 Feature List

FEATURE LIST

- Conformance with PSI5 protocol specification V1.3
- Data rates of 125 kbit/s and 189 kbit/s supported
- 4 PSI5 channels implemented
- All implemented channels are working independently in parallel
- Supports 6 sensor slots per channel
- Asynchronous and synchronous data transmission modes
  (control by micro controller in synchronous mode)
- Decoding manchester protocol
- Error recognition in manchester code
- Configurable data word length 8, 10, 16, 20, 24 bit according to standard (PSI5 V1.3)
- Support of non standard V1.3 frame length: 11… 33 bit
- CRC check of received sensor data implemented but CRC code transparent
- Support of Enhanced Serial Messages according to SENT SAE J2716 JAN 2010
- 24-Bit time stamp (resolution: 1µs)
- Storage of up to 32 frames per channel with time stamp
- FIFO access and management
- Buffer overrun detection
- Buffer Memory Status Overview Registers
- Support of ECU to Sensor communication
- Three 64-bit downstream data registers for data input, data preparation and data output
- Downstream data transmission by variation of pulse length (2.0)
- Downstream data transmission by leaving out sync pulses (V1.3)
- Staggering Sync Pulses of all channels to avoid too many channels sending sync pulses in parallel
- Generation of 3 or 6 bit CRC for downstream data
- Start sequence and CRC generator for downstream data
- One sum interrupt for general events
- One sum interrupt on selectable errors
- Sticky interrupt flags, error interrupt optional (default disabled)
- Optional output inversion for use of external open drain transistor
- Optional input inversion for use of external open transistor for level shifting

42.2 Overview

The PSI5 IP-module performs communication according to the PSI5 specification V1.3.

The Peripheral Sensor Interface is an interface for automotive sensor applications. PSI5 is an open standard based on existing sensor interfaces for peripheral airbag sensors, already proven in millions of airbag systems.

This module supports many additional features as discussed today in the standard organization.

The PSI5 interface provides a current loop based serial communication link typically used to connect airbag sensors or other peripheral devices.

While the physical layer is done externally, this module manages protocol handling and data representation to the application. Note that there is no on chip phy, i.e. the current to voltage and voltage to sync pulse translation is done externally.

Receive data on a PSI5 channel can be set up according to the underlying application. In particular the number of bits forming one value is configured.

The message storage consists of a 32-bit buffer register for each channel and an additional 32-bit register containing the 24-bit time stamp and additional status bits. These 64 bits per frame are additionally stored in a buffer of 32 lines per channel.

In Host to sensor communication mode, the module can provide regular sync pulses and select between internal and external timer/trigger resources. It supports as well CPU to sensor communication, e.g. for setting up the sensor and retrieving sensor status information.

The register set of the PSI5 module can be accessed directly by the CPU for configuration, data read out and status query.

The PSI5 module supports the following features:

42.3 Functional Description

42.3.1 General Operation

PSI5 is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/D converters and PWM and as a simpler low cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

Figure 635 shows a typical device application in which a PSI5 interface reads a sensor device.

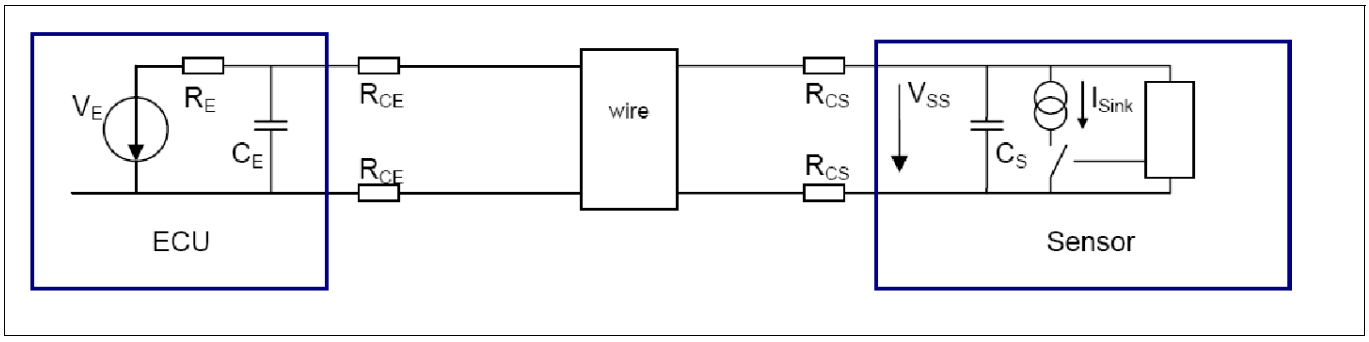


Figure 635 PSI5 to External Device Connection

PSI5 communication is

- asynchronous, unidirectional from sensor to controller without any synchronization
- synchronous, bidirectional with a sync pulse from the ECU triggering messages
- synchronous, bidirectional, sync pulses additionally coding data from ECU to sensor

The sensor signal is transmitted as a series of current pulses in Manchester coding.

The sync pulses are transmitted by increasing the voltage.

Figure 636 shows a typical device application in which a PSI5 ECU reads multiple PSI5 sensor devices on a bus.

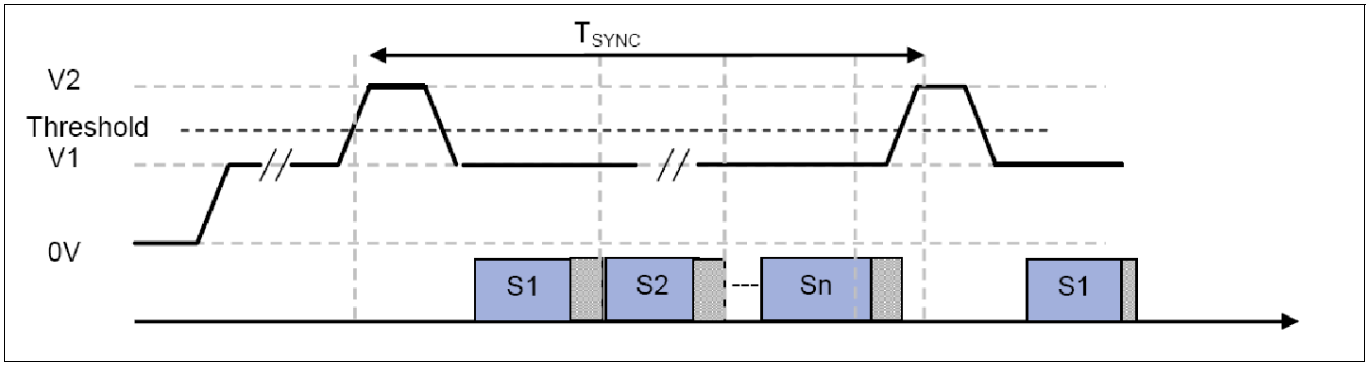


Figure 636 PSI5 Frames with Sync pulses

42.3.2 Definitions

SENT: Single Edge Nibble Transmission

Nibble: Four Bit value between 0 and 15 = half a Byte = one character in hex (0 to F)

ASIC: Application Specific Integrated Circuit

CAN: Controller Area Network

LIN: Local Interconnect Network

ISO: International Organization for Standardization

ECU: Electronic Control Unit

FSM: Finite State Machine

SOF: Start of Frame

EOF: End of Frame

STS: Start Sequence

AD: Address Data (e.g. of ECU to Sensor Frame)

SD: Send Data (Payload of ECU to Sensor Frame)

CRC: Cyclic Redundancy Check

42.3.3 PSI5 Operation

The Module supports two modes:

Standard PSI5 Mode supports communication fully compliant to V1.3

Extended PSI5 Mode supporting additional bit fields and longer frames. (See below)

In standard V1.3 mode, a specific start condition and a start sequence switch the sensor in ECU to Sensor communication mode. In this mode, the sync pulses are no longer used as triggers for data frames but are interpreted as data. A pulse represents a 1 and a missing pulse represents a 0. This requires isochronous pulses.

In extended Mode (Power Train sub standard), the pulse width is modulated. A standard pulse represents a 0, a longer pulse a 1. This way continuos data transmission can be achieved.

42.3.4 Frame Formats and Definitions

This section describes the frame formats and definitions of the PSI5 protocol.

42.3.4.1 PSI5 V1.3 Frame

Figure 637 shows the layout and definitions of a standard PSI5 frame. Note that the PSI5 standard specifies that the least significant bit is sent out first. See standard for CRC and Parity definition.

Example: 16 Bit Data Word „0xAD2C“ with 3-Bit CRC

S1	S2	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	C2	C1	C0
0	0	0	0	1	1	0	1	0	0	1	0	1	1	0	1	0	1	1	0	0
		C = „1100“				2 = „0010“				D = „1101“				A = „1010“						

Figure 637 Standard PSI5 Frame

42.3.4.2 Extended PSI5 Frame (non standard)

Figure 638 shows the layout and definitions of an extended PSI5 frame as defined in V2.0 for automotive power train application. Note that the PSI5 standard specifies that the least significant bit is sent out first.

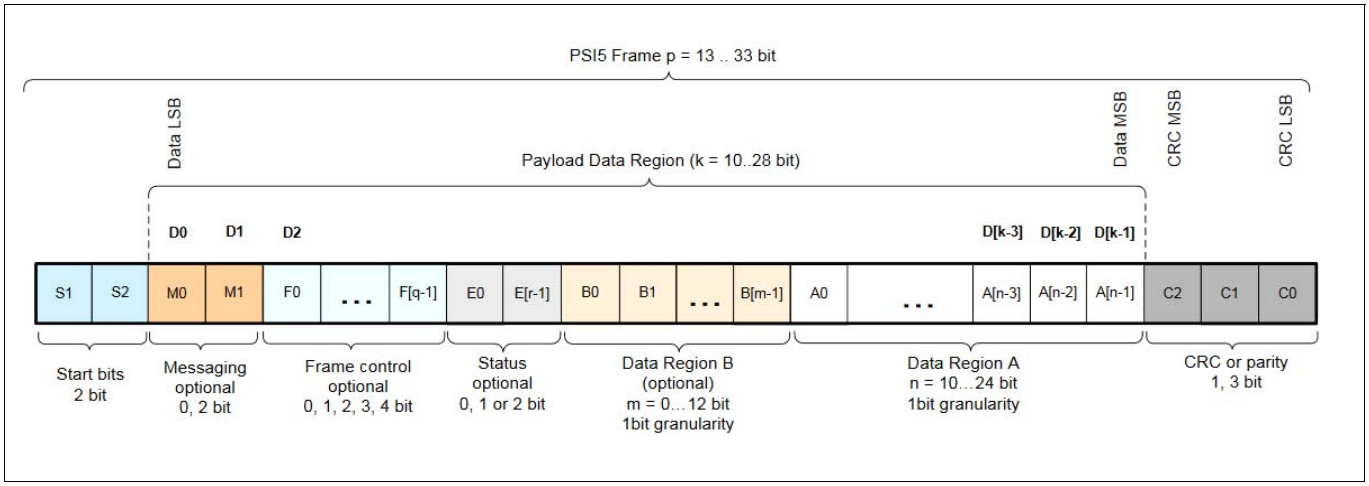


Figure 638 Extended PSI5 Frame

Enhanced PSI5 Sensor to ECU Message CRC

The transmission of PSI5 data is checked by the following protection modes. The transmission error detection must be selectable:

1) 1-bit even parity

2) 3-bit CRC

The applied generator polynomial of the CRC is g(x) = 1 + x + x3 with a binary start value (seed) “111”. The transmitter shall extend the data bits by three zeros (as MSBs). This augmented data word shall be fed (LSB first) into the shift registers of the CRC check. Start bits shall be ignored in this check. When the last zero of the augmentation is pending on the input adder, the shift registers contain the CRC checksum. These three check bits shall be transmitted in reverse order (MSB first: C2, C1, C0)

In this implementation the number of elements is configurable and thus longer or shorter than 8…28 bits!

42.3.4.3 Enhanced Serial Message (“Slow Channel”)

The serial data is transmitted bit wise per frame in bits [1:0] of the Messaging bit field of 18 consecutive messages from the transmitter.

42.3.4.4 Enhanced Serial Data Frame

Serial data (“Slow Channel”) is communicated in an 18 frame sequence as shown in **Figure 639**. The start of an Enhanced Serial Message is indicated by the unique pattern “1111110” in bit M1 constructed across 7 consecutive PSI5 messages. In message 1 - 6 they are set to “1”. In message 7, 13 and 18 they are set to “0”. 18 consecutive PSI5 messages must be successfully received (no errors) for the Enhanced Serial Message data to be received. The CRC generation is different from the CRC generation on the data nibbles. (See SENT Standard SAE_J2716_201001)

The serial message frame contains 21 bits of message data and a 6-bit frame-check sequence. Two different configurations can be chosen:

- 12-bit data and 8-bit message ID
- 16-bit data and 4-bit message ID

A configuration bit (serial data bit M1, serial communication Messaging bits of frame No. 8) determines the configuration of the enhanced serial message frame. It determines how the PSI5 module automatically interprets the serial data.

- Configuration bit = 0: 12-bit data and 8-bit message ID
- Configuration bit = 1: 16-bit data and 4-bit message ID

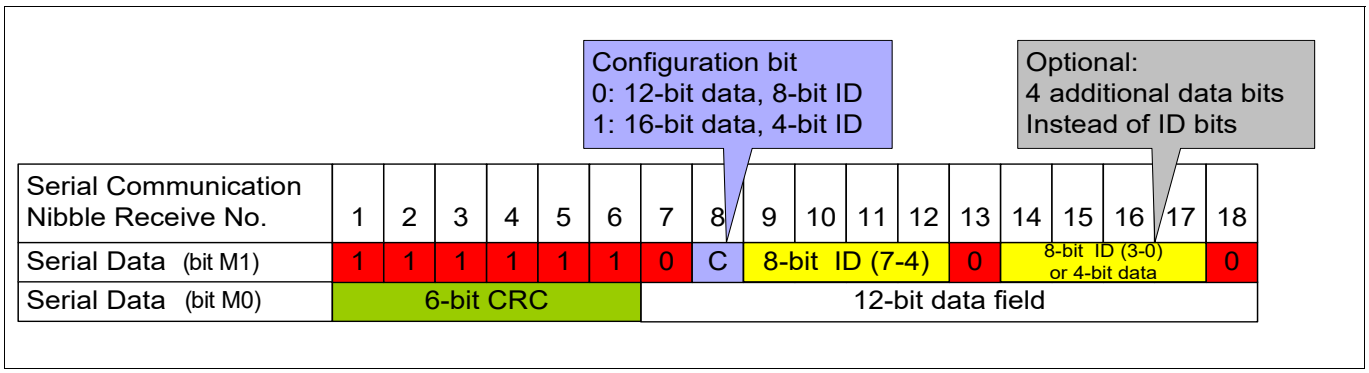


Figure 639 Enhanced Serial Data Frame

Figure 640 shows in detail the frame structure for Enhanced Serial Data Frames with Configuration Bit = 0.

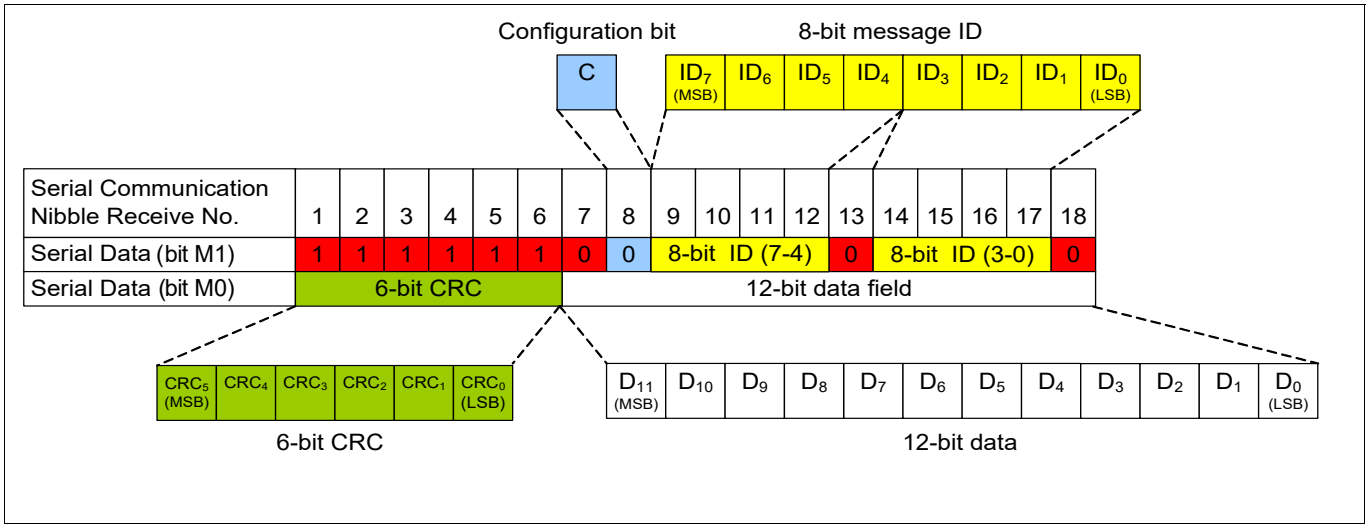


Figure 640 Configuration Bit = 0

Figure 641 shows in detail the frame structure for Enhanced Serial Data Frames with Configuration Bit = 1.

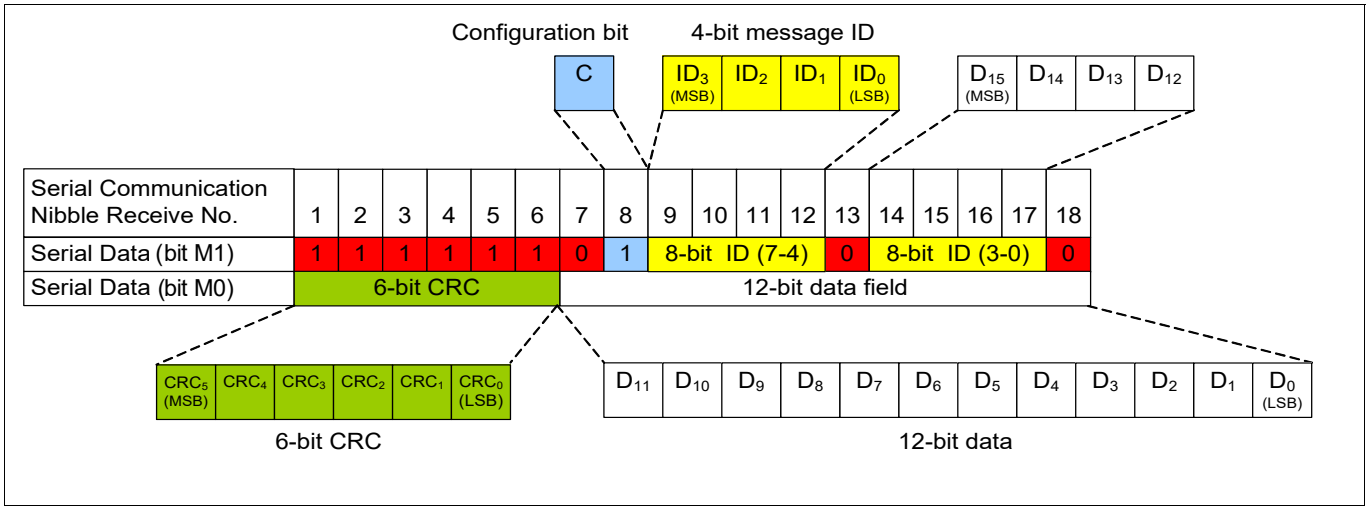


Figure 641 Configuration Bit = 1

42.3.5 Sync Pulses

Sync Pulses are used for two different purposes:

- Triggering a data frame for data acquisition from a sensor or
- ECU to sensor communication.

42.3.5.1 Synchronous Transmission

In the “synchronous” mode, the sensor (slave) starts to transfer a complete data frame only after a sync pulse is forced by the master on the OUT pin. An external phy translates this pulse into a voltage increase. The sensor then initiates a measurement and starts to calculate the new output data value. The data follows in a standard PSI5 frame, starting with the 2 start bits, data and Parity or CRC. The timing diagram in **Figure 642** visualizes a synchronous transmission

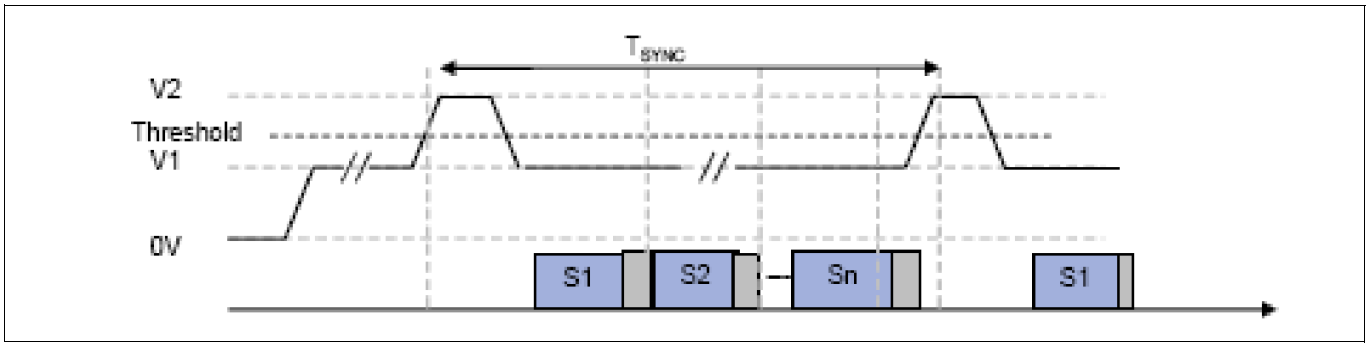


Figure 642 Synchronous Transmission

42.3.5.2 ECU to Sensor Communication

Two versions are supported:
legacy PSI5 V1.3 (see **Figure 643**) and
revised version (see **Figure 644** and **Figure 645**).

The legacy version

uses sync pulses as start condition and start bits to signal the start of the ECU to Sensor Communication. From this start on, the sync pulses are interpreted by the sensor as data. See standard for CRC and Parity definition.

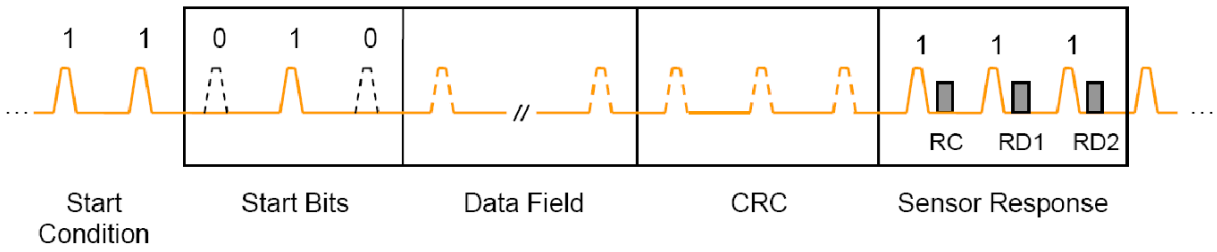


Figure 643 ECU to Sensor (legacy)

The revised version

allows for modulation of the sync pulse duration. This way bidirectional communication is possible continuously. Every ECU-to-Sensor frame starts with a nine-bit frame-start-sequence [0 1 1 1 1 1 1 1 0]. Synchronization is possible at the beginning of each frame.

Sync Bits

A consecutive train of seven ones within the transmitted data could be misinterpreted as a frame start condition. Therefore a “sync bit” (logical 0) is inserted at every seventh bit position (after every six bits of the sensor address and the payload data). With respect to the data transmission, these sync bits are treated like stuffing bits, which are removed at the receiver side.

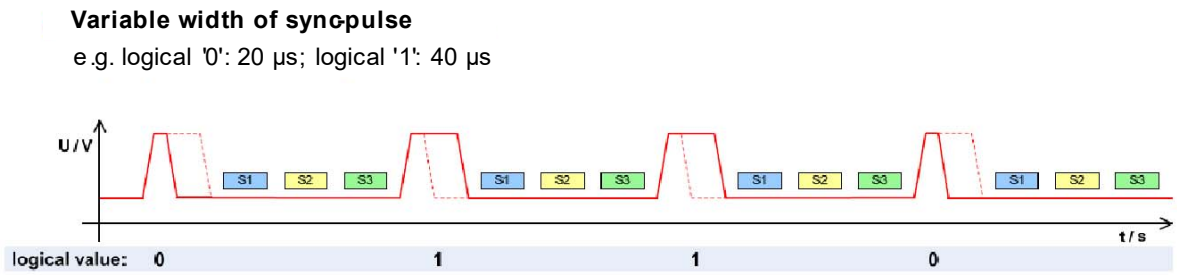


Figure 644 ECU to Sensor, Bit transmission (revised)

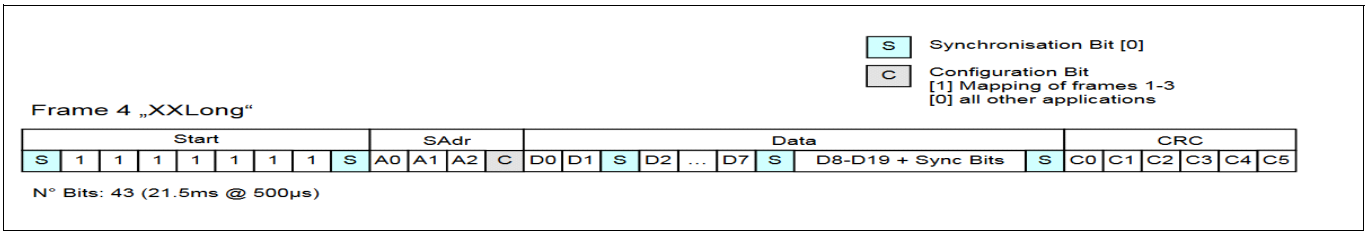


Figure 645 ECU to Sensor, Frame format (revised)

Enhanced ECU to Sensor Message CRC

The ECU-to-Sensor Frame has a 6-bit CRC.
The generator polynomial of the CRC is
g(x)=x6 + x4 +x3 +1
with a binary CRC initialization value “010101”.
The transmitter extends the data bits by six zeros (as MSBs).

This augmented data word shall be fed (LSB first) into the shift registers of the CRC check. Start bits and sync bits are ignored in this check.

When the last zero of the augmentation is pending on the input adder, the shift registers contain the CRC checksum.

These six check bits shall be transmitted LSB first [C0, C1…C5].

In this implementation the number of elements is configurable and thus longer or shorter than 24 bits (standard is 4 address and 20 data bits)!

42.3.6 Manchester Decoding

Decoding of the Manchester protocol:

- The Manchester decoder uses a nominal 4 MHz / 6.048 MHz sampling clock (32 samples per bit).
- A sampling counter, running at the sampling clock rate controls the sampling of the incoming Manchester encoded data.
- The Manchester signal transition occurring in the middle of the bit time resets the counter, making the counter count from 0-31 (nominally) between successive mid-bit times.
- Detection of the mid-bit transitions uses three consecutive samples (011 = rising edge, 100 = falling edge).
- The maximum value of the 6 bit counter shall be 41, at which point a Manchester error is detected due to lack of a transition, and the counter rolls over to 0. (“Idle Time”)
- The value of a Manchester encoded input data bit is determined by proper logic levels before and after a transition and proper timing of the transition.
- Bit-value determination nominally takes place at the 1/4 and 3/4 bit times, with the mid-bit transition occurring at the 1/2 bit time. This is nominally at sampling counter values of 8 (1/4 bit time) and 24 (3/4 bit time).
- Logic levels are determined using a 2 of 3 majority vote of consecutive samples (samples 23, 24, 25 and 7, 8, 9)
  - The combinations 111, 011, 101, 110 indicate a 1
  - The combinations 000, 001, 010, 100 indicate a 0

Manchester-2 is used by standard PSI5 V1.3:

- Start bits = ‘00’
- Falling edge corresponds to logical ‘1’
- Rising edge corresponds to logical ‘0’
- Least significant bit (LSB) is sent out first

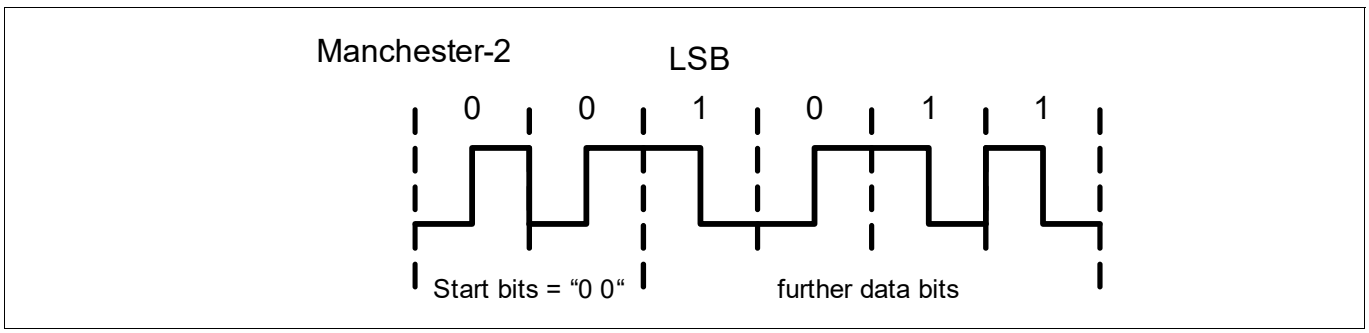


Figure 646 Manchester-2

Sampling Details

- Recognition of Frame Start

  - Manchester Code requires a deterministic Start of Frame
  - Frame Separation: An “Idle Time” of at least one Bit time is required
  - “Idle Time” is the time without edges on the line that is required to safely determine that no frame is being transferred (1 Bit time + tolerance = > 41 samples)
    The Idle Counter starts counting
    - on a falling edge, after sync pulse or after BOT, if the channel is switched on
    - if the line is low
    - up to 41 and stops and
    - activates the idle flag
    - until the line shows a rising edge, this clears the idle counter
  - After this idle time, a determined start bit sequence is required for synchronization of the phase of receiver and transmitter (‘0 0’ in case of the „Manchester 2“-code employed)
  - If no idle time occurs at the end of a frame, bits after the configured length are dropped.
  - If a 2nd frame comes too early (collision = no idle time) the 2nd frame is dropped. The decoder waits for an “Idle Time” for re synchronization.
  - Idle time is particularly required in case of Manchester Errors and frame collisions for a save re synchronization. Otherwise, two successive zeros in a damaged frame / a second colliding frame could lead to reception of wrong data. Note that back to back transfers of correct frames are forbidden by standard (T_gap required).
  - This idle time is required as well after a regular End Of Frame, Sync Pulse or a Blank Out Time to avoid inadvertent synchronization inside frames under transmission.
- Bit sampling
  - After the start of the protocol is detected and an edge on the line signal is detected and the counter is greater 25 the counter is set to ‘0’
  - If the counter is 41 the counter is set to ‘0’ and stops
  - The counter will be incremented each clock cycle
  - When the sample counter has the value 9, the level is stored in register cnt9_val
  - When the sample counter has the value 25, the level is stored in register cnt25_val
  - This register is reset if no frame is detected. Otherwise a wrong start bit detection could occur.
- Bit evaluation
  - When the sample counter value is 16, the decision is done for the new sample value:
  - Check if the last edge was detected correctly i.e.: edge detected between sample counter value 25 and sample counter value 41
  - If true:
  - Cnt_25_lev = 0 & cnt_9_lev = 1 => new sample = ‘0’
  - Cnt_25_lev = 1 & cnt_9_lev = 0 => new sample = ‘1’
  - If the edge check fails or cnt_25_lev = cnt_9_lev, Manchester Error is assumed.

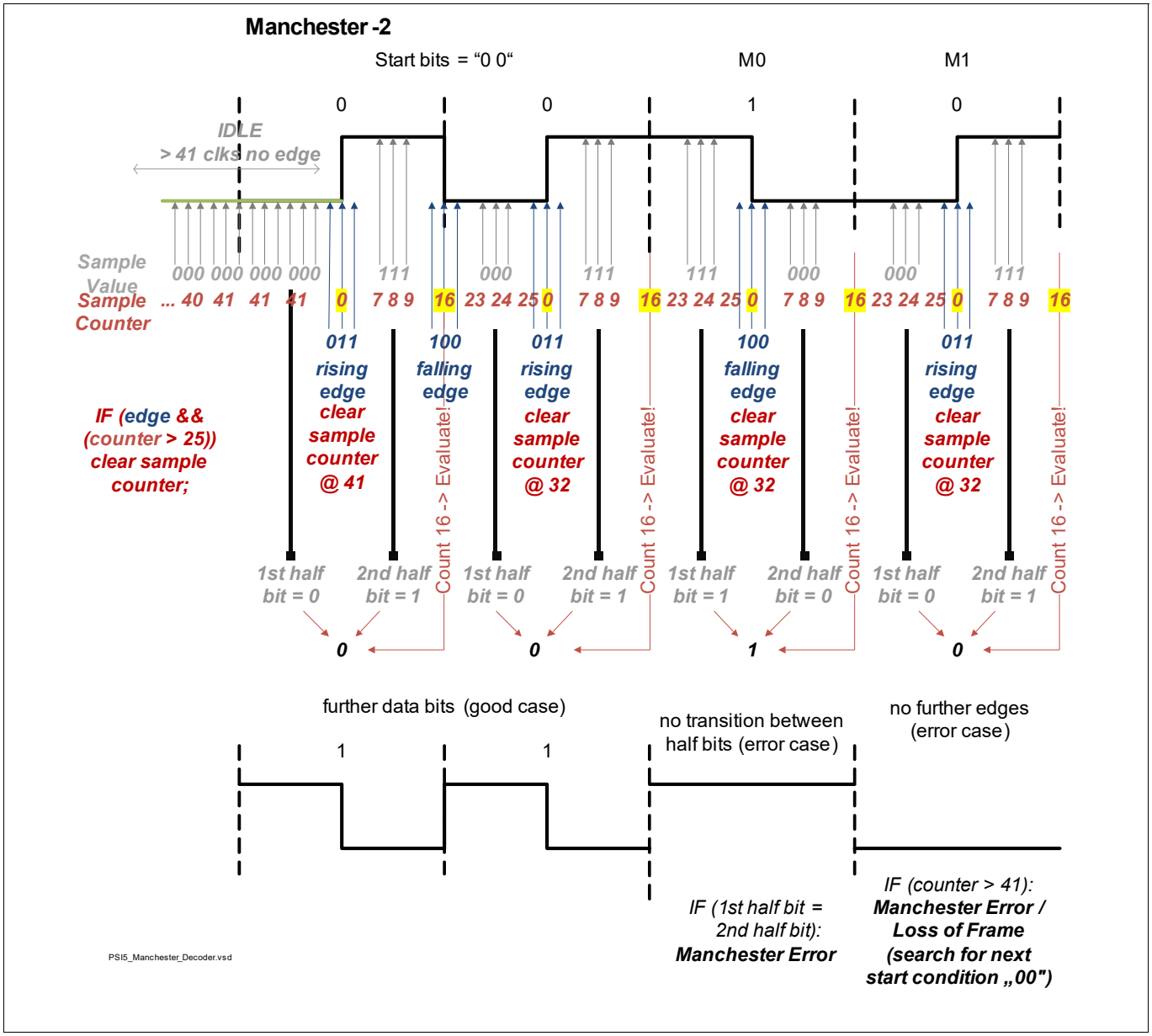


Figure 647 Manchester Decoder

Definition of Manchester Error (ME)

- Two half bits have the same value => ME, the decoder waits for an "Idle Time"
- > 41 sample times no edge => ME
- The ME is no dedicated interrupt but will lead to various error responses depending on where it occurs. See **Table 426**.

Table 426 Error Responses due to ME

Manchester Error (ME) Case	Response	Comment
First ME in start bits	Discard frame	NFI (if FEC is set) and empty frame stored (if FEC and VBS is set), Time Stamp = end of slot where SOF happened. This is true even if these start bits should start before WDL and end after WDL
Start bits cut off by BOT or Sync Pulse (frame starts without leading idle time)	Discard frame	NFI (if FEC is set) and empty frame stored (if FEC and VBS is set)
First ME in Messaging bits	If MSG is set: MEI	Frame is stored, bits after error are lost (“0”)
	If MSG cleared: NBI	
First ME in data, parity or CRC / Frame ends too early	NBI	Frame is stored, bits after error are lost (“0”)
Frame ends too late (longer than PDL)	no NBI	Bits after PDL are cut off and EOF is detected. Frame is stored.
Collision (Idle Time missing)	Discard 2nd frame	If the 1st frame was too late and running over WDL or If the 2nd frame was too early and running over WDL then for 2nd frame, NFI (if FEC is set) and empty frame stored (if FEC and VBS is set), Time Stamp = end of 2nd slot
In all cases, the receiver waits for an idle time followed by the start bits “00” before a new frame is detected.		

42.3.7 Bit Rate Generation

The bit rate of each channel is individually selectable from the 2 standard bit rates 125 kHz and 189 kHz. The two referring over sampling frequencies are generated centrally.

This chapter shows in detail how the bit rate for each channel is adjusted.

In the first stage, a global fractional divider serves as pre divider. Its intermediate frequency $f_{fracdiv}$ can be set up so that it is handy as input frequency for the different PSI5 channels. Usually this can be left unused.

For each of the two bit rates there is an own fractional divider. Each channel can select one out of the two clock signals f_{125} and f_{189}.

For details on the principles of a fractional divider, please refer to the SCU chapter of the device section “Clock Control”.

The PSI5 module provides 5 clock signals centrally (**Figure 648**):

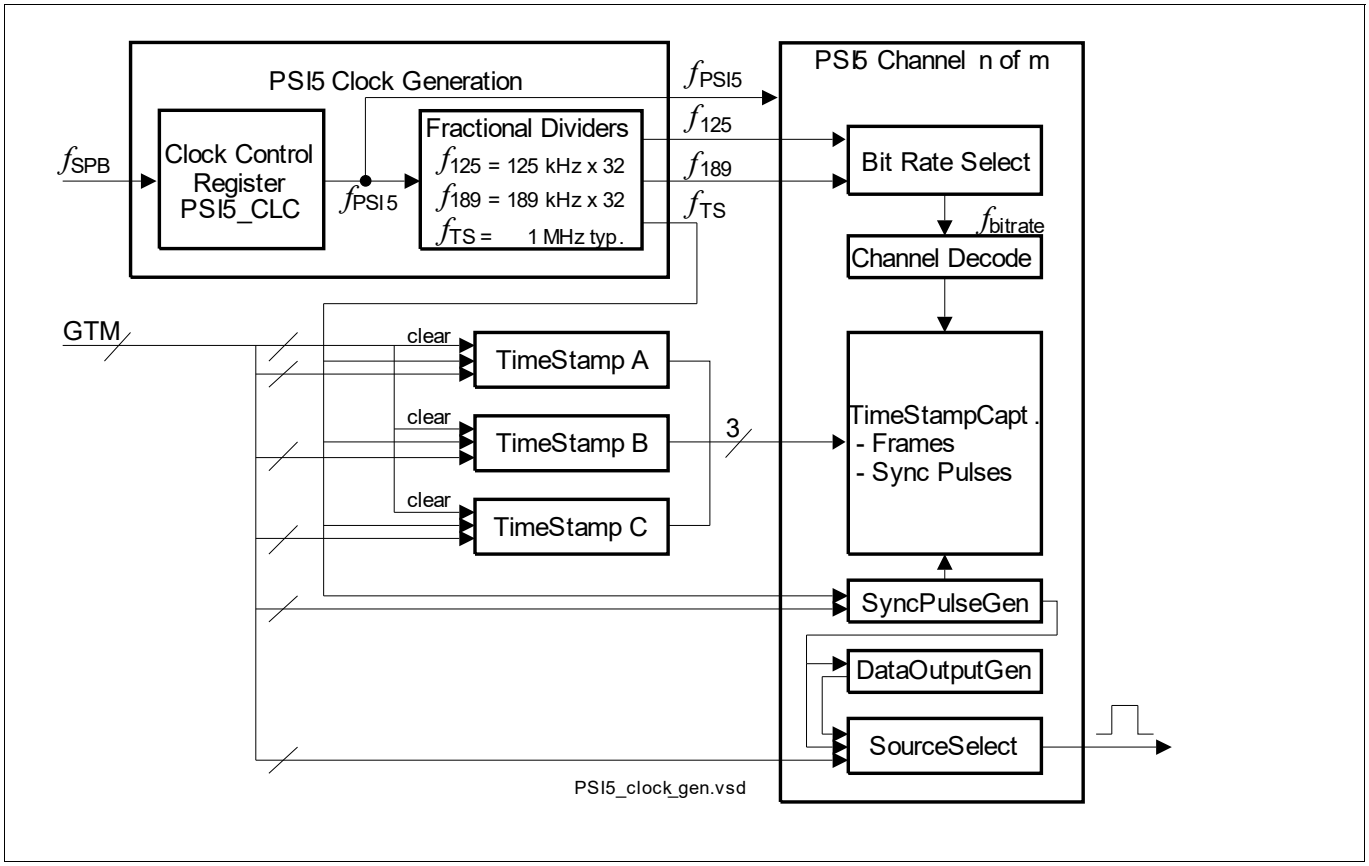


Figure 648 PSI5 Module Clock Generation

- f_{PSI5}
  This is the module clock that is used inside the PSI5 kernel for control purposes such as clocking of control logic and register operations. The frequency of f_{PSI5} is always identical to the system clock frequency f_{SPB}. The clock control register PSI5_CLC makes it possible to enable/disable f_{PSI5} under certain conditions.
- $f_{fracdiv}$
  This clock is the module clock that is used inside the PSI5 kernel for bit rate generation of the serial channels. The fractional divider register PSI5_FDR controls the frequency of $f_{fracdiv}$. Usually not required and set to 1.
- f_{125}
  This clock is the over sampling clock that can be selected for use inside a PSI5 channel as the bit rate frequency. The fractional divider register PSI5_FDRL controls the frequency of f_{125}.The clock drives the manchester decoder and needs to be 32 times faster than the nominal bus frequency
- f_{189}
  This clock is the over sampling clock that can be selected for use inside a PSI5 channel as the bit rate frequency. The fractional divider register PSI5_FDRH controls the frequency of f_{189}.The clock drives the manchester decoder and needs to be 32 times faster than the nominal bus frequency.
- f_{TS}
  This clock is used to drive the central time stamp counter.
  It can be selected to drive the global sync pulse time base and
  inside the PSI5 channels to drive the local watch dog timers. The fractional divider register PSI5_FDRT controls the frequency of f_{TS}.This is usually adjusted to 1 MHz / 1 µs period time.

The following two formulas define the frequency of $f_{fracdiv}$:

$f_{fracdiv} = f_{PSI5}$ / (1024 - PSI5_FDR.STEP); FDR.DM = 01B (42.1)

$f_{fracdiv} = f_{PSI5}$ x PSI5_FDR.STEP / 1024 with STEP = 0 … 1023; FDR.DM = 10B (42.2)

The following two formulas define the frequency of f_{125}.

$f_{125} = f_{fracdiv}$ / (1024 - PSI5_FDRL.STEP); FDRL.DM = 01B (42.3)

$f_{125} = f_{fracdiv}$ x PSI5_FDRL.STEP / 1024 with STEP = 0 … 1023; FDRL.DM = 10B (42.4)

The following two formulas define the frequency of f_{189}.

$f_{189} = f_{fracdiv}$ / (1024 - PSI5_FDRH.STEP); FDRH.DM = 01B (42.5)

$f_{189} = f_{fracdiv}$ x PSI5_FDRH.STEP / 1024 with STEP = 0 … 1023; FDRH.DM = 10B (42.6)

The following two formulas define the frequency of f_{TS}:

$f_{TS} = f_{PSI5}$ / (1024 - PSI5_FDRT.STEP); FDRT.DM = 01B (42.7)

$f_{TS} = f_{PSI5}$ x PSI5_FDRT.STEP / 1024 with STEP = 0 … 1023; FDRT.DM = 10B (42.8)

42.3.8 Digital Glitch Filter

Very slow slopes and signal noise can lead to fast transitions of the input signal. These unwanted transitions are suppressed by a digital glitch filter similar to a Filter and Prescaler Cell (FPC) of Infineons GPTA® in Delayed Debounce Filter Mode with up and down (no reset).

It is built for filtering very fast transitions only. The filter calculates the integral of the signal. If the integral reaches a programmable saturation point, the signal change is notified to the pulse measurement unit. Thus it helps to find the exact pulse length for said slow slopes in noisy environment.

The glitch filter is clocked with f_{PSI5}. If the state of the input sample differs from the current output signal value, the internal counter is incremented by one. When the state of the input sample matches the current output signal value and the timer is not in idle, the timer is decremented by one. When the timer matches the compare value stored in IOCRx.DEPTH, the level of the output signal line is inverted. The timer will be reset and set to idle state again.

The depth of the filter can be programmed to a value between 1 and 15. Typically a depth of 3 to 5 T_{PSI5} is sufficient. By default it is cleared. If IOCRx.DEPTH is cleared the filter is inactive/bypassed. Nevertheless, the input signal is sampled with f_{PSI5}.

The internal signal after the filter will change value not before the new value was sampled DEPTH times. If during this period a spike occurs, it takes 2 x T_{PSI5} times longer for the internal signal to change value. The filters principal implies a delay of the signal by (DEPTH x T_{PSI5}).

Upon detection of glitch during rising or falling edge, IOCRx.REG or IOCRx.FEG is set. The rising / falling edge glitch flags must be reset by software.

Figure 649 shows the digital glitch filter:

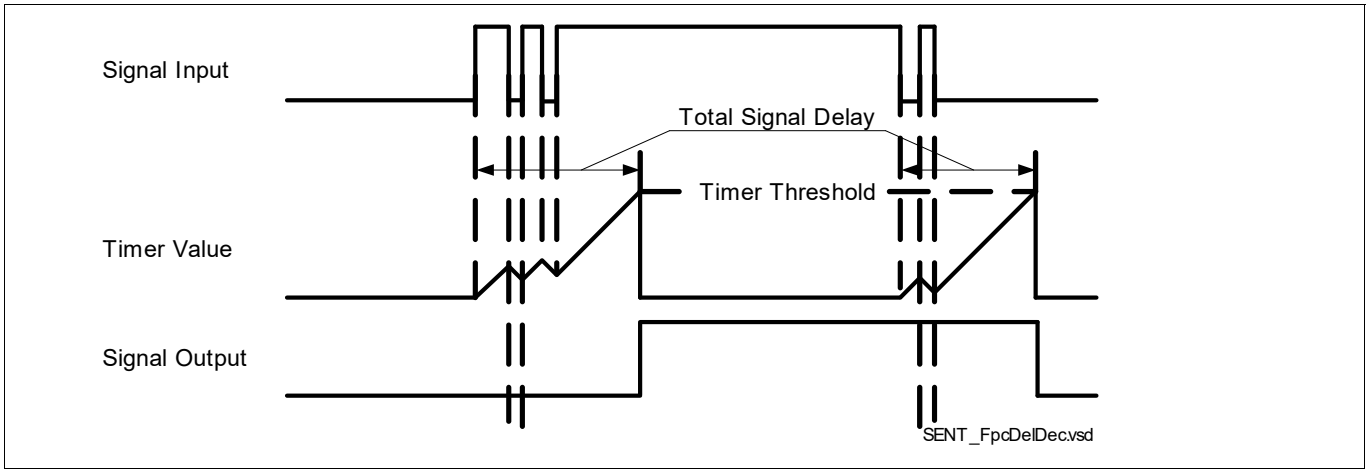


Figure 649 Digital Glitch Filter

42.3.9 Time Stamp Generation

Three time bases can be selected for Time Stamping:

Any of TSRA, TSRB and TSRC provide one time stamp counter. Each of them can be configured to be driven by either f_{TS} or by an external timer, i.e. one of the GTM inputs. The frequency of f_{TS} is controllable by the fractional divider in register FDRT.

All of them can be cleared by a selectable GTM signal or by SW, single or synchronously.

Two times are captured for each channel:

The last sync pulse sent on this channel is stored in the sync pulse time stamp capture register SPTSCx.

The last start of frame (SOF) on this channel is stored in the SOF capture register SFTSCx.

For each channel the capture time (SPTSC or SFTSC) can be selected, that will be stored in the Receive Data Register and the Receive Data Memory.

Figure 650 shows the time stamp generation:

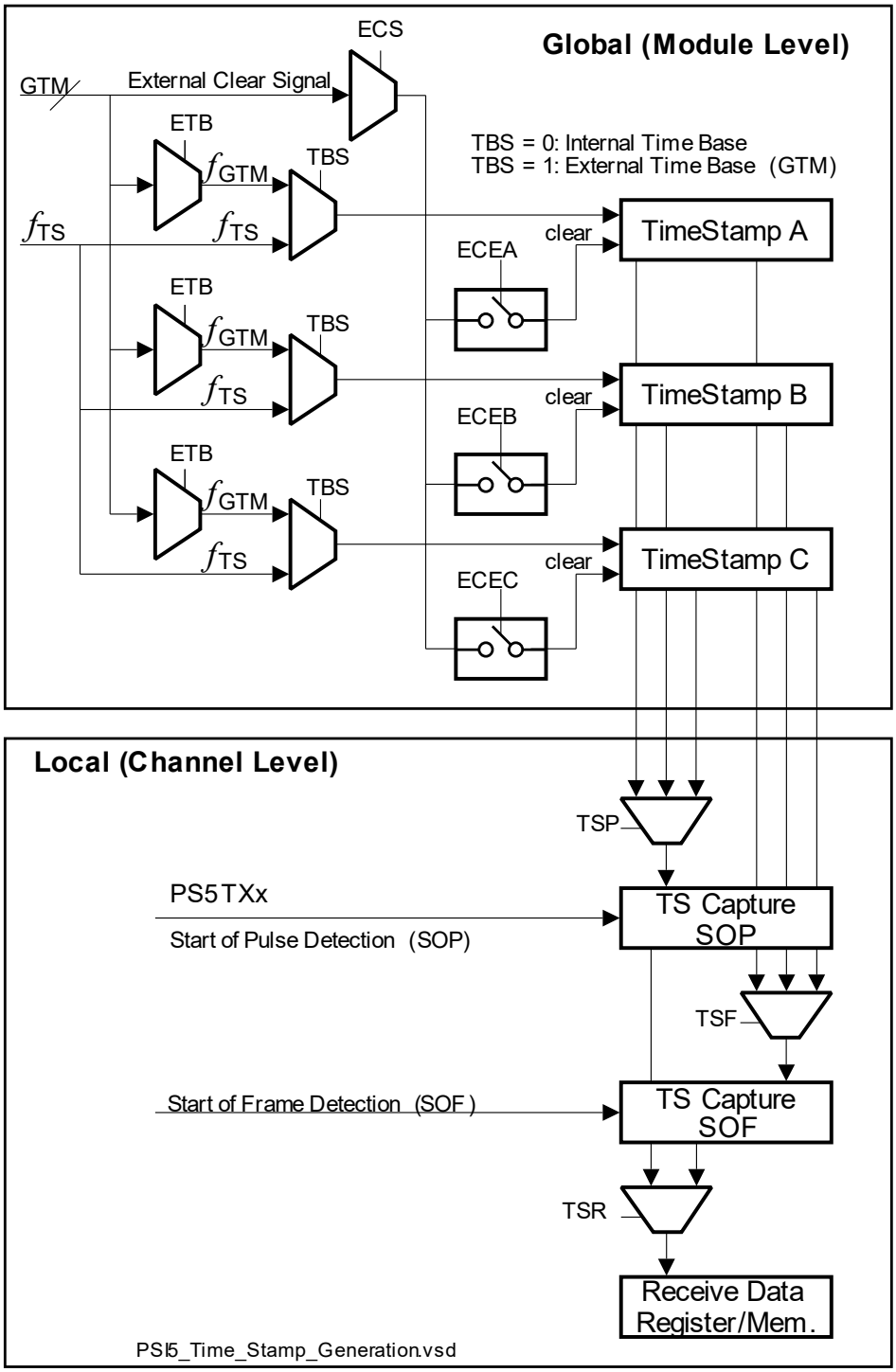


Figure 650 Time Stamp Generation

42.3.10 Watch Dog Timer

Synchronous Mode

Selected with RCRAx.ASYN = 0.

The Watch Dog Timer Registers WDTxy contain 7 time limits defining 6 frame slots.

The internal Watch Dog timer is started automatically at the start of a sync pulse. The value entered here defines the time in multiples of TTS (defined in FDRT) from the last start of a sync pulse on channel x.

The internal Watch Dog Timer is compared to WDLxy. A match triggers:

- the automatic incrementing of slot counter RDRH.SCx
- starting from WDLx1 on: check if a frame with the corresponding SC y has been received or if such a frame is still being received since last WDLxy expired. If this is not the case but a frame was expected in this slot y (configured so in RCRBx.FECy) “No Frame Received” Interrupt NFI is issued. If configured so (in RCRBx.VBSy), an empty frame (all data fields are ‘0’) is stored in RDRL/Hx and the referring RDML/H.
- Frames that start before a slot boundary and end after this boundary but are too short and thus discarded will result in an NFI / empty frame as well (if configured).
  - If no frame was received before the SOF in this slot,
    SC = slot where SOF happened.
    Time Stamp = end of slot where SOF happened.
  - If a good frame was received in this slot before a second SOF crosses WDL, the parameters for the next slot are taken over already and
    if no good frame is received in this next slot:
    SC = next slot (SC where SOF happened + 1)
    Time Stamp = end of next slot (SC where SOF happened + 1).
  - If a good frame was received in this slot before a second SOF crosses WDL, the parameters for the next slot are taken over already and
    if a good frame is received in this next slot: no NFI is issued.
- (The remarks about the Time Stamps assume that RCRCx.TSR selects SFTSCx.)
- A frame starting before a slot y and ending in slot y will not prevent NFI for slot y.

Detection of Frame Boundaries:

- SOF is the first rising edge of the first start bit after an “Idle Time”, a time stamp is triggered and SC is captured for the frame at this point in time
- EOF is determined based on the configured frame length (RCRAx.PDLy and RCRCx.CRCy). An unexpected “Idle Time” or Manchester Error will result in earlier EOF detection.
- For each frame, the receiver checks if slot counter SC at SOF is equal to SC at EOF. If not, the interrupt TEI is issued. FEC is not relevant for TEI.

If WDLxy is cleared no check is performed but RDRH.SC is still incremented when it is its turn in the sequence of the WDLs. Thus such unconfigured WDLxy must appear only at the end. If an unconfigured WDLxy appears between configured WDLxy, the slot count will be confused.

If a frame is received in a slot where no frame is expected, no dedicated flag is set but the frame is received with the parameters (PDL, SC and on) given for this slot. SW will detect this as SC for this frame is wrong.

Asynchronous Mode

Selected with RCRAx.ASYN = 1.
The watch dog timers for the PSI5 channel x, slot WDL1 .. 6 are disabled. If WDL0 is not cleared TEI is issued if the distance between two frames is longer than specified in WDL0. Slot Counter SC is incremented with each received frame (works as frame counter) with roll over after 6. If AVBS (verbose mode) is set too, an empty frame is stored in RDRL/Hx and the referring RDML/H.

Channel Configuration at Slot Boundaries / Take over of Slot Parameters

The receiver takes over the parameters for a frame at the beginning of a slot
- with 3 exceptions:

Exception 1)

During SLOT0 (this is the slot after the last configured slot), after the sync pulse, the configuration of SLOT1 is already used. An internal shadow slot count value makes sure, the messaging bits are routed to slow channel for SLOT1. This way, a frame for SLOT1 that is too early will be received correctly, the messaging bits are sorted correctly. The time violation is still visible by TEI = 1 as such a frame would still start in a different slot (SLOT0) than it ends (SLOT1). SC = 1 will be stored in the receive buffer/memory.

Exception 2)

Frames that come too early could destroy the slow channel of the preceding slot. To prevent this, during SLOT1 .. SLOT5, the configuration of the next slot is loaded immediately after EOF was received. Note that an SOF must be seen first in the same slot before an EOF is looked for. This way, the HW prevents the destruction of a good frame by a preceding frame that is too late. An internal shadow slot count value makes sure, the messaging bits are routed to the slow channel for the following SLOT. This way, a frame for e.g. SLOT3 that is too early will be received correctly, the messaging bits are sorted correctly in SLOT3 and not in SLOT2 where the SOF of this frame is. The time violation is still visible by TEI = 1 as such a frame would still start in a different slot (SLOT2) than it ends (SLOT3). SC = 3 will be stored in the receive buffer/memory.

Exception 3)

If a frame does not end at the configured end of a slot, the new parameters of the following slot are taken over by the receiver only after the EOF was detected. This allows to receive a frame properly that is too late.

Note that orphan frames, i.e. frames that come with a timely shift of of one ore more slots can not be detected as such. Thus they will be stored with the SC as they appear.

Notes on Frames coming too early

A frame may come too early and thus start in a slot that is configured for no frame (FEC = 0).

If the user wants to make sure that this frame is stored in RDR/RDM, the slot with FEC = 0 must be configured in the same way as its follower.

At least, the length field PDL must be set to its maximum value (28) to allow for reception of all frame lengths. This allows to capture such a frame with referring error bits. (TEI, NBI, CRC in most cases)

If this is not done and PDL is (by default) = 0 the following happens:

CASE1: SOF is too early and thus in the slot with FEC = 0, due to PDL = 0 also EOF is detected in the slot with FEC = 0.

The Frame is stored in RDR/RDM with Time Stamp, 0 bits of data (as configured in PDL), Slot Count of the too early slot (with FEC =0), CRCI is generated as the FSM assumes that no CRC can be checked ok on no data, NBI is issued.

CASE2: SOF is too early and thus in the slot with FEC = 0, although PDL = 0 EOF is detected in the slot FOLLOWING the slot with FEC = 0 (because CRC transmission end here, i.e. the SOF was very close to the end of the slot)

The Frame is stored in RDR/RDM with Time Stamp, 0 bits of data (as configured in PDL), Slot Count of the too early slot (with FEC =0), CRCI is generated as the FSM assumes that no CRC can be checked ok on no data, NBI and TEI bits are set.

Usage of the Watch Dog Timer Limits

The example below shows some use cases.

F1: “Frame too early”: Slot Count SC at SOF is 0 and 1 at EOF. Timing Error is issued (TEI). F1 is stored with configuration of Slot 1 and SC = 1.

WDL2: Slot Count SC incremented by HW, no “No Receive Frame” Interrupt as no frame expected (FEC=0)

F3: “Frame too late”: Slot Count SC at SOF is 3 and 4 at EOF. Timing Error is issued (TEI). The configuration for Slot 3 is kept until the end of F3 so that it is not destroyed. Note that the EOF of Frame 3 does not lead to take over the configuration of Slot 5! But the EOF of F4 does. This way a F5 coming too early would be received correctly.

WDL5: “Frame missing”: No SOF or EOF received since WDL4. A frame was expected (FEC=1), No Receive Frame Interrupt is issued.

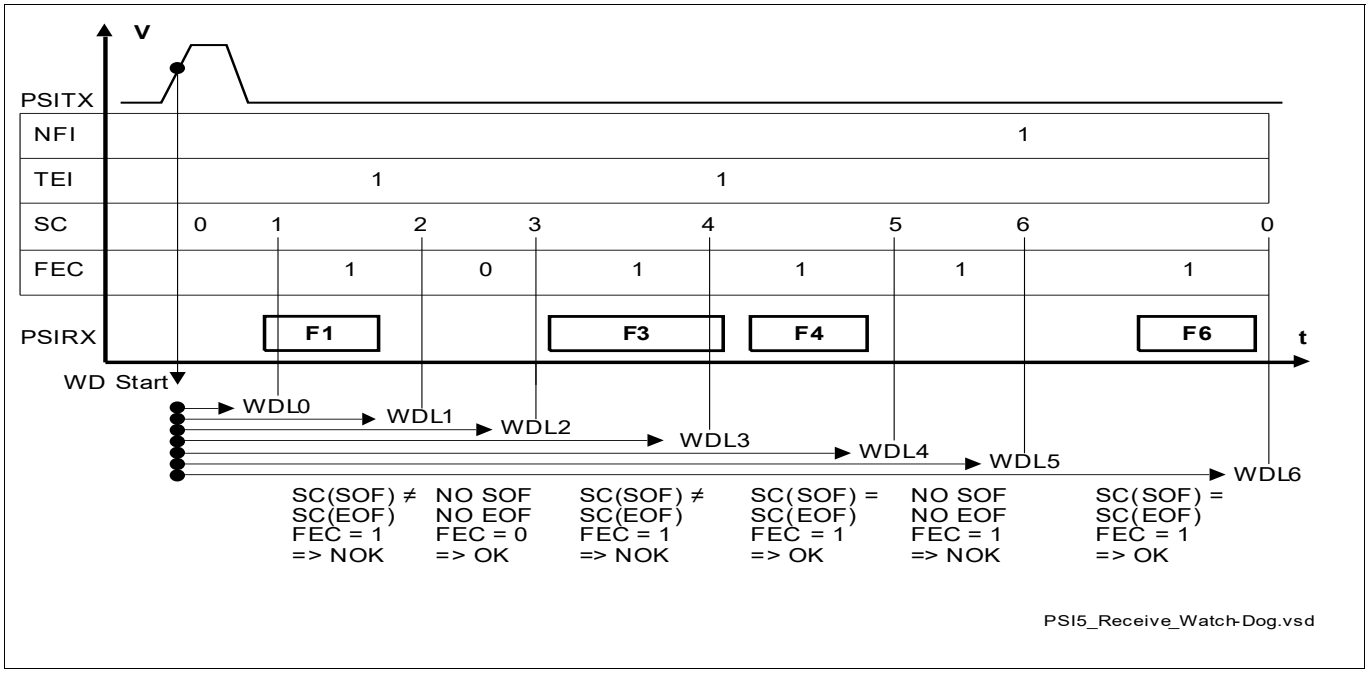


Figure 651 Watch Dog Timer Limits

42.3.11 Receive Data Memory

The module design allows for usage in 2 ways without mode selection:

Ring Buffer Usage

The Receive Data Memory x is built as a ring buffer. I.e. the first frame is stored in RDML/Hx0, the next in RDML/Hx1 and so on. At buffer 31 a wrap around is performed automatically. I.e. the next frame is stored in buffer 0. RFCx.WRP is showing the last location written and automatically incremented after write.

Receive Buffer Overrun Interrupt Flag RBI is set, if a buffer is written before the SW cleared Receive Data Interrupt Overview Flag RDIOVx (unchecked or erroneous frame received or No Frame Received (NFI) message to be stored).

FIFO Usage

Writing to the Receive Data Memory by the module moves the write pointer after writing.
Reading from RDFx 2 times with a 32 bit word access moves the read pointer to the next younger buffer automatically after the read access. It clears as well all interrupt overview bits for this buffer line in NBIOVx, CRCIOVx, TEIOVx, RMIOVx, RSIOVx; RDIOVx, NFIOVx, MEIOVx. This allows for fully automatic DMA transfers without CPU intervention. The DMA simply needs to be set up in a way that it reads exactly the number of buffers configured with the FIFO warning level FWL. The application SW configures “FIFO Warning Level Request Interrupt” FWI so that it is routed to the referring DMA channel as trigger source. Note that FWI does not need to be cleared to repetitively trigger the DMA.

If RFCx.WRAP is cleared and the read pointer (after the auto increment said above) equals the write pointer while reading from RDFx, a Receive Memory Underrun Interrupt RUIx is flagged and the read pointer is not moved on automatically even if RDFx is read.

IF RFCx.FRQ is set, the FIFO needs to be flushed by RFCx.FLU.

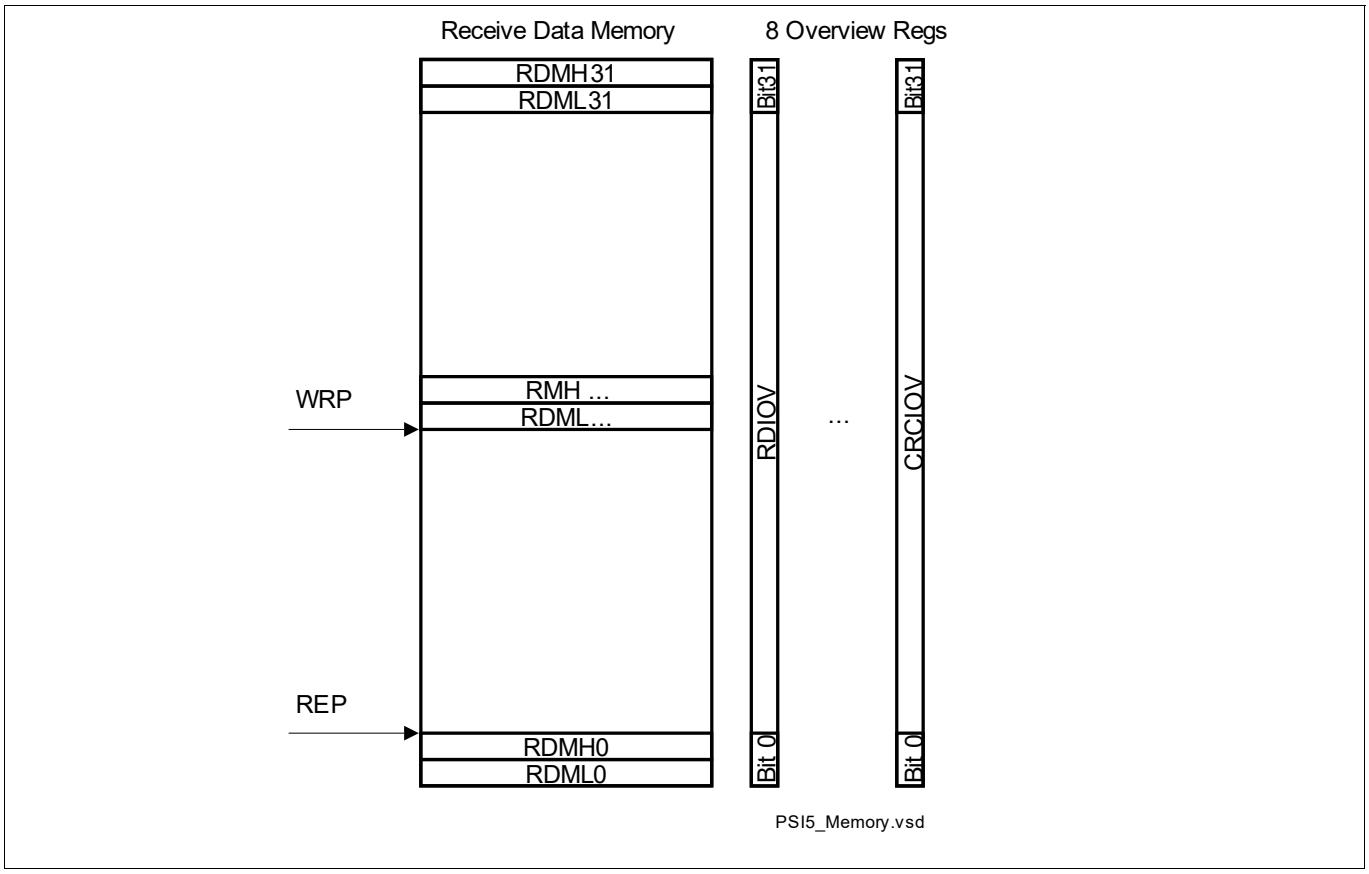


Figure 652 Receive Data Memory

42.3.12 Sync Pulse Control

The sync pulse generation provides the following functionality:

- periodic sync pulses, driven by a common module time base (PTE is set)
- angle synchronous sync pulses, driven by GTM (ETE is set)
- sync pulses for ECU to Sensor communication (PSI5 V1.3) (EPS is cleared)
- sync pulses for ECU to Sensor communication (PSI5 enhanced) (EPS is set)
- sync pulses provided by GTM (Bypass BYP set)
- Blank Out (switch off receiver during and after the Sync Pulse)

It is possible to switch on and off the periodic trigger or the angle synchronous trigger at any time. If data is available (TOF is set) the pulses will be modulated according to the selected standard. EPS selects between extended standard with pulse length modulation and legacy mode according to PSI5 standard V1.3 with pulse dropping (“tooth gap”). The minimum pulse length is defined by PGC.PLEN for both standards. For the extended standard, the pulses are prolonged by PGC.DEL if a one is coded. Exception is the bypass path (BYP is set). These pulses are transmitted without any modulation.

Any combination of PTE, ETE and BYP can produce too long pulses!

An ECU to Sensor communication can be stopped at any point in time by setting the flush bit. This will clear the send register and flag TOF.

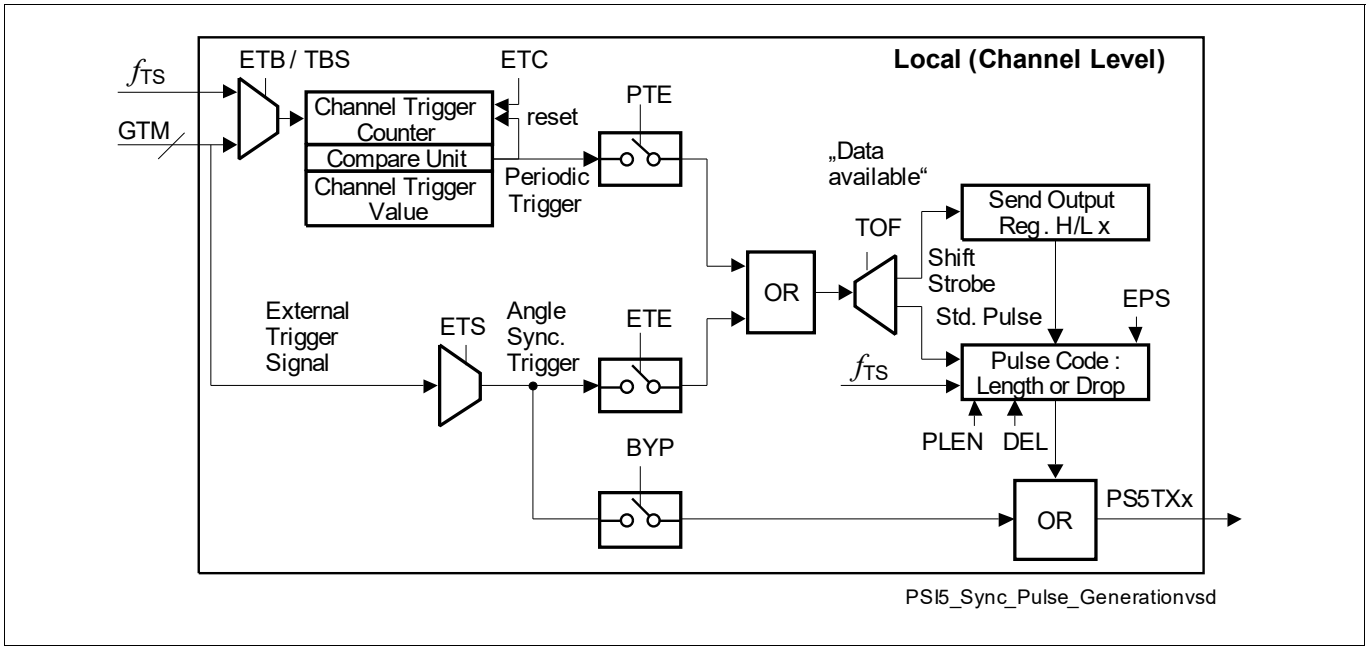


Figure 653 PSI5 Sync Pulse Generation

Blank Out Time

BOT defines the length of the blank out time in TTS times (defined by FDRT) starting from the end of a Sync Pulse. The receiver is always switched off starting from the beginning of a Sync Pulse. BOT keeps the receiver switched off additionally after the Sync Pulse. This suppresses potential noise on the receive line after the pulse and thus eases the design of the transceiver.

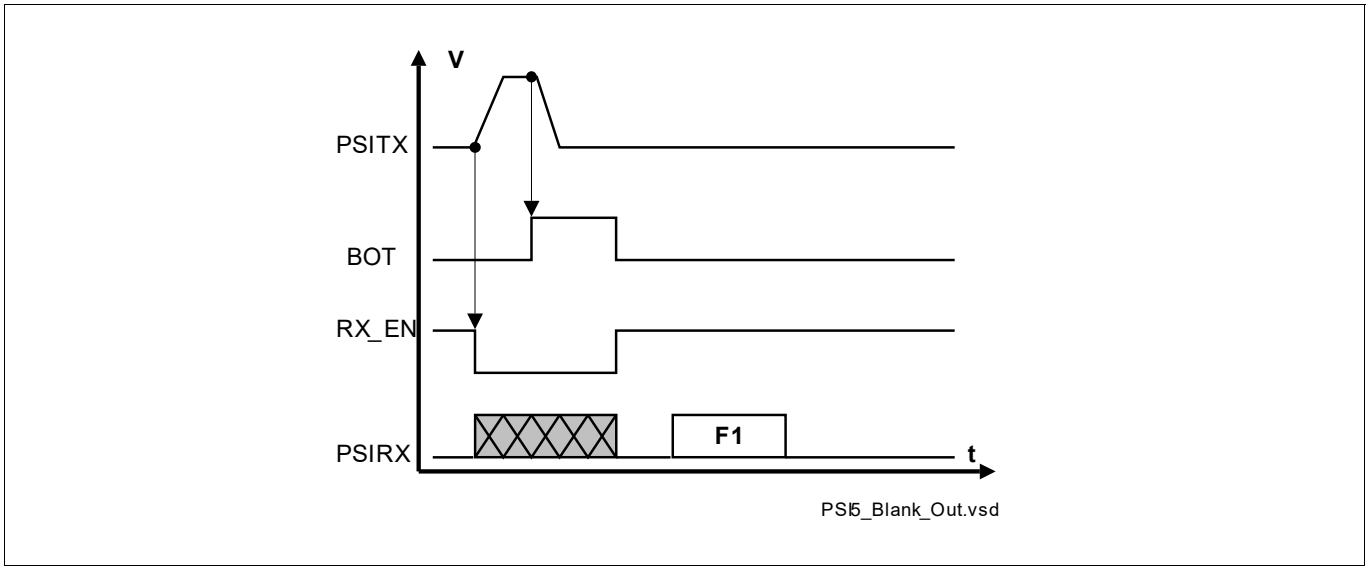


Figure 654 PSI5 Blank Out Time

42.3.13 Channel Trigger

The Channel Trigger Value Register x contains a two cell reset counter. It allows for setting up periodic sync pulses for channel x.

The Channel Trigger Value CTV is the compare value which clears the counter CTC on equality and only on equality. At exactly this time, a sync pulse is triggered.

The compare on equality allows to preset CTC and thus configure an offset between the CTVx.CTCs. This allows

for staggering periodic sync pulses that must have frequencies with common multiples (e.g. same frequency, double etc.).

In this document OFFSET = 65535 - “CTC preset value”

I.e. CTC preset is > CTV so that the time to to roll over is the offset.
Of course CTC preset can be < CTV so that this CTC has some advance.
Note that CTC can be written only if it is disabled by clearing GCR.ETCx.

Staggering the pulses might be required if the current source for the external phys can provide only limited current so that not all channels can be provided with the additional current that is required to apply the voltage increase representing a proper sync pulse.
For a synchronization of all counters CTVx.CTC, bits GCR.ETC0 .. 4 can be used.

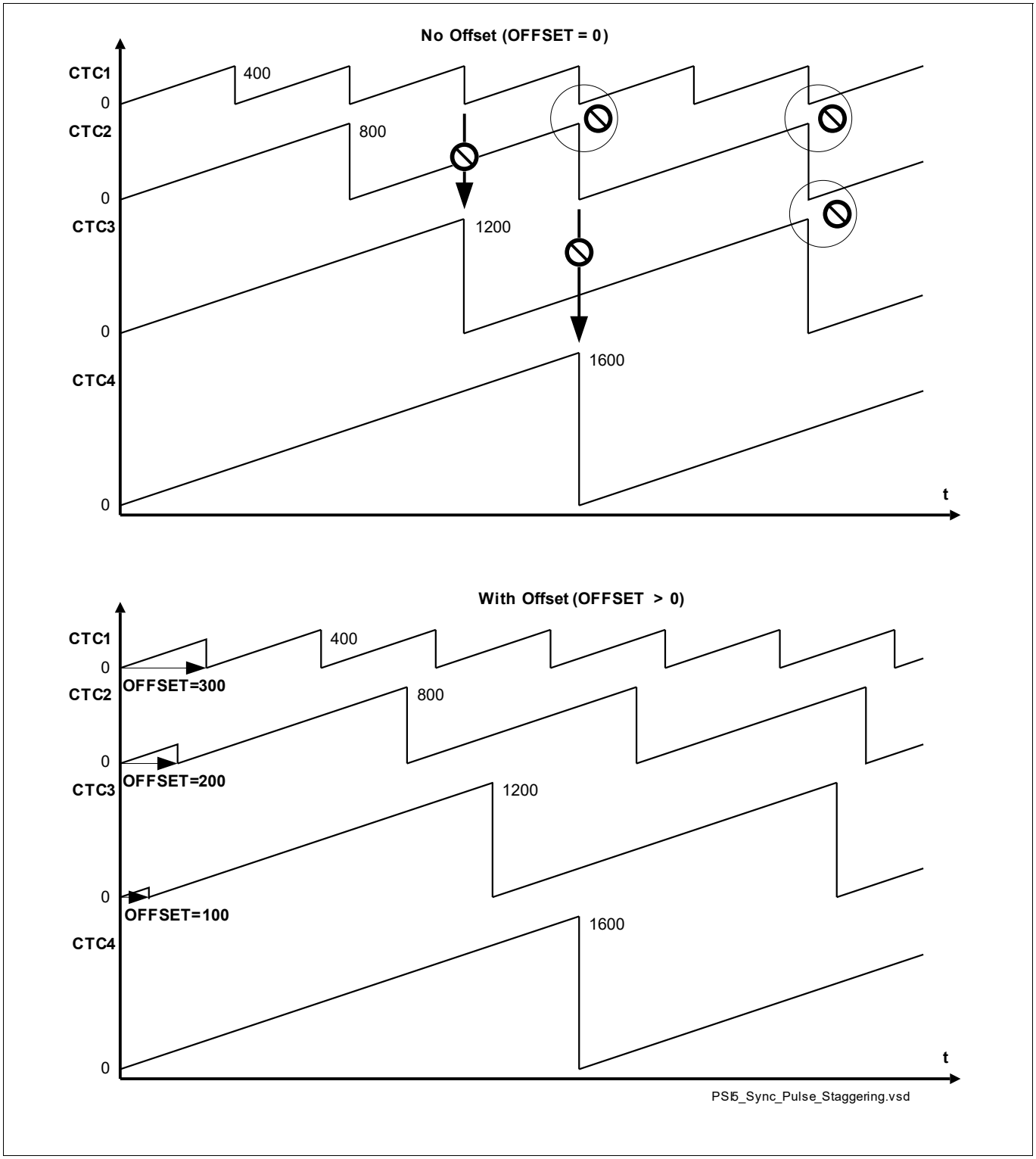


Figure 655 PSI5 Pulse Staggering

42.3.14 Send Control

The Send Control Register SCR contains control data bits required for sending data from the PSI5 module to the sensor / external PSI5 device.

Data is collected in SDRL/Hx and the control bits are collected in SCRx.

The send control unit is build to transmit complete ECU to sensor frames.

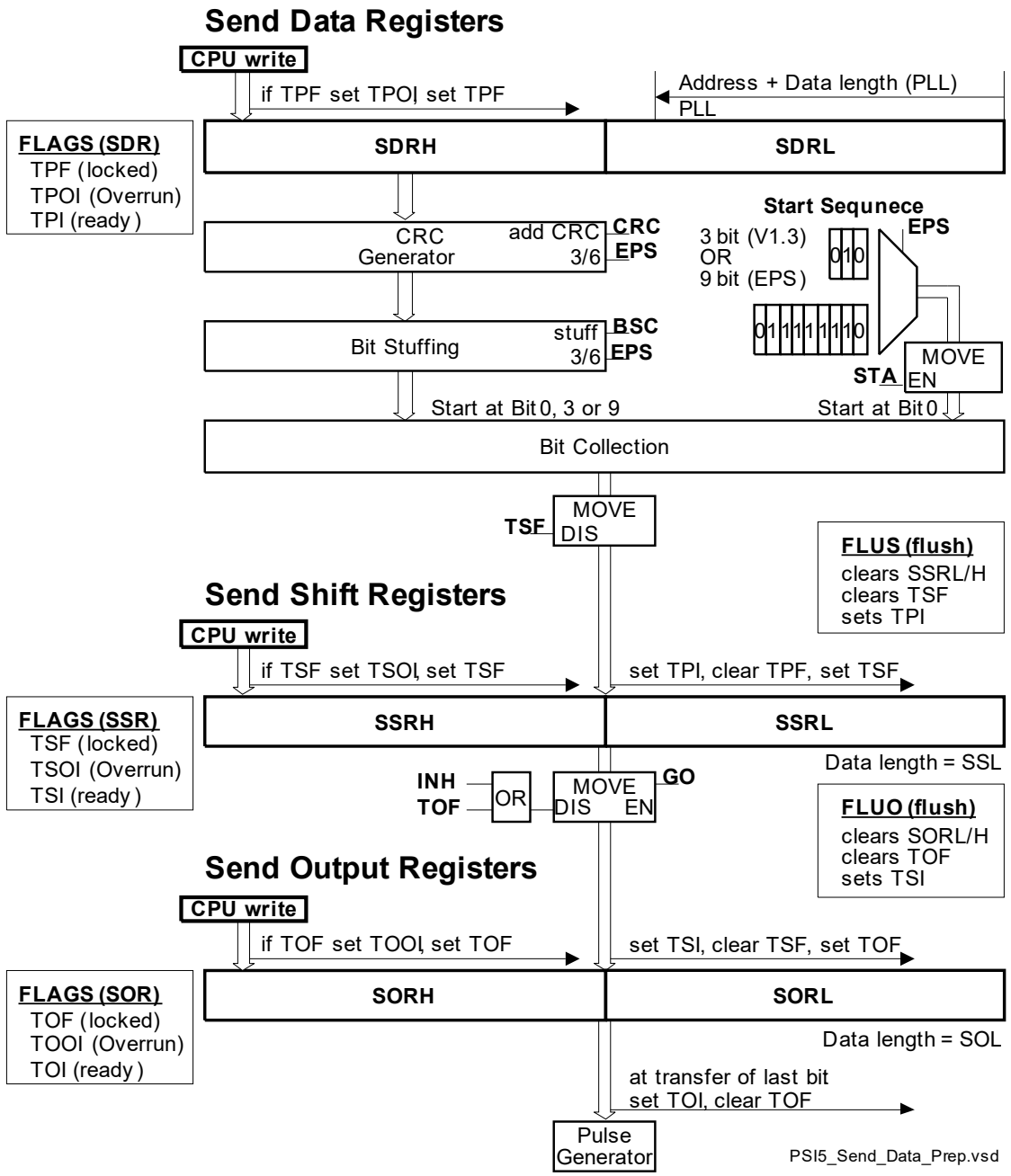


Figure 656 PSI5 Send Data Preparation

42.3.15 Error Detection Capabilities

Each PSI5 channel can detect and signal the following error conditions:

Protocol Level:

- No frame received / Manchester coding error in start bits (NFI)
- Messaging Bits with Manchester coding Error (MEI)
- Number of bits too small / Manchester coding error but messaging bits ok (NBI)
- Checksum error (CRCI)
- Frame not sent in expected time slot (TEI)
- Serial Communication: Wrong Status and Communication bit field (WSI)
- Serial Communication: CRC error (SCRI)

Transfer Management Level:

- Receive Data Buffer Overrun (RBI)
- Receive Data Memory Overrun (RMI)
- FIFO Warning Level (FWI)
- Receive Memory Underrun (RUI)
- ECU to Sensor Data Buffer Underrun (TBI)

42.3.16 Interrupts

8 Interrupt sources are available for the PSI5 module. For each trigger source of a channel x one interrupt can be selected in register INPx.

RDI indicates a receive data interrupt. It is activated when a received frame is moved to a Receive Data Register RDR. This happens always if at least the start bits were received without Manchester coding error.

RSI indicates a receive frame success interrupt, i.e. non of the interrupts NFI, MEI, NBI, CRCI, TEI, RBI, RMI was detected. The referring interrupt must be enabled in GCR to be considered for RSI. This allows e.g. to use only the Receive data register RDRL/Hx and ignore the Receive Data Memory RDML/Hx. Both RDI and RSI will be issued together in normal use cases where reception is correct.

RBI indicates a receive buffer overrun interrupt. It is activated when a new frame is transferred to a Receive Data Register RDR while the old value was still not read by the host (“overwrite”), i.e. the kernel wants to set any of the two interrupts RSI and RDI and finds any of these two interrupts already set.

RMI is issued if no memory buffer is available. RMI is set after a frame has been received or a No Frame Received (NFI) message is to be stored while RDIOVx.RDI[RFCx.WRP] is set. I.e. the kernel wants to set flag INTSTATAx.RDI[RFCx.WRP] and finds this flag already set. The old data is overwritten by the new data in the buffer memory.

FWI is set if the configured warning level of the FIFOx was reached. (See RFCx)

TPI, TSI and TOI indicate a transmit interrupt. They are activated when data is moved from a SDR to SSR (TPI), from SSR to SOR (TSI) and from SOR to the pulse generator (TOI).

TPOI, TSOI and TOOI indicate a transfer (send) register overrun interrupt. They are set if data is written to one of the registers in the transmission chain while the referring register is locked (occupied). The data written is ignored in this case.

In addition the protocol error interrupts are available:

TEI, NBI, MEI, CRCI. If one of the protocol interrupts is activated, data is to be treated as invalid according to standard V1.3. Note that the wrong data is still available in all buffers. In particular, on MEI the wrong data is still fed into the serial data analyzer (slow channel).

NFI is set, if a frame is missing in a slot configured by the watch dog timer. The module can be configured to store a complete message anyhow with flag NFI set. The slot count and the time stamp can be helpful for debugging. This provides a method for easy data handling, e.g. by using DMA transfers only without checking single interrupt bits.

WSI, SDI, SCRI treat the interrupts referring to the Status and Communication nibble.

For acceleration of the interrupt service routine, a Register INTOV is implemented that shows if an interrupt class (RSI, RDI, RBI, TDI, TBI, ERRI, SDI, FWI) is active on any of the channels. It holds a flag for each class which ORs the states from all channels of each node pointer in interrupt node pointer registers INPx. This flag is automatically set if there is an interrupt pending for the node pointer which is enabled. It is automatically reset, if no more enabled interrupt is pending for interrupt class.

In addition, there are 8 overview registers per channel holding the flags NBI, CRCI, TEI, RMI, RSI, RDI, NFI, MEI for each buffer line: NBIOVx, CRCIOVx, TEIOVx, RMIOVx, RSIOVx; RDIOVx, NFIOVx, MEIOVx; x = [3…0]. E.g. NBIOV0 holds 32 bits, one for each buffer line of channel 0 reflecting if NBI is set in this buffer line.

The interrupt request or the corresponding interrupt set bit (in register INTSETA/B) can trigger the interrupt generation at the selected interrupt node. The service request pulse is generated independently from the interrupt flag in register INTSTATA/Bx. The interrupt flag can be cleared by software by writing to the corresponding bit in register INTCLRA/B.
If more than one interrupt source is connected to the same interrupt node (in register INPx), the requests are combined to one common line.

42.3.17 Trigger Outputs

Any interrupt source can be used as trigger output TRIGO outside the module. Each TRIGO is connected to the interrupt router.

42.4 Registers

This section describes the kernel registers of the PSI5 module. All PSI5 kernel register names described in this section will be referenced in other parts of the document by the module name prefix “PSI5_” for the PSI5 interface.

All registers in the PSI5 address spaces are reset with the application reset with one exception: OCS is reset with debug reset only. (Definition see SCU section “Reset Operation”)

w can take the values 0 … 6 (watch dog limit values)

x can take the values 0 … 3 (PSI5 channels)

y can take the values 0 … 31

z can take the values 0 … 5

Table 427 Register Overview - PSI5 (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	000_H	SV,U	SV,E,P	Application Reset	30
	Reserved (004_H Byte)	004_H	BE	BE		
ID	Module Identification Register	008_H	SV,U	BE	Application Reset	31
FDR	PSI5 Fractional Divider Register	$00C_H$	SV,U	SV,E,P	Application Reset	32

Table 427 Register Overview - PSI5 (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
FDRL	Fractional Divider Register for Lower Bit Rate	010_H	SV,U	SV,E,P	Application Reset	32
FDRH	Fractional Divider Register for Higher Bit Rate	014_H	SV,U	SV,E,P	Application Reset	33
FDRT	Fractional Divider Register for Time Stamp	018_H	SV,U	SV,E,P	Application Reset	34
TSRA	Module Time Stamp Register A	$01C_H$	SV,U	SV,E,P	Application Reset	35
TSRB	Time Stamp Register B	020_H	SV,U	SV,E,P	Application Reset	36
TSRC	Module Time Stamp Register C	024_H	SV,U	SV,E,P	Application Reset	37
	Reserved (004_H Byte)	028_H	BE	BE		
GCR	Global Control Register	$02C_H$	SV,U	SV,E,P	Application Reset	38
IOCRx	Input and Output Control Register x	030_H+x* 90_H	SV,U	SV,E,P	Application Reset	39
RCRAx	Receiver Control Register A x	034_H+x* 90_H	SV,U	SV,E,P	Application Reset	41
RCRBx	Receiver Control Register B x	038_H+x* 90_H	SV,U	SV,E,P	Application Reset	42
RCRCx	Receiver Control Register C x	$03C_H$+x* 90_H	SV,U	SV,E,P	Application Reset	43
WDTxw	Watch Dog Timer Register xw	040_H+x* 90_H+w*4	SV,U	SV,E,P	Application Reset	44
RSRx	Receive Status Register x	$05C_H$+x* 90_H	SV,U	BE	Application Reset	44
SDSxz	Serial Data and Status Register xz	060_H+x* 90_H+z*4	SV,U	BE	Application Reset	45
SPTSCx	Start of Pulse Time Stamp Capture Register x	078_H+x* 90_H	SV,U	BE	Application Reset	46
SFTSCx	Start of Frame Time Stamp Capture Register x	$07C_H$+x* 90_H	SV,U	BE	Application Reset	46
RDRLx	Receive Data Register Low x	080_H+x* 90_H	SV,U	BE	Application Reset	47
RDRHx	Receive Data Register High x	084_H+x* 90_H	SV,U	BE	Application Reset	48
PGCx	Pulse Generation Control Register x	088_H+x* 90_H	SV,U	SV,E,P	Application Reset	49
CTVx	Channel Trigger Value Register x	$08C_H$+x* 90_H	SV,U	SV,E,P	Application Reset	51

Table 427 Register Overview - PSI5 (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
SCRx	Send Control Register x	090_H+x*90_H	SV,U	SV,E,P	Application Reset	52
SDRLx	Send Data Register Low x	094_H+x*90_H	SV,U	SV,U,P	Application Reset	55
SDRHx	Send Data Register High x	098_H+x*90_H	SV,U	SV,U,P	Application Reset	55
SSRLx	Send Shift Register Low x	$09C_H$+x*90_H	SV,U	SV,U,P	Application Reset	56
SSRHx	Send Shift Register High x	$0A0_H$+x*90_H	SV,U	SV,U,P	Application Reset	56
SORLx	Send Output Register Low x	$0A4_H$+x*90_H	SV,U	SV,U,P	Application Reset	57
SORHx	Send Output Register High x	$0A8_H$+x*90_H	SV,U	SV,U,P	Application Reset	57
INTOV	Interrupt Overview Register	$2F8_H$	SV,U	BE	Application Reset	58
INPx	Interrupt Node Pointer Register x	$2FC_H$+x*4	SV,U	SV,E,P	Application Reset	59
INTSTATAx	Interrupt Status Register A x	310_H+x*4	SV,U	BE	Application Reset	61
INTSTATBx	Interrupt Status Register B x	324_H+x*4	SV,U	BE	Application Reset	65
INTSETAx	Interrupt Set Register A x	338_H+x*4	nBE	SV,E,P	Application Reset	66
INTSETBx	Interrupt Set Register B x	$34C_H$+x*4	nBE	SV,E,P	Application Reset	68
INTCLRAx	Interrupt Clear Register A x	360_H+x*4	nBE	SV,U,P	Application Reset	69
INTCLRBx	Interrupt Clear Register A x	374_H+x*4	nBE	SV,U,P	Application Reset	71
INTENAx	Interrupt Enable Register A x	388_H+x*4	SV,U	SV,E,P	Application Reset	72
INTENBx	Interrupt Enable Register B x	$39C_H$+x*4	SV,U	SV,E,P	Application Reset	73
	Reserved ($01C_H$ Byte)	$3B0_H$	BE	BE		
OCS	OCDS Control and Status	$3CC_H$	U,SV	SV,P	Debug Reset	74
ACCEN0	Access Enable Register 0	$3D0_H$	U,SV	SV,SE	Application Reset	75
ACCEN1	Access Enable Register 1	$3D4_H$	U,SV	SV,SE	Application Reset	75

Table 427 Register Overview - PSI5 (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
KRST0	Kernel Reset Register 0	$3D8_H$	U,SV	SV,P,E	Application Reset	76
KRST1	Kernel Reset Register 1	$3DC_H$	U,SV	SV,P,E	Application Reset	77
KRSTCLR	Kernel Reset Status Clear Register	$3E0_H$	U,SV	SV,P,E	Application Reset	77
RFCx	Receive FIFO Control Register x	$3E4_H$+x*4	SV,U	SV,U,P	Application Reset	78
RDFx	Receive Data FIFO x	$3F8_H$+x*4	SV,U	BE	Application Reset	79
RSIOVx	RSI Overview Register x	$40C_H$+x*4	SV,U	BE	Application Reset	80
RMIOVx	RMI Overview Register x	420_H+x*4	SV,U	BE	Application Reset	80
NBIOVx	NBI Overview Register x	434_H+x*4	SV,U	BE	Application Reset	80
TEIOVx	TEI Overview Register x	448_H+x*4	SV,U	BE	Application Reset	81
CRCIOVx	CRCI Overview Register x	$45C_H$+x*4	SV,U	BE	Application Reset	81
RDIOVx	RDI Overview Register x	470_H+x*4	SV,U	BE	Application Reset	82
NFIOVx	NFI Overview Register x	484_H+x*4	SV,U	BE	Application Reset	82
MEIOVx	MEI Overview Register x	498_H+x*4	SV,U	BE	Application Reset	83
RSISETx	RSI Overview Set Register x	$4AC_H$+x*4	nBE	SV,U,P	Application Reset	83
RMISETx	RMI Overview Set Register x	$4C0_H$+x*4	nBE	SV,U,P	Application Reset	84
NBISETx	NBI Overview Set Register x	$4D4_H$+x*4	nBE	SV,U,P	Application Reset	84
TEISETx	TEI Overview Set Register x	$4E8_H$+x*4	nBE	SV,U,P	Application Reset	85
CRCISETx	CRCI Overview Set Register x	$4FC_H$+x*4	nBE	SV,U,P	Application Reset	85
RDISETx	RDI Overview Set Register x	510_H+x*4	nBE	SV,U,P	Application Reset	86
NFISETx	NFI Overview Set Register x	524_H+x*4	nBE	SV,U,P	Application Reset	86

Table 427 Register Overview - PSI5 (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
MEISETx	MEI Overview Set Register x	538_H+x*4	nBE	SV,U,P	Application Reset	87
RSICLRx	RSI Overview Clear Register x	$54C_H$+x*4	nBE	SV,U,P	Application Reset	87
RMICLRx	RMI Overview Clear Register x	560_H+x*4	nBE	SV,U,P	Application Reset	88
NBICLRx	NBI Overview Clear Register x	574_H+x*4	nBE	SV,U,P	Application Reset	88
TEICLRx	TEI Overview Clear Register x	588_H+x*4	nBE	SV,U,P	Application Reset	89
CRCICLRx	CRCI Overview Clear Register x	$59C_H$+x*4	nBE	SV,U,P	Application Reset	89
RDICLRx	RDI Overview Clear Register x	$5B0_H$+x*4	nBE	SV,U,P	Application Reset	90
NFICLRx	NFI Overview Clear Register x	$5C4_H$+x*4	nBE	SV,U,P	Application Reset	90
MEICLRx	MEI Overview Clear Register x	$5D8_H$+x*4	nBE	SV,U,P	Application Reset	91
	Reserved (014_H Byte)	$5EC_H$	BE	BE		
RDMLxy	Receive Data Memory Low xy	600_H+x*100_H+y*8	SV,U	BE	Application Reset	91
RDMHxy	Receive Data Memory High xy	604_H+x*100_H+y*8	SV,U	BE	Application Reset	92
	Reserved (100_H Byte)	$A00_H$	BE	BE		

Note:

3. *All registers belong to Application Reset except OCS, which belongs to Debug Reset.*
4. *Access to reserved addresses delivers bus error (BE).*
5. *Registers RDFx, RDMLxy, RDMHxy reflect RAM content, so reset does not change the content.*

42.4.1 Detailed Register Description

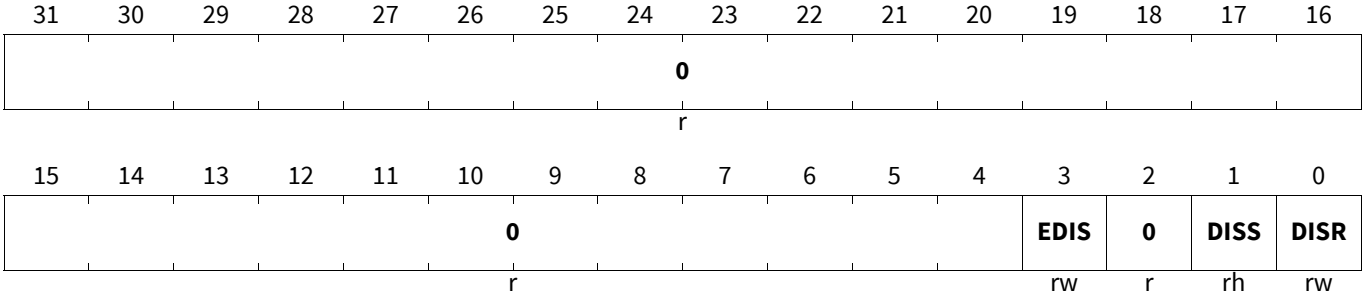
Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock, sleep mode and disable mode for the module.

Note: After a hardware reset operation, the f_{PSI5} clock is switched off and the PSI5 module is disabled (DISS set).

CLC

Clock Control Register **(000_H)** **Application Reset Value: 0000 0003_H**



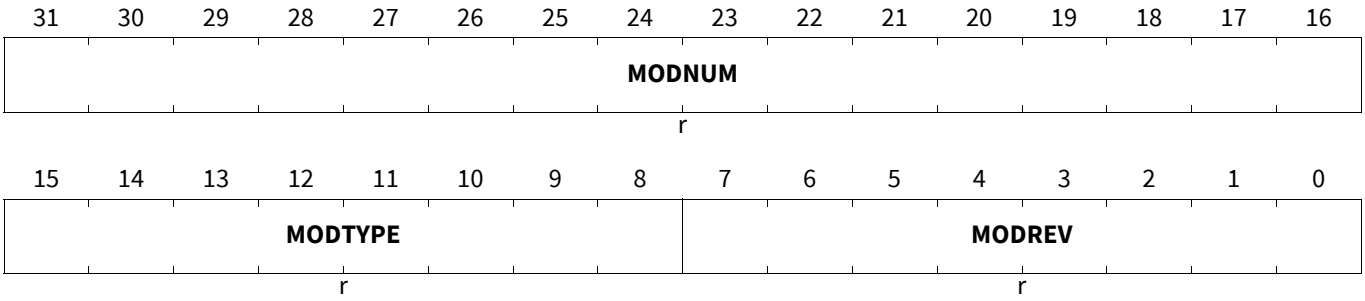
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**External Sleep Mode Request Disable Bit** Used to control module's sleep mode. *Note: If this bit is cleared the kernel clock f_{PSI5} is disabled during System Sleep Mode.*
0	2, 31:4	r	**Reserved** Read as 0; should be written with 0.

Module Identification Register

The PSI5 Module Identification Register ID contains read-only information about the module version.

ID

Module Identification Register **(008_H)** **Application Reset Value: 00C3 $C0XX_H$**



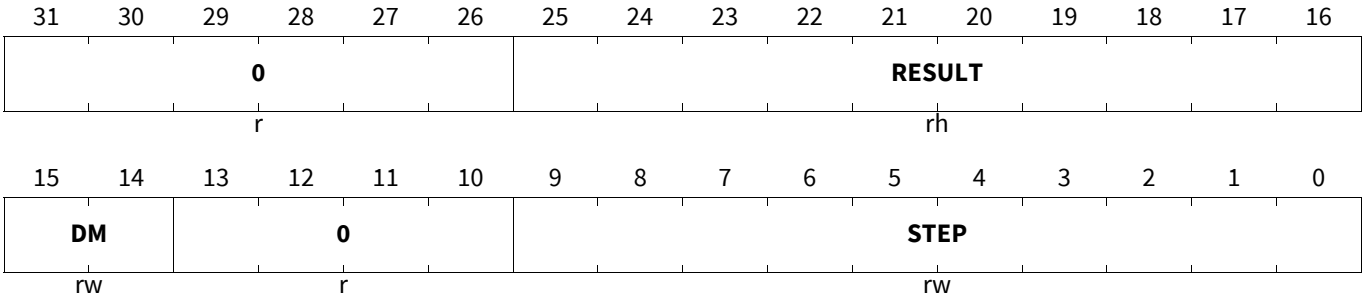
Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01_H (first revision).
MODTYPE	15:8	r	**Module Type** This bit field defines the module as a 32-bit module: $C0_H$

Field	Bits	Type	Description
MODNUM	31:16	r	**Module Number Value** This bit field defines the module identification number for the PSI5: $00C3_H$

PSI5 Fractional Divider Register

The Fractional Divider Register controls the input clock $f_{fracdiv}$.

FDR
PSI5 Fractional Divider Register ($00C_H$) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
STEP	9:0	rw	**Step Value** Reload or addition value for RESULT.
DM	15:14	rw	**Divider Mode** DM selects normal or fractional divider mode. 00_B Fractional divider is switched off; no output clock is generated. The Reset External Divider signal is 1. RESULT is not updated (default after System Reset). 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.
RESULT	25:16	rh	**Result Value** Bit field for the addition result.
0	13:10, 31:26	r	**Reserved** Read as 0; should be written with 0.

Fractional Divider Register for Lower Bit Rate

The Fractional Divider Register for lower Bit Rate contains the pre divider that defines the time resolution of the f_{125}. It divides $f_{fracdiv}$ by a factor as given in **Equation (42.3)** and **Equation (42.4)** and provides f_{125} to all channels.

FDRL

Fractional Divider Register for Lower Bit Rate (010$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						RESULT									
r						rh									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		0				STEP									
rw		r				rw									

Field	Bits	Type	Description
STEP	9:0	rw	**Step Value** Reload or addition value for RESULT.
DM	15:14	rw	**Divider Mode** DM selects normal or fractional divider mode. 00_B Fractional divider is switched off; no output clock is generated. 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.
RESULT	25:16	rh	**Result Value** Bit field for the addition result.
0	13:10, 31:26	r	**Reserved** Read as 0; should be written with 0.

Fractional Divider Register for Higher Bit Rate

The Fractional Divider Register for Higher Bit Rate contains the pre divider that defines the time resolution of f_{189}. It divides $f_{fracdiv}$ by a factor as given in **Equation (42.5)** and **Equation (42.6)** and provides f_{189} to all channels.

FDRH

Fractional Divider Register for Higher Bit Rate (014$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						RESULT									
r						rh									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		0				STEP									
rw		r				rw									

Field	Bits	Type	Description
STEP	9:0	rw	**Step Value** Reload or addition value for RESULT.

Field	Bits	Type	Description
DM	15:14	rw	**Divider Mode** DM selects normal or fractional divider mode. 00_B Fractional divider is switched off; no output clock is generated. 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.
RESULT	25:16	rh	**Result Value** Bit field for the addition result.
0	13:10, 31:26	r	**Reserved** Read as 0; should be written with 0.

Fractional Divider Register for Time Stamp

The PSI5 Module Time Stamp Predivider Register contains the pre divider that defines the time resolution of the Time Stamp Registers TSRA/B/C and the Sync Pulse Time Base Counter SBC. It divides f_{PSI5} by a factor as given in **Equation (42.7)** and **Equation (42.8)**. It contains as well the bits for reset control of the time stamp counters.

FDRT

Fractional Divider Register for Time Stamp (018$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECEC	ECEB	ECEA	ECS			RESULT									
rw	rw	rw	rw			rh									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		0				STEP									
rw		r				rw									

Field	Bits	Type	Description
STEP	9:0	rw	**Step Value** Reload or addition value for RESULT.
DM	15:14	rw	**Divider Mode** DM selects normal or fractional divider mode. 00_B Fractional divider is switched off; no output clock is generated. 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.
RESULT	25:16	rh	**Result Value** Bit field for the addition result.

Field	Bits	Type	Description
ECS	28:26	rw	**External Time Stamp Clear Source Select** Selects the external trigger line that clears the global time stamp counters TSRA/B/C.CTS if this is enabled by ECEA/ECEB/ECEC. Channel must be disabled if changed (GCR.CEN = 0). 000_B TRIG0 001_B TRIG1 … 101_B TRIG5 110_B reserved, no trigger selected 111_B reserved, no trigger selected
ECEA	29	rw	**External Time Stamp Clear Enable A** Enables the external trigger line selected by ECS to clear the global time stamp counter TSRA.CTS on rising edge of the external trigger. 0_B disabled 1_B enabled
ECEB	30	rw	**External Time Stamp Clear Enable B** Enables the external trigger line selected by ECS to clear the global time stamp counter TSRB.CTS on rising edge of the external trigger. 0_B disabled 1_B enabled
ECEC	31	rw	**External Time Stamp Clear Enable C** Enables the external trigger line selected by ECS to clear the global time stamp counter TSRC.CTS on rising edge of the external trigger. 0_B disabled 1_B enabled
0	13:10	r	**Reserved** Read as 0; should be written with 0.

Module Time Stamp Register A

This PSI5 Module Time Stamp Register contains read-only information about the current time given in clock cycles of f_{TS} or f_{GTM} since last reset while module was clocked. It also contains the control bits for selection of input clock and reset of this counter.

TSRA

Module Time Stamp Register A **($01C_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR	**ACLR**	**0**		**TBS**	**ETB**			**CTS**							
w	w	r		rw	rw			r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTS															
r															

Field	Bits	Type	Description
CTS	23:0	r	**Current Time Stamp for the Module** This bit field shows the current time stamp.
ETB	26:24	rw	**External Time Base Select** Selects the external clock line for counter CTS. Channel must be disabled if changed (GCR.CEN = 0). 000_B TRIG0 001_B TRIG1 … 101_B TRIG5 110_B reserved, no trigger selected 111_B reserved, no trigger selected
TBS	27	rw	**Time Base Select** This bit selects the clock source for CTS 0_B Internal, CTS counts in clock cycles of f_{TS} 1_B External, CTS counts in clock cycles of f_{GTM}
ACLR	30	w	**Clear All Current Time Stamp Counters** If set, this bit clears TSRA/B/C.CTS. TSRA/B/C.CTS count on, starting from 0. This bit is automatically cleared by HW and will always be read as zero.
CLR	31	w	**Clear Current Time Stamp for the Module** If set, this bit clears CTS. CTS counts on, starting from 0. This bit is automatically cleared by HW and will always be read as zero.
0	29:28	r	**Reserved** Read as 0; should be written with 0.

Time Stamp Register B

This PSI5 Module Time Stamp Register contains read-only information about the current time given in clock cycles of f_{TS} or f_{GTM} since last reset while module was clocked. It also contains the control bits for selection of input clock and reset of this counter.

TSRB

Time Stamp Register B **(020_H)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR	**ACLR**	**0**		**TBS**	**ETB**			**CTS**							
w	w	r		rw	rw			r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTS															
r															

Field	Bits	Type	Description
CTS	23:0	r	**Current Time Stamp for the Module** This bit field shows the current time stamp.

Field	Bits	Type	Description
ETB	26:24	rw	**External Time Base Select** Selects the external clock line for counter CTS. Channel must be disabled if changed (GCR.CEN = 0). 000_B TRIG0 001_B TRIG1 … 101_B TRIG5 110_B reserved, no trigger selected 111_B reserved, no trigger selected
TBS	27	rw	**Time Base Select** This bit selects the clock source for CTS 0_B Internal, CTS counts in clock cycles of f_{TS} 1_B External, CTS counts in clock cycles of f_{GTM}
ACLR	30	w	**Clear All Current Time Stamp Counters** If set, this bit clears TSRA/B/C.CTS. TSRA/B/C.CTS count on, starting from 0. This bit is automatically cleared by HW and will always be read as zero.
CLR	31	w	**Clear Current Time Stamp for the Module** If set, this bit clears CTS. CTS counts on, starting from 0. This bit is automatically cleared by HW and will always be read as zero.
0	29:28	r	**Reserved** Read as 0; should be written with 0.

Module Time Stamp Register C

This PSI5 Module Time Stamp Register contains read-only information about the current time given in clock cycles of f_{TS} or f_{GTM} since last reset while module was clocked. It also contains the control bits for selection of input clock and reset of this counter.

TSRC

Module Time Stamp Register C **(024_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLR	**ACLR**	**0**		**TBS**	**ETB**			**CTS**							
w	w	r		rw	rw			r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTS															
r															

Field	Bits	Type	Description
CTS	23:0	r	**Current Time Stamp for the Module** This bit field shows the current time stamp.

Field	Bits	Type	Description
ETB	26:24	rw	**External Time Base Select** Selects the external clock line for counter CTS. Channel must be disabled if changed (GCR.CEN = 0). 000_B TRIG0 001_B TRIG1 … 101_B TRIG5 110_B reserved, no trigger selected 111_B reserved, no trigger selected
TBS	27	rw	**Time Base Select** This bit selects the clock source for CTS 0_B Internal, CTS counts in clock cycles of f_{TS} 1_B External, CTS counts in clock cycles of f_{GTM}
ACLR	30	w	**Clear All Current Time Stamp Counters** If set, this bit clears TSRA/B/C.CTS. TSRA/B/C.CTS count on, starting from 0. This bit is automatically cleared by HW and will always be read as zero.
CLR	31	w	**Clear Current Time Stamp for the Module** If set, this bit clears CTS. CTS counts on, starting from 0. This bit is automatically cleared by HW and will always be read as zero.
0	29:28	r	**Reserved** Read as 0; should be written with 0.

Global Control Register

The Global Control Register defines module wide settings.

The first 5 bits define, which error flags are regarded at RSI. RSI indicates that the referring frame is free of the selected errors. Each of these errors can be selected: NFI, TEI, NBI, MEI, CRCI.

GCR
Global Control Register **(02C$_H$)** **Application Reset Value: 0000 001F$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											CEN4	CEN3	CEN2	CEN1	CEN0
r											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			ETC4	ETC3	ETC2	ETC1	ETC0	0			TEI	NFI	MEI	NBI	CRCI
r			rw	rw	rw	rw	rw	r			rw	rw	rw	rw	rw

Field	Bits	Type	Description
CRCI	0	rw	**CRCI** is selected if bit is set.
NBI	1	rw	**NBI** is selected if bit is set.
MEI	2	rw	**MEI** is selected if bit is set.

Field	Bits	Type	Description
NFI	3	rw	**NFI** is selected if bit is set.
TEI	4	rw	**TEI** is selected if bit is set.
ETCx (x=0-4)	x+8	rw	**Enable Channel Trigger Counter CTVx.CTC** This bit enables CTVx.CTC. The bits ETC0 ..x can be set with one write access to synchronously start all counters. This is required for proper sync pulse staggering. If set, CTCx counts on, starting from its current value. CTCx can be written only if ETCx is cleared (stopped). In the device only ETC0…3 are available. All other must be written with 0, and always read 0.
CENx (x=0-4)	x+16	rw	**Enable Channel** This bit enables PSI5 Channel x. If cleared, all internal state machines of the receiver and the sender are forced to default idle state while all registers can be read and written. Used for configuration of a channel. In the device only CEN0…3 are available. All other must be written with 0, and always read 0.
0	7:5, 15:13, 31:21	r	**Reserved** Read as 0; should be written with 0.

Input and Output Control Register x

The Input and Output Control Register IOCRx determines for the PSI5 channel x:

for the receiver:

- the alternate input
- the filter depth
- the input signal polarity

for the transmitter

- the output signal polarity

IOCRx (x=0-3)

Input and Output Control Register x **(030_H+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXM	**RXM**	**0**													
rh	rh	r													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFEG	**CREG**	**FEG**	**REG**	**0**		**IIE**	**OIE**	**DEPTH**				**0**		**ALTI**	
w	w	rh	rh	r		rw	rw	rw				r		rw	

Field	Bits	Type	Description
ALTI	1:0	rw	**Alternate Input Select** Selects the alternate input for channel x: 00_B Alternate Input 0 selected 01_B Alternate Input 1 selected 10_B Alternate Input 2 selected 11_B Alternate Input 3 selected
DEPTH	7:4	rw	**Digital Glitch Filter Depth** DEPTH determines the number of port input samples clocked with *f*PSI5 that are taken into account for the calculation of the floating average. The higher DEPTH is chosen to be, the longer the glitches that are suppressed and the longer the delay of the input signal introduced by this filter. 0_H off, default 1_H 1 T_{PSI5} 2_H 2 … F_H 15
OIE	8	rw	**Output Inverter Enable Channel x** Selects the Pulse Polarity of the output of channel x 0_B Pulse polarity is not inverted 1_B Pulse polarity is inverted
IIE	9	rw	**Input Inverter Enable Channel x** Selects the Pulse Polarity of the input of channel x 0_B Pulse polarity is not inverted 1_B Pulse polarity is inverted
REG	12	rh	**Rising Edge Glitch Flag for Channel x** Shows the status of the glitch detection of channel x REG is cleared by setting CREG. 0_B No Glitch detected on rising edge 1_B Glitch detected on rising edge
FEG	13	rh	**Falling Edge Glitch Flag for Channel x** Shows the status of the glitch detection of channel x FEG is cleared by setting CFEG. 0_B No Glitch detected on falling edge 1_B Glitch detected on falling edge
CREG	14	w	**Clear Rising Edge Glitch Flag for Channel x** Clears the status flag REG CREG always read zero. 0_B REG is not cleared 1_B REG is cleared
CFEG	15	w	**Clear Falling Edge Glitch Flag for Channel x** Clears the status flag FEG CFEG always read zero. 0_B FEG is not cleared 1_B FEG is cleared

Field	Bits	Type	Description
RXM	30	rh	**Receive Monitor for Channel x** Shows the status of the receive signal of channel x after glitch filtering and inverted as specified by IIE. 0_B Current signal is low. 1_B Current signal is high.
TXM	31	rh	**Transmit Monitor for Channel x** Shows the status of the transmit signal of channel x inverted as specified by OIE. 0_B Current signal is low. 1_B Current signal is high.
0	3:2, 11:10, 29:16	r	**Reserved** Read as 0; should be written with 0.

Receiver Control Register A x

The Receiver Control Registers RCRA contains control bits/bit fields that are related to the PSI5 receiver operation. It configures the payload length of the up to 6 frames in the up to 6 slots.

RCRAx (x=0-3)
Receiver Control Register A x **(034_H+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AVBS	**ASYN**	**PDL5**					**PDL4**					**PDL3**			
rw	rw	rw					rw					rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDL3	**PDL2**					**PDL1**					**PDL0**				
rw	rw					rw					rw				

Field	Bits	Type	Description
PDLy (y=0-5)	5*y+4:5*y	rw	**Payload Data Length** PDL determines the number of data bits per frame that the PSI5 channel x is setup for in slot y. PDL does not include the start bits and the Parity/CRC bits. PDL includes the Messaging bits, the Frame Control bits and the Status bits. 00_H 0 bit data 01_H 1 bit data 02_H 2 bit data 03_H 3 bit data … $1C_H$ 28 bit data $1D_H$ 0 bit data … $1F_H$ 0 bit data

Field	Bits	Type	Description
ASYN	30	rw	**Asynchronous Mode** If set, • the watch dog timers for the PSI5 channel x, slot WDL1 .. 6 are disabled • If WDL0 is not cleared TEI is issued if the distance between two frames is longer than specified in WDL0 • Slot Counter SC is incremented with each received frame (works as frame counter) with roll over after 6. • No Sync Pulses generated, TX path is inactive 0_B off (default) 1_B on
AVBS	31	rw	**Verbose Mode for Asynchronous Mode** If set, and ASYN is set and WDL0 is > 0 an empty frame is stored in RDRL/Hx and the referring RDML/H each time, the watch dog timer for the PSI5 channel x without reception of a frame. 0_B off (default) 1_B on

Receiver Control Register B x

The Receiver Control Registers RCRBx contains control bits/bit fields that are related to the PSI5 receiver operation. It configures up to 6 frame types in the up to 6 slots.

RCRBx (x=0-3)
Receiver Control Register B x **(038_H+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								VBS5	FEC5	CRC5	MSG5	VBS4	FEC4	CRC4	MSG4
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VBS3	FEC3	CRC3	MSG3	VBS2	FEC2	CRC2	MSG2	VBS1	FEC1	CRC1	MSG1	VBS0	FEC0	CRC0	MSG0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
MSGy (y=0-5)	4*y	rw	**Messaging Bits** If set, the 2 Messaging bits are configured for the PSI5 channel x in slot y. If set, Enhanced Serial Message processing on bits D[1:0] is activated. (18 frames, 4 or 8 bit ID, 12 or 16 bit data, 6 bit CRC) If Messaging bits are transmitted they are always presented in RDRx independently from status of MSGy. I.e. RDRx always stores exactly the data that was received. 0_B No Messaging bits (default) 1_B 2 Messaging bits

Field	Bits	Type	Description
CRCy (y=0-5)	4*y+1	rw	**CRC or Parity Selection** If set, a 3 bit CRC checksum is expected for the PSI5 channel x in slot y. Else, 1 bit Parity is assumed. 0_B 1 Parity Bit is configured (default) 1_B 3 CRC bits are configured
FECy (y=0-5)	4*y+2	rw	**Frame Expectation Control** If set, a frame is expected for the PSI5 channel x in slot y. A No Frame Received error interrupt NFI is issued each time, the watch dog timer for the PSI5 channel x, slot y expires without reception of a SOF or EOF. 0_B No Message expected (default) 1_B A Message is expected, SC is checked
VBSy (y=0-5)	4*y+3	rw	**Verbose Mode** If set, and a message is expected for the PSI5 channel x in slot y (FECy) an empty frame is stored in RDRL/Hx and the referring RDML/H each time, the watch dog timer for the PSI5 channel x, slot y expires without reception of a frame. 0_B off (default) 1_B on
0	31:24	r	**Reserved** Read as 0; should be written with 0.

Receiver Control Register C x

The Receiver Control Registers RCRCx contains control bits/bit fields that are related to the PSI5 receiver operation. Bit Rate and Time Stamp Sources are configured here.

RCRCx (x=0-3)

Receiver Control Register C x **($03C_H$+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										TSR	TSF		TSP		BRS
r										rw	rw		rw		rw

Field	Bits	Type	Description
BRS	0	rw	**Bit Rate Select** 0_B f_{125} selected for channel x 1_B f_{189} selected for channel x

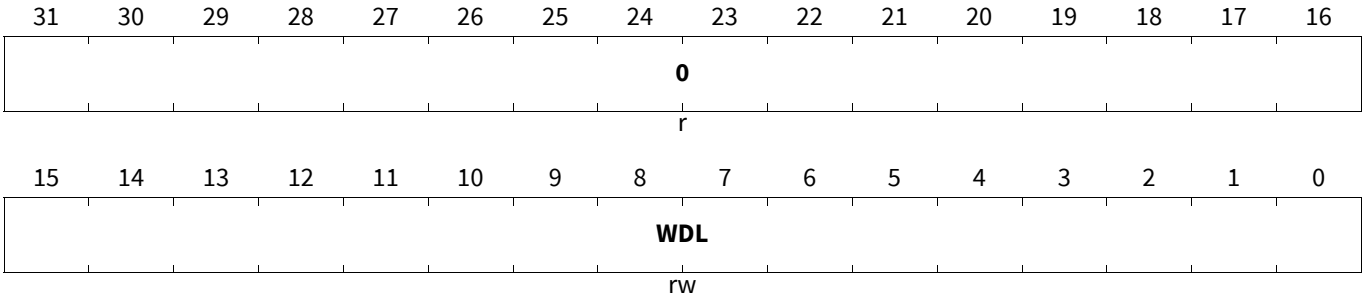
Field	Bits	Type	Description
TSP	2:1	rw	**Time Stamp Select for Pulses** 00_B TSRA.CTS is captured in SPTSC 01_B TSRB.CTS is captured in SPTSC 10_B TSRC.CTS is captured in SPTSC 11_B not defined. Do not use
TSF	4:3	rw	**Time Stamp Select for Start of Frame (SOF)** 00_B TSRA.CTS is captured in SFTSC 01_B TSRB.CTS is captured in SFTSC 10_B TSRC.CTS is captured in SFTSC 11_B not defined. Do not use
TSR	5	rw	**Time Stamp Select for Receive Data Registers** 0_B SPTSC is stored in RDRHx 1_B SFTSC is stored in RDRHx
0	31:6	r	**Reserved** Read as 0; should be written with 0.

Watch Dog Timer Register xw

For details, see chapter "Watch Dog Timer".

WDTxw (w=0-6;x=0-3)

Watch Dog Timer Register xw **(040_H+x*90_H+w*4)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
WDL	15:0	rw	**Watch Dog Timer Limit** for channel x limit w.
0	31:16	r	**Reserved** Read as 0; should be written with 0.

Receive Status Register x

The Receive Status Register provides the status information of channel x.

RSRx (x=0-3)

Receive Status Register x **($05C_H$+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						MSG		0					CRC		
r						r		r					r		

Field	Bits	Type	Description
CRC	2:0	r	**CRC** of last frame. CRC0 is on bit position 0.
MSG	9:8	r	**Messaging Bits** of last frame. MSG0 is on bit position 8.
0	7:3, 31:10	r	**Reserved** Read as 0; should be written with 0.

Serial Data and Status Register xz

The Serial (Receive) Data and Status Register provides the data and status information of channel x slot z. The data is presented in bit field SD independent from the SCRC check result in SCRI.

SDSxz (x=0-3;z=0-5)

Serial Data and Status Register xz **(060_H+x*90_H+z*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CON	SCRI	SCRC						MID							
r	r	r						r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD															
r															

Field	Bits	Type	Description
SD	15:0	r	**Serial Data** of last serial data frame on channel x slot z. SD0 is on bit position 0. If SDS.CON is not set, bits [15:12] are zero.
MID	23:16	r	**Message ID** of last serial data frame. ID0 is on bit position 16. If SDS.CON is set, bits [23:20] are zero.
SCRC	29:24	r	**SCRC** CRC of last serial data frame. CRC0 is on position 24.

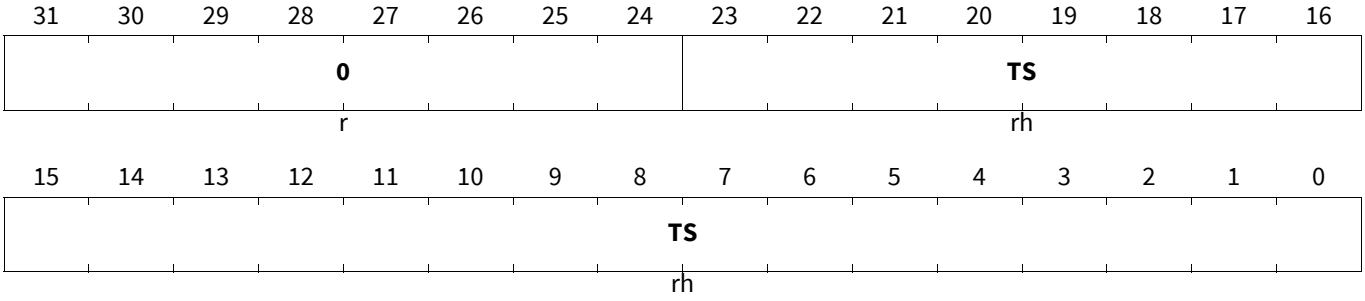
Field	Bits	Type	Description
SCRI	30	r	**CRC of Serial Message failed Interrupt Flag.** This bit is set if the CRC of the serial message fails. This includes a check of the Messaging bit field for correct 0 values of bit 1 in frames 7, 13 and 18. See INTSTATx.
CON	31	r	**Configuration bit** of last serial frame. 0_B 12-bit data and 8-bit message ID 1_B 16-bit data and 4-bit message ID

Start of Pulse Time Stamp Capture Register x

Each time a sync pulse is sent on channel x, the time stamp is captured in this register. RCRC.TSP selects, if TSRA, B or C is used. RCRC.TSR determines if SPTSCx or SFTSCx is stored in RDRHx.

SPTSCx (x=0-3)

Start of Pulse Time Stamp Capture Register x(078_H+x*90_H) **Application Reset Value: 0000 0000_H**



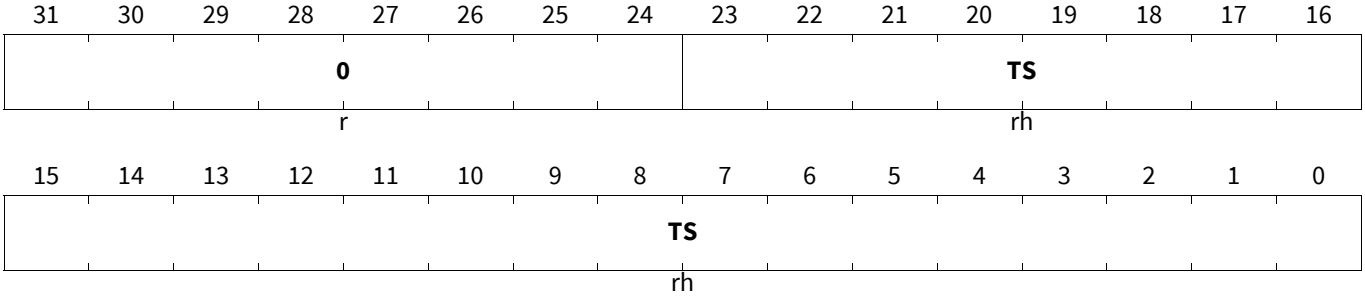
Field	Bits	Type	Description
TS	23:0	rh	**Time Stamp** of the last sync pulse sent on channel x. The Time Stamp is taken at the rising edge of the pulse.
0	31:24	r	**Reserved** Read as 0; should be written with 0.

Start of Frame Time Stamp Capture Register x

Each time a start of frame is received on channel x, the time stamp is captured in this register. The first rising edge of the first start bit is captured. Note that a preceding “Idle Time” is required for start bit detection. In case the two start bits should not be received successfully (i.e. without manchester coding errors) the frame is completely dropped and the latest captured value is overwritten with the next SOF or at the referring WDLy for “empty messages” if configured in RCRBx.VBSy. RCRC.TSF selects, if TSRA, B or C is used. RCRCx.TSR determines if SPTSCx or SFTSCx is stored in RDRHx.

SFTSCx (x=0-3)

Start of Frame Time Stamp Capture Register x($07C_H$+x*90_H) **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TS	23:0	rh	**Time Stamp** of the last frame received on channel x. The Time Stamp is taken at the rising edge of the first start bit S1 that was qualified by the receiver.
0	31:24	r	**Reserved** Read as 0; should be written with 0.

Receive Data Register Low x

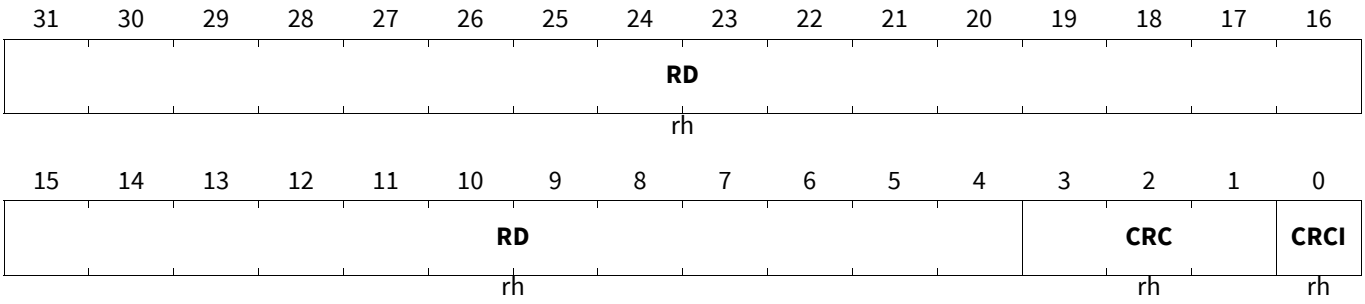
The Receive Data Register RDRLx for channel x shows the data content of a received data frame.

The internal receive buffer is always cleared (0x0000 0000_H) at each frame start. Thus unused nibbles are always read as zero.

The bit CRCI reflect, wether the conditions are met to issue an interrupt signal and set the referring interrupt status bit in INTSTATA for the latest frame. Thus it is independent from the old status of the referring sticky bit in INTSTATA before latest frame reception.

RDRLx (x=0-3)

Receive Data Register Low x **(080_H+x*90_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CRCI	0	rh	**CRC Error Flag** This bit is set if the CRC or Parity check fails. Set as well if CRC can not be calculated: PDL = 0, CRC cut off by Manchester Error, too few bits, BOT, Sync Pulse) 0_B correct CRC/Parity 1_B wrong CRC/Parity

Field	Bits	Type	Description
CRC	3:1	rh	**CRC** CRC/parity bit of last PSI5 frame. CRC0 or Parity bit is on bit position 1. In case of NFI, this field is cleared.
RD	31:4	rh	**RD** Receive data of last PSI5 frame. D0 is on bit position 4. In case of NFI, this field is cleared.

Receive Data Register High x

The Receive Data Register RDRHx for channel x shows the data content of a received data frame.

The internal receive buffer is always cleared (0x0000 0000_H) at each frame start. Thus unused bits are always read as zero.

The bits RBI, RMI, RSI, NBI and TEI reflect, wether the conditions are met to issue an interrupt signal and set the referring interrupt status bit in INTSTATA for the latest frame. Thus it is independent from the old status of the referring sticky bit in INTSTATA before latest frame reception.

RDRHx (x=0-3)

Receive Data Register High x **(084_H+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RBI	**NFI**	**MEI**	**NBI**	**TEI**	**SC**			**TS**							
rh	rh	rh	rh	rh		rh		rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS															
rh															

Field	Bits	Type	Description
TS	23:0	rh	**Time Stamp** of the last received PSI5 frame. RCRC.TSR determines if SPTSCx or SFTSCx is stored in RDRHx. (It can be selected if the time stamp of the last sync pulse is stored or the time stamp for the start of frame).
SC	26:24	rh	**Slot Counter** In Synchronous Mode (RCRAx.ASYN=0): Number of the slot the SOF of the frame was received in. In Asynchronous Mode (RCRAx.ASYN=1): revolving frame number between 1 and 6. 000_B not valid (SOF too early) 001_B Slot 1 … 110_B Slot 6 111_B not valid

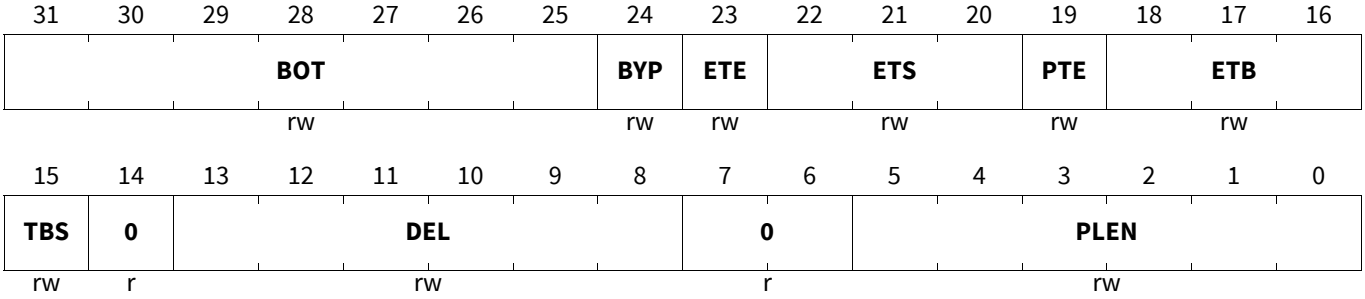
Field	Bits	Type	Description
TEI	27	rh	**Time Slot Error Flag** In Synchronous Mode (RCRAx.ASYN = 0), this bit is set if the SOF and the EOF of a frame are received in different time slots. The slots are constraint by the watch dog timer (see WDTxy). In Asynchronous Mode (RCRAx.ASYN = 1), this bit is set if the distance between two frames is longer than specified in WDL0. 0_B no error 1_B error
NBI	28	rh	**Number of bits Error Flag** This bit is set if the last frame received less bits than expected but no manchester coding error occurred until S1, S2, M0 and M1 where received. Note that the frame after the error might be completely wrong. 0_B all bits received 1_B not all bits received
MEI	29	rh	**Error in Message Bits Flag** This bit is set if a manchester error occurred in Bit M0 or M1. (Only if configured in RCRBx.MSG) Note that the frame after the error might be completely wrong. 0_B frame is correct 1_B frame is not correct
NFI	30	rh	**No Frame Received Flag** This bit is set when the NFI condition as described in the first paragraph of INTSTATA.NFI occurs. If only the two start bits are received and no further data, this “SOF” is discarded completely and treated like no frame received as well. If RCRBx.VBS is set, an empty frame is stored anyhow and RDIx is issued as well. The content of RDRL/Hx / RDML/Hx is valid for SC and for this bit (NFI). Bit fields RD and CRC are cleared (all zero). If RCRC.VBSx is 0 then TS is cleared. If RCRC.VBSx is 1 then TS is set to either SPTSCx.TS or SFTSCx.TS depending on the value of RCRC.TSR. 0_B no error 1_B error
RBI	31	rh	**Receive Buffer Overflow Flag** This bit is set after a frame has been received while the old one was not read from RDRHx. I.e. the kernel wants to set any of the two interrupts RSI and RDI and finds any of these two interrupts already set. The old data is overwritten by the new data. 0_B No overflow 1_B Overflow

Pulse Generation Control Register x

The Pulse Generation Control Register PGC contains control data for the sync pulse generation. It contains as well the trigger control bits required for sending data from the PSI5 module to the sensor / external PSI5 device.

PGCx (x=0-3)

Pulse Generation Control Register x **(088_H+x*90_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
PLEN	5:0	rw	**Pulse Length** Defines the length of the pulse in TTS times (defined by FDRT). This is the standard pulse width without data coding into the pulse width or for coding a ‘0’. 00_H Pulse length is 0 TTS 01_H Pulse length is 1 TTS … $3F_H$ Pulse length is 63 TTS
DEL	13:8	rw	**Delay Length** In case data is coded into the pulse length, this defines the ADDITONAL length of the pulse in TTS times. The resulting sum length is the pulse width for coding a ‘1’ into the pulse width. 00_H Pulse length is 0 TTS 01_H Pulse length is 1 TTS … $3F_H$ Pulse length is 63 TTS
TBS	15	rw	**Time Base Select** This bit selects the clock source for CTVx 0_B Internal, CTV counts in clock cycles of f_{TS} 1_B External, CTV counts in clock cycles of f_{GTM}
ETB	18:16	rw	**External Time Base Select** Selects the external clock line for counter CTVx. 000_B TRIG0 001_B TRIG1 … 101_B TRIG5 110_B reserved, no trigger selected 111_B reserved, no trigger selected
PTE	19	rw	**Periodic Trigger Enable** Periodic trigger is defined by CTVx. Should be 0 if ETE or BYP is set. 0_B disabled 1_B enabled

Field	Bits	Type	Description
ETS	22:20	rw	**External Trigger Select** Selects the external trigger line for pulse generation (e.g. angle synchronous). 000_B TRIG0 001_B TRIG1 … 101_B TRIG5 110_B reserved, no trigger selected 111_B reserved, no trigger selected
ETE	23	rw	**External Trigger Enable** “Angle sync. trigger”, external line is selected by ETS. Should be 0 if PTE or BYP is set. 0_B disabled 1_B enabled
BYP	24	rw	**Bypass Enable** An external signal, selected by ETS directly drives the output of channel x. Should be 0 if PTE or ETE is set. 0_B disabled 1_B enabled
BOT	31:25	rw	**Blank Out Time** BOT defines the length of the blank out time in TTS times (defined by FDRT) starting from the end of a Sync Pulse. The receiver is always switched off starting from the beginning of a Sync Pulse. BOT keeps the receiver switched off additionally after the Sync Pulse. This suppresses potential noise on the receive line after the pulse and thus eases the design of the transceiver. 00_H Blank Out Time is 0 TTS 01_H Blank Out Time is 1 TTS … $7F_H$ Blank Out Time is 127 TTS
0	7:6, 14	r	**Reserved** Read as 0; should be written with 0.

Channel Trigger Value Register x

The Channel Trigger Value Register x contains a two cell reset counter. It allows for setting up periodic sync pulses for channel x.

CTVx (x=0-3)

Channel Trigger Value Register x **($08C_H$+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTC															
rwh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTV															
rw															

Field	Bits	Type	Description
CTV	15:0	rw	**Channel Trigger Value CTV** Contains the compare value (exact match) of Channel Trigger CTC at which a sync pulse is triggered for channel x and the counter CTC is cleared. If cleared, CTC is stopped and no pulse is generated.
CTC	31:16	rwh	**Channel Trigger Counter** This bit field allows to read the current counter value of the reset timer cell CTVx. If GCR.ETCx is cleared, CTC can be written.

Send Control Register x

The Send Control Register SCR contains control data bits required for sending data from the PSI5 module to the sensor / external PSI5 device.

SCRx (x=0-3)
Send Control Register x **(090_H+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRQ	**0**		**TOF**	**TSF**	**TPF**	**GO**	**INH**	**STA**	**CRC**	**SOL**					
r	r		r	r	r	w	rw	rw	rw	rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLUO	**FLUS**	**SSL**						**BSC**	**EPS**	**PLL**					
w	w	rw						rw	rw	rw					

Field	Bits	Type	Description
PLL	5:0	rw	**Pay Load Length of Registers SDRL/H** If PLL > 31, SDRH needs to be written to trigger the HW for automatic STA, BSC or CRC generation or just moving data to SSRL/H. Else writing to SDRL is sufficient. PLL needs to written before SDRL/H is used for proper operation. If insertion of STA, BSC and CRC results in more than 64 bits, the MSBs are truncated in SSR. Defines the length that is taken into account: 00_H length is 1 01_H length is 2 … $3F_H$ length is 64
EPS	6	rw	**Enhanced Protocol Selection** 0_B PSI5 V1.3. No Pulse length extension 1_B Enhanced Protocol with Pulse length extension.

Field	Bits	Type	Description
BSC	7	rw	**Bit Stuffing Control** Depending from bit EPS after 3 bits a '1' is inserted (V1.3) or after 6 bits a '0' is inserted (enhanced Power Train Mode) Note that this makes the frame longer. 0_B No automatic bit stuffing 1_B Automatic bit stuffing is enabled.
SSL	13:8	rw	**Pay Load Length of Registers SSRL/H** If SSL > 31, SSRH needs to be written to trigger the HW for automatic moving data to SORL/H. Else writing to SSRL is sufficient. SSL needs to be written before SSRL/H is used for proper operation. Defines the length that is taken into account. Start Sequence, BSC and CRC need to be added to PLL by SW if SSL is calculated based on PLL. 00_H length is 1 01_H length is 2 … $3F_H$ length is 64
FLUS	14	w	**Flush SSRH/Lx** Setting this bit stops the shifting process and flushes (clears) SSRH/Lx and TSF. TPIx is issued at the end of successful flushing. Reads always as zero.
FLUO	15	w	**Flush SORH/Lx** Setting this bit stops the sending process and flushes (clears) SORH/Lx and TOF. TSIx is issued at the end of successful flushing. Reads always as zero.
SOL	21:16	rw	**Pay Load Length of Registers SORL/H** If SOL > 31, SORH needs to be written to trigger the HW for automatic moving data to the pulse generator. Else writing to SORL is sufficient. SOL needs to written before SORL/H is used for proper operation. Defines the length that is taken into account. Start Sequence, CRC and Stuffing needs to be added by SW to PLL if SOL is calculated based on PLL. 00_H length is 1 01_H length is 2 … $3F_H$ length is 64
CRC	22	rw	**CRC Generation Control** 0_B CRC is not generated automatically. 1_B CRC is automatically generated and added to the frame by HW (according to EPS). Note that this makes the frame longer. CRC calculation is not performed, if at least 1 CRC bit does not fit into 64 bit, e.g. due to STA or BSC. In this case, one or two zero bits are inserted.

Field	Bits	Type	Description
STA	23	rw	**Start Sequence Generation Control** 0_B no start sequence generated, payload starts at bit 0! 1_B automatically generated and added to the frame by HW (according to EPS) payload starts at bit 3/9 (EPS = 0/1). Note that this makes the frame longer.
INH	24	rw	**Inhibit Transfer** This inhibits the automatic transfer from the shift registers SSRL/H to SORL/H after the preparation of all automatic bit fields (STA, BSC, CRC). After a write access to SDRL/Hx, the automatic preparation is done by HW if selected. If automatic preparation of the start sequence is not selected, data is moved by HW without change to SSRL/H. When the HW is done with the transfer to the internal shift registers, interrupt TPI is activated, flag TSF is set and flag TPF is cleared automatically. Depending on INH, GO must be set after TPI before transfer to SORL/H takes place: After the transfer to SORL/H, TSI is activated, TOF is set and and TSF is cleared automatically. . INH defines the start mode: 0_B automatic transfer 1_B GO bit must be set before transfer
GO	25	w	**Release prepared Send data** This is only relevant if INH is set It allows for manual control of the transfer from SSRH/L to SORL/H. It is always read as zero and is automatically cleared. 0_B no transfer 1_B start transfer
TPF	26	r	**Transmit Preparation Flag** If set, preparation is in progress: start sequence and/ or CRC and/or stuffing and/or at least copying data from SDRL/H. If set, write access to SDRL/H will not change any data and issue TPOI. It is cleared automatically after preparation is finished.
TSF	27	r	**Transmit Shift Flag** If set, data in SSRL/H is waiting to be shifted to SORL/H. This can be caused by an automatic transfer from SDRL/H or from a CPU write access to SSRL/H. If SSL < 32 it is sufficient to write SSRL. Else, only writing SSRH will set TSF. If set, write access to SSRL/H will not change any data and issue TSOI. It is cleared automatically after data was shifted to SORL/H or after flushing the register by setting bit FLUS.
TOF	28	r	**Transmit Output Flag** If set, data in SORL/H is waiting to be transmitted to the sensor. This can be caused by an automatic transfer from SSRL/H or from a CPU write access to SORL/H. If SOL < 32 it is sufficient to write SORL. Else, only writing SDRH will set TSF. If set, write access to SORL/H will not change any data and issue TOOI. It is cleared automatically after data was transmitted to the sensor or after flushing the register by setting bit FLUO.

Field	Bits	Type	Description
TRQ	31	r	**Transfer Request in Progress** While a transfer is being sent this bit is set. Write access is ignored.
0	30:29	r	**Reserved** Read as 0; should be written with 0.

Send Data Register Low x

The Send Data Register SDRLx for channel x shows the data content of a data frame to be sent.

Data to be sent must be written while the channel is enabled (GCR.CEN is set). While CEN is cleared the HW can not detect the write access as all state machines are stopped.

SDRLx (x=0-3)

Send Data Register Low x **(094_H+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SD31	**SD30**	**SD29**	**SD28**	**SD27**	**SD26**	**SD25**	**SD24**	**SD23**	**SD22**	**SD21**	**SD20**	**SD19**	**SD18**	**SD17**	**SD16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD15	**SD14**	**SD13**	**SD12**	**SD11**	**SD10**	**SD9**	**SD8**	**SD7**	**SD6**	**SD5**	**SD4**	**SD3**	**SD2**	**SD1**	**SD0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SDy (y=0-31)	y	rw	**SDy** Send data of next ECU to Sensor frame. The unused MSBs (bit position SCRx.PLL and higher) must be written with '0'. If PLL is < 32, SDRHx must not be written.

Send Data Register High x

The Send Data Register SDRHx for channel x shows the data content of a data frame to be sent.

SDRHx (x=0-3)

Send Data Register High x **(098_H+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SD63	**SD62**	**SD61**	**SD60**	**SD59**	**SD58**	**SD57**	**SD56**	**SD55**	**SD54**	**SD53**	**SD52**	**SD51**	**SD50**	**SD49**	**SD48**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD47	**SD46**	**SD45**	**SD44**	**SD43**	**SD42**	**SD41**	**SD40**	**SD39**	**SD38**	**SD37**	**SD36**	**SD35**	**SD34**	**SD33**	**SD32**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SDy (y=32-63)	y-32	rw	**SDy** Send data of next ECU to Sensor frame. The unused MSBs (bit position SCRx.PLL and higher) must be written with '0'. If PLL is < 32, SDRHx must not be written.

Send Shift Register Low x

The Send Shift Register SSRLx for channel x shows the data that will be shifted to the Send Output Register SORL. It contains as well the stuff bits as specified in PSI5 V1.3 or in PSI5 V2.1 frame formats 1-3. During transmission, all bits are shifted to LSB with each sync pulse. MSB is filled up with zero.

SSRLx (x=0-3)

Send Shift Register Low x **($09C_H$+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SD31	**SD30**	**SD29**	**SD28**	**SD27**	**SD26**	**SD25**	**SD24**	**SD23**	**SD22**	**SD21**	**SD20**	**SD19**	**SD18**	**SD17**	**SD16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD15	**SD14**	**SD13**	**SD12**	**SD11**	**SD10**	**SD9**	**SD8**	**SD7**	**SD6**	**SD5**	**SD4**	**SD3**	**SD2**	**SD1**	**SD0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SDy (y=0-31)	y	rwh	**SDy** Send data of next ECU to Sensor frame. The sequence is: STS, AD, SD, CRC. Depending on individual field length, the bit stream is continued in SSRHx.

Send Shift Register High x

Continues the bit stream as described in SSRLx.

SSRHx (x=0-3)

Send Shift Register High x **($0A0_H$+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SD63	**SD62**	**SD61**	**SD60**	**SD59**	**SD58**	**SD57**	**SD56**	**SD55**	**SD54**	**SD53**	**SD52**	**SD51**	**SD50**	**SD49**	**SD48**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD47	**SD46**	**SD45**	**SD44**	**SD43**	**SD42**	**SD41**	**SD40**	**SD39**	**SD38**	**SD37**	**SD36**	**SD35**	**SD34**	**SD33**	**SD32**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SDy (y=32-63)	y-32	rwh	**SDy** Continues the bit stream as described in SSRLx.

Send Output Register Low x

The Send Output Register SORLx for channel x shows the data that will be shifted to the pulse generator. During transmission, all bits are shifted to LSB with each sync pulse. MSB is filled up from SORH bit 0, while the MSB of SORH is filled up with zeros.

SORLx (x=0-3)

Send Output Register Low x **($0A4_H$+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SD31	**SD30**	**SD29**	**SD28**	**SD27**	**SD26**	**SD25**	**SD24**	**SD23**	**SD22**	**SD21**	**SD20**	**SD19**	**SD18**	**SD17**	**SD16**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD15	**SD14**	**SD13**	**SD12**	**SD11**	**SD10**	**SD9**	**SD8**	**SD7**	**SD6**	**SD5**	**SD4**	**SD3**	**SD2**	**SD1**	**SD0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SDy (y=0-31)	y	rwh	**SDy** Send data of next ECU to Sensor frame. The sequence is: STS, AD, SD, CRC. Depending on individual field length, the bit stream is continued in SORHx.

Send Output Register High x

Continues the bit stream as described in SORLx.

SORHx (x=0-3)

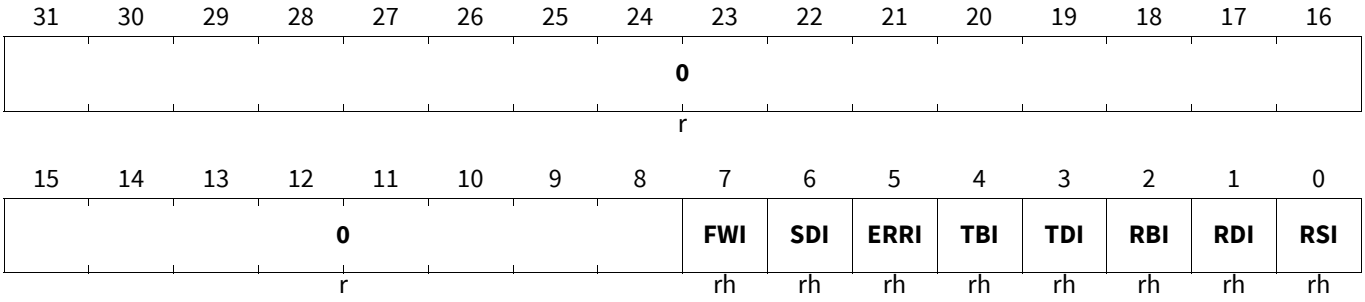
Send Output Register High x **($0A8_H$+x*90_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SD63	**SD62**	**SD61**	**SD60**	**SD59**	**SD58**	**SD57**	**SD56**	**SD55**	**SD54**	**SD53**	**SD52**	**SD51**	**SD50**	**SD49**	**SD48**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD47	**SD46**	**SD45**	**SD44**	**SD43**	**SD42**	**SD41**	**SD40**	**SD39**	**SD38**	**SD37**	**SD36**	**SD35**	**SD34**	**SD33**	**SD32**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SDy (y=32-63)	y-32	rwh	**SDy** Continues the bit stream as described in SORLx.

Interrupt Overview Register

INTOV
Interrupt Overview Register **($2F8_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RSI	0	rh	**Interrupt Pending on any Node Pointer RSI** If any interrupt requested flag is set for any Node Pointer RSI in register INTSTATx AND the referring interrupt is enabled in INTENx then INTOV.RSI is set. It is automatically reset if all flags in INTSTATx are cleared for which the referring interrupt is enabled in INTENx.
RDI	1	rh	**Interrupt Pending on any Node Pointer RDI** If any interrupt requested flag is set for any Node Pointer RDI in register INTSTATx AND the referring interrupt is enabled in INTENx then INTOV.RDI is set. It is automatically reset if all flags in INTSTATx are cleared for which the referring interrupt is enabled in INTENx.
RBI	2	rh	**Interrupt Pending on any Node Pointer RBI** If any interrupt requested flag is set for any Node Pointer RBI in register INTSTATx AND the referring interrupt is enabled in INTENx then INTOV.RBI is set. It is automatically reset if all flags in INTSTATx are cleared for which the referring interrupt is enabled in INTENx.
TDI	3	rh	**Interrupt Pending on any Node Pointer TDI** If any interrupt requested flag is set for any Node Pointer TDI in register INTSTATx AND the referring interrupt is enabled in INTENx then INTOV.TDI is set. It is automatically reset if all flags in INTSTATx are cleared for which the referring interrupt is enabled in INTENx.
TBI	4	rh	**Interrupt Pending on any Node Pointer TBI** If any interrupt requested flag is set for any Node Pointer TBI in register INTSTATx AND the referring interrupt is enabled in INTENx then INTOV.TBI is set. It is automatically reset if all flags in INTSTATx are cleared for which the referring interrupt is enabled in INTENx.
ERRI	5	rh	**Interrupt Pending on any Node Pointer ERRI** If any interrupt requested flag is set for any Node Pointer ERRI in register INTSTATx AND the referring interrupt is enabled in INTENx then INTOV.ERRI is set. It is automatically reset if all flags in INTSTATx are cleared for which the referring interrupt is enabled in INTENx.

Field	Bits	Type	Description
SDI	6	rh	**Interrupt Pending on any Node Pointer SDI** If any interrupt requested flag is set for any Node Pointer SDI in register INTSTATx AND the referring interrupt is enabled in INTENx then INTOV.SDI is set. It is automatically reset if all flags in INTSTATx are cleared for which the referring interrupt is enabled in INTENx.
FWI	7	rh	**Interrupt Pending on any Node Pointer FWI** If any interrupt requested flag is set for any Node Pointer FWI in register INTSTATx AND the referring interrupt is enabled in INTENx then INTOV.FWI is set. It is automatically reset if all flags in INTSTATx are cleared for which the referring interrupt is enabled in INTENx.
0	31:8	r	**Reserved** Read as 0.

Interrupt Node Pointer Register x

The Interrupt Node Pointer Register INPx contains the node pointers of PSI5 channel x.

Node Pointer ERRI is one single node pointer for the following interrupts:

- TEI
- NFI
- NBI
- MEI
- CRCI
- RUI
- RMI
- WSIy
- SOIy
- SCRIy

Node Pointer TBI is one single node pointer for the following interrupts:

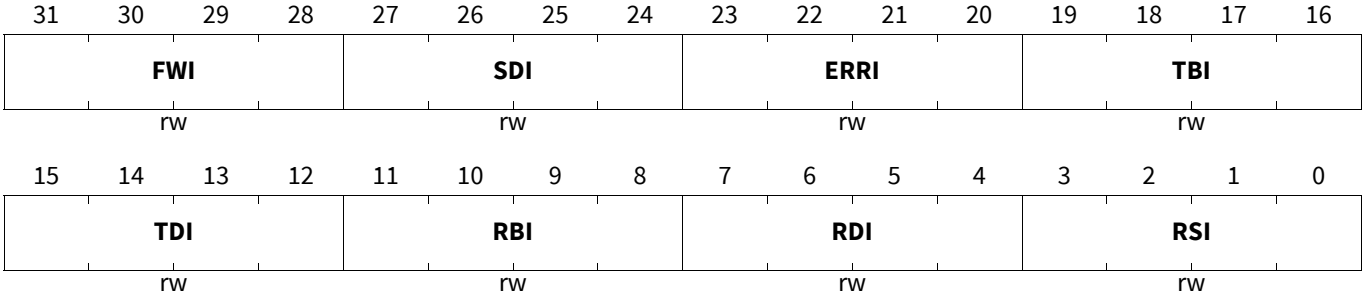
- TPOI
- TSOI
- TOOI

Node Pointer TDI is one single node pointer for the following interrupts:

- TPI
- TSI
- TOI

INPx (x=0-3)

Interrupt Node Pointer Register x **($2FC_H$+x*4)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RSI	3:0	rw	**Interrupt Node Pointer for Interrupt RSI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATAx.RSI (if enabled by bit INTENAx.RSI). 0_H Trigger Output TRIGO 0 is selected 1_H Trigger Output TRIGO 1 is selected … 7_H Trigger Output TRIGO 7 is selected 8_H reserved … F_H reserved
RDI	7:4	rw	**Interrupt Node Pointer for Interrupt RDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATAx.RDI (if enabled by bit INTENAx.RDI). For bit field definition, see RSI.
RBI	11:8	rw	**Interrupt Node Pointer for Interrupt RBI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATAx.RBI (if enabled by bit INTENAx.RBI). For bit field definition, see RSI.
TDI	15:12	rw	**Interrupt Node Pointer for Interrupt TDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATAx.TPI, TSI, TOI (if enabled by bit INTENAx.TPI, TSI, TOI). For bit field definition, see RSI.
TBI	19:16	rw	**Interrupt Node Pointer for Interrupt TBI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATAx.TPOI, TSOI, TOOI (if enabled by bit INTENAx.TPOI, TSOI, TOOI). For bit field definition, see RSI.
ERRI	23:20	rw	**Interrupt Node Pointer for Interrupt ERRI** This bit field defines the interrupt node, that is requested due to the set condition for bit TEI, NFI, NBI, MEI, CRCI, RUI, RMI, CRCIy, WSIy, SOIy, SCRIy. For bit field definition, see RSI.

Field	Bits	Type	Description
SDI	27:24	rw	**Interrupt Node Pointer for Interrupt SDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.SDI (if enabled by bit INTENx.SDI). For bit field definition, see RSI.
FWI	31:28	rw	**Interrupt Node Pointer for FWI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.FWI. For bit field definition, see RSI.

Interrupt Status Register A x

The Interrupt Status Registers INTSTATA/Bx contains status bits that show the status of any interrupt of PSI5 channel x.

The bits are set independently from the referring Interrupt Enable in Register INTENx. Thus they can be used as status bits as well e.g. by a SW based on polling.

INTSTATAx (x=0-3)
Interrupt Status Register A x **(310_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															NFI
r															rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOI	TOI	TSOI	TSI	TPOI	TPI	RMI	RUI	FWI	CRCI	MEI	NBI	TEI	RBI	RDI	RSI
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RSI	0	rh	**Receive Success Interrupt Request Flag** This bit is set at the successfully received end of a frame. It indicates that this frame is free of the errors NFI, TEI, NBI, MEI, CRCI if selected to be taken into account in register GCR. This bit can be cleared by bit INTCLRAx.RSI. This bit can be set by bit INTSETAx.RSI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
RDI	1	rh	**Receive Data Interrupt Request Flag** RDI is activated when a received frame is moved to a Receive Data Register RDRL/Hx. Both RDI and RSI will be issued together at correct reception. This bit can be cleared by bit INTCLRAx.RDI. This bit can be set by bit INTSETAx.RDI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

Field	Bits	Type	Description
RBI	2	rh	**Receive Buffer Overflow Interrupt Request Flag** This bit is set after a frame has been received while the old one was not read from RDRHx. I.e. the kernel wants to set any of the two interrupts RSI and RDI and finds any of these two interrupts already set. The old data is overwritten by the new data. Thus, RBI can be ignored if the receive memory is used. This bit is NOT cleared by reading RDRx. This bit can be cleared by bit INTCLRAx.RBI. This bit can be set by bit INTSETAx.RBI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
TEI	3	rh	**Time Slot Error Interrupt Request Flag** In Synchronous Mode (RCRAx.ASYN = 0), this bit is set if the SOF and the EOF of a frame are received in different time slots. The slots are constraint by the watch dog timer (see WDTxy). In Asynchronous Mode (RCRAx.ASYN = 1), this bit is set if the distance between two frames is longer than specified in WDL0. This bit can be cleared by bit INTCLRAx.TEI. This bit can be set by bit INTSETAx.TEI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
NBI	4	rh	**Number of Bits Wrong Request Flag** This bit is set if the last frame received less bits than expected but no Manchester coding error occurred until S1, S2, M0 and M1 where received. Note that the frame after the error might be completely wrong. This bit can be cleared by bit INTCLRAx.NBI. This bit can be set by bit INTSETAx.NBI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
MEI	5	rh	**Error in Message Bits Flag** This bit is set if a manchester error occurred in Bit M0 or M1. (Only if configured in RCRBx.MSG) Note that the frame after the error might be completely wrong. This bit can be cleared by bit INTCLRAx.MEI. This bit can be set by bit INTSETAx.MEI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

Field	Bits	Type	Description
CRCI	6	rh	**CRC Error Request Flag** This bit is set if the CRC fails. Set as well if CRC can not be calculated: PDL = 0, CRC cut off by Manchester Error, too few bits, BOT, Sync Pulse) This bit can be cleared by bit INTCLRAx.CRCI. This bit can be set by bit INTSETAx.CRCI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
FWI	7	rh	**FIFO Warning Level Request Flag** This bit is set after if the configured warning level of the FIFOx was reached. (See RFCx) This bit can be cleared by bit INTCLRAx.FWI. This bit can be set by bit INTSETAx.FWI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
RUI	8	rh	**Receive Memory Underrun Interrupt Request Flag** This bit is set after a SPB master reads from FIFO while no new data was available. (See RFCx) This bit can be cleared by bit INTCLRAx.RUI. This bit can be set by bit INTSETAx.RUI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
RMI	9	rh	**Receive Memory Overflow Flag** This bit is set after a frame has been received while RDIOVx.RDI[RFCx.WRP] is set. I.e. the kernel wants to set flag RDIxWRP and finds this flag already set. The old data is overwritten by the new data in the buffer memory. This bit can be cleared by bit INTCLRAx.RMIx. This bit can be set by bit INTSETAx.RMIx. This bit is set independently from INTENAx. 0_B No overflow 1_B Overflow
TPI	10	rh	**Transfer Preparation Interrupt Request Flag** This bit is set after data to be transferred has been moved from SDRL/H to SSRL/H. Thus a new value can be written to SDRL/Hx and the prepared data can be checked and modified by SW. This bit is automatically cleared by writing SDRx. This is an exception! It allows for DMA transfers without CPU activity. This bit can be cleared by bit INTCLRAx.TDI. This bit can be set by bit INTSETAx.TDI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

Field	Bits	Type	Description
TPOI	11	rh	**Transmit Preparation Overflow Interrupt Request Flag** This bit is set after if SDRL/H is written while TPF is set. The old data is NOT overwritten. This bit can be cleared by bit INTCLRAx.TPOI. This bit can be set by bit INTSETAx.TPOI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
TSI	12	rh	**Transfer Shift Interrupt Request Flag** This bit is set after data to be transferred has been moved from SSRL/H to SORL/H. Thus a new value can be written to SSRL/Hx. This bit is NOT cleared by writing SSRL/Hx. This bit can be cleared by bit INTCLRAx.TSI. This bit can be set by bit INTSETAx.TSI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
TSOI	13	rh	**Transmit Shift Overflow Interrupt Request Flag** This bit is set after if SSRL/H is written while TSF is set. The old data is NOT overwritten. This bit can be cleared by bit INTCLRAx.TSOI. This bit can be set by bit INTSETAx.TSOI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
TOI	14	rh	**Transfer Output Interrupt Request Flag** This bit is set after last bit to be transferred has been moved from SORL/H to the pulse generator. Thus a new value can be written to SORL/Hx. This bit is NOT cleared by writing SORL/Hx. This bit can be cleared by bit INTCLRAx.TOI. This bit can be set by bit INTSETAx.TOI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
TOOI	15	rh	**Transmit Shift Overflow Interrupt Request Flag** This bit is set after if SORL/H is written while TOF is set. The old data is NOT overwritten. This bit can be cleared by bit INTCLRAx.TOOI. This bit can be set by bit INTSETAx.TOOI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

Field	Bits	Type	Description
NFI	16	rh	**No Frame Received Interrupt Flag** This bit is set at the end of a slot if a frame is expected in the slot and SOF was not received in the slot. A special case is where an SOF is received in slot 0. In this scenario, the NFI bit is only set at the end of slot 1 if a frame is expected in slot 1 and an EOF was not received in slot 1. If only the two start bits are received and no further data, this “SOF” is discarded completely and treated like no frame received as well. If the “SOF only frame” crosses the WDL, NFI occurs after WDL and EOF detection. If RCRBx.VBS is set, an empty frame is stored anyhow. In this case RDIx is issued as well. The only valid information is SC, TS (captured at the end of the slot) and this bit (NFI). This bit can be cleared by bit INTCLRAx.NFI. This bit can be set by bit INTSETAx.NFI. This bit is set independently from INTENAx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
0	31:17	r	**Reserved** Read as 0.

Interrupt Status Register B x

INTSTATBx (x=0-3)

Interrupt Status Register B x **(324_H+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SCRI5	SCRI4	SCRI3	SCRI2	SCRI1	SCRI0	SOI5	SOI4
r								rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SOI3	SOI2	SOI1	SOI0	SDI5	SDI4	SDI3	SDI2	SDI1	SDI0	WSI5	WSI4	WSI3	WSI2	WSI1	WSI0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
WSIy (y=0-5)	y	rh	**Wrong Serial Protocol Error Request Flag** This bit is set if the Messaging bits are configured and do not show a start sequence for more than 18 frames of slot y. This bit can be cleared by bit INTCLRBx.WSIy. This bit can be set by bit INTSETBx.WSIy. This bit is set independently from INTENBx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

Field	Bits	Type	Description
SDIy (y=0-5)	y+6	rh	**Serial Data Receive Interrupt Request Flag** This bit is set after all serial data bits have been received via the Messaging bit field of slot y. This does NOT indicates a successful check of the CRC. This bit can be cleared by bit INTCLRBx.SDIy. This bit can be set by bit INTSETBx.SDIy. This bit is set independently from INTENBx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
SOIy (y=0-5)	y+12	rh	**Serial Data Buffer Overrun Interrupt Request Flag** This bit is set if the HW wants to set SDIy while SDIy is still set. This bit can be cleared by bit INTCLRBx.SOIy. This bit can be set by bit INTSETBx.SOIy. This bit is set independently from INTENBx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
SCRIy (y=0-5)	y+18	rh	**Serial Data CRC Error Request Flag** This bit is set if the CRC of the serial message fails. This includes a check of the Messaging bit field for correct 0 values of bit 1 in frames 7, 13 and 18. This bit can be cleared by bit INTCLRBx.SCRIy. This bit can be set by bit INTSETx.SCRIy. This bit is set independently from INTENBx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
0	31:24	r	**Reserved** Read as 0.

Interrupt Set Register A x

The Interrupt Set Register INTSETA/Bx contain control bits that trigger an interrupt pulse for any interrupt of PSI5 channel x.

INTSETAx (x=0-3)

Interrupt Set Register A x **(338_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							0								**NFI**
							r								w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOI	**TOI**	**TSOI**	**TSI**	**TPOI**	**TPI**	**RMI**	**RUI**	**FWI**	**CRCI**	**MEI**	**NBI**	**TEI**	**RBI**	**RDI**	**RSI**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RSI	0	w	**Set Interrupt Request Flag RSI** Setting this bit set bit INTSTATx.RSI. Clearing this bit has no effect. Reading this bit returns always zero.
RDI	1	w	**Set Interrupt Request Flag RDI** Setting this bit set bit INTSTATx.RDI. Clearing this bit has no effect. Reading this bit returns always zero.
RBI	2	w	**Set Interrupt Request Flag RBI** Setting this bit set bit INTSTATx.RBI. Clearing this bit has no effect. Reading this bit returns always zero.
TEI	3	w	**Set Interrupt Request Flag TEI** Setting this bit set bit INTSTATx.TEI. Clearing this bit has no effect. Reading this bit returns always zero.
NBI	4	w	**Set Interrupt Request Flag NBI** Setting this bit set bit INTSTATx.NBI. Clearing this bit has no effect. Reading this bit returns always zero.
MEI	5	w	**Set Interrupt Request Flag MEI** Setting this bit set bit INTSTATx.MEI. Clearing this bit has no effect. Reading this bit returns always zero.
CRCI	6	w	**Set Interrupt Request Flag CRCI** Setting this bit set bit INTSTATx.CRCI. Clearing this bit has no effect. Reading this bit returns always zero.
FWI	7	w	**Set Interrupt Request Flag FWI** Setting this bit set bit INTSTATx.FWI. Clearing this bit has no effect. Reading this bit returns always zero.
RUI	8	w	**Set Interrupt Request Flag RUI** Setting this bit set bit INTSTATx.RUI. Clearing this bit has no effect. Reading this bit returns always zero.
RMI	9	w	**Set Interrupt Request Flag RMI** Setting this bit set bit INTSTATx.RMI. Clearing this bit has no effect. Reading this bit returns always zero.
TPI	10	w	**Set Interrupt Request Flag TPI** Setting this bit set bit INTSTATx.TPI. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
TPOI	11	w	**Set Interrupt Request Flag TPOI** Setting this bit set bit INTSTATx.TPOI. Clearing this bit has no effect. Reading this bit returns always zero.
TSI	12	w	**Set Interrupt Request Flag TSI** Setting this bit set bit INTSTATx.TSI. Clearing this bit has no effect. Reading this bit returns always zero.
TSOI	13	w	**Set Interrupt Request Flag TSOI** Setting this bit set bit INTSTATx.TSOI. Clearing this bit has no effect. Reading this bit returns always zero.
TOI	14	w	**Set Interrupt Request Flag TOI** Setting this bit set bit INTSTATx.TOI. Clearing this bit has no effect. Reading this bit returns always zero.
TOOI	15	w	**Set Interrupt Request Flag TOOI** Setting this bit set bit INTSTATx.TOOI. Clearing this bit has no effect. Reading this bit returns always zero.
NFI	16	w	**Set Interrupt Request Flag NFI** Setting this bit set bit INTSTATx.NFI. Clearing this bit has no effect. Reading this bit returns always zero.
0	31:17	r	**Reserved** Read as 0; should be written with 0.

Interrupt Set Register B x

INTSETBx (x=0-3)

Interrupt Set Register B x **($34C_H$+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SCRI5	SCRI4	SCRI3	SCRI2	SCRI1	SCRI0	SOI5	SOI4
r								w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SOI3	SOI2	SOI1	SOI0	SDI5	SDI4	SDI3	SDI2	SDI1	SDI0	WSI5	WSI4	WSI3	WSI2	WSI1	WSI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
WSIy (y=0-5)	y	w	**Set Interrupt Request Flag WSIy** Setting this bit set bit INTSTATx.WSIy. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
SDIy (y=0-5)	y+6	w	**Set Interrupt Request Flag SDIy** Setting this bit set bit INTSTATx.SDIy. Clearing this bit has no effect. Reading this bit returns always zero.
SOIy (y=0-5)	y+12	w	**Set Interrupt Request Flag SOIy** Setting this bit set bit INTSTATx.SOIy. Clearing this bit has no effect. Reading this bit returns always zero.
SCRIy (y=0-5)	y+18	w	**Set Interrupt Request Flag SCRIy** Setting this bit set bit INTSTATx.SCRIy. Clearing this bit has no effect. Reading this bit returns always zero.
0	31:24	r	**Reserved** Read as 0; should be written with 0.

Interrupt Clear Register A x

The Interrupt Clear Register INTCLRA/Bx contain control bits that clear the status of any interrupt of PSI5 channel x.

INTCLRAx (x=0-3)
Interrupt Clear Register A x **(360_H+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															NFI
r															w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOI	TOI	TSOI	TSI	TPOI	TPI	RMI	RUI	FWI	CRCI	MEI	NBI	TEI	RBI	RDI	RSI
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RSI	0	w	**Clear Interrupt Request Flag RSI** Setting this bit clears bit INTSTATx.RSI. Clearing this bit has no effect. Reading this bit returns always zero.
RDI	1	w	**Clear Interrupt Request Flag RDI** Setting this bit clears bit INTSTATx.RDI. Clearing this bit has no effect. Reading this bit returns always zero.
RBI	2	w	**Clear Interrupt Request Flag RBI** Setting this bit clears bit INTSTATx.RBI. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
TEI	3	w	**Clear Interrupt Request Flag TEI** Setting this bit clears bit INTSTATx.TEI. Clearing this bit has no effect. Reading this bit returns always zero.
NBI	4	w	**Clear Interrupt Request Flag NBI** Setting this bit clears bit INTSTATx.NBI. Clearing this bit has no effect. Reading this bit returns always zero.
MEI	5	w	**Clear Interrupt Request Flag MEI** Setting this bit clears bit INTSTATx.MEI. Clearing this bit has no effect. Reading this bit returns always zero.
CRCI	6	w	**Clear Interrupt Request Flag CRCI** Setting this bit clears bit INTSTATx.CRCI. Clearing this bit has no effect. Reading this bit returns always zero.
FWI	7	w	**Clear Interrupt Request Flag FWI** Setting this bit clears bit INTSTATx.FWI. Clearing this bit has no effect. Reading this bit returns always zero.
RUI	8	w	**Clear Interrupt Request Flag RUI** Setting this bit clears bit INTSTATx.RUI. Clearing this bit has no effect. Reading this bit returns always zero.
RMI	9	w	**Clear Interrupt Request Flag RMI** Setting this bit clears bit INTSTATx.RMI. Clearing this bit has no effect. Reading this bit returns always zero.
TPI	10	w	**Clear Interrupt Request Flag TPI** Setting this bit clears bit INTSTATx.TPI. Clearing this bit has no effect. Reading this bit returns always zero.
TPOI	11	w	**Clear Interrupt Request Flag TPOI** Setting this bit clears bit INTSTATx.TPOI. Clearing this bit has no effect. Reading this bit returns always zero.
TSI	12	w	**Clear Interrupt Request Flag TSI** Setting this bit clears bit INTSTATx.TSI. Clearing this bit has no effect. Reading this bit returns always zero.
TSOI	13	w	**Clear Interrupt Request Flag TSOI** Setting this bit clears bit INTSTATx.TSOI. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
TOI	14	w	**Clear Interrupt Request Flag TOI** Setting this bit clears bit INTSTATx.TOI. Clearing this bit has no effect. Reading this bit returns always zero.
TOOI	15	w	**Clear Interrupt Request Flag TOOI** Setting this bit clears bit INTSTATx.TOOI. Clearing this bit has no effect. Reading this bit returns always zero.
NFI	16	w	**Clear Interrupt Request Flag NFI** Setting this bit clears bit INTSTATx.NFI. Clearing this bit has no effect. Reading this bit returns always zero.
0	31:17	r	**Reserved** Read as 0; should be written with 0.

Interrupt Clear Register A x

INTCLRBx (x=0-3)

Interrupt Clear Register A x **(374_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SCRI5	SCRI4	SCRI3	SCRI2	SCRI1	SCRI0	SOI5	SOI4
r								w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SOI3	SOI2	SOI1	SOI0	SDI5	SDI4	SDI3	SDI2	SDI1	SDI0	WSI5	WSI4	WSI3	WSI2	WSI1	WSI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
WSIy (y=0-5)	y	w	**Clear Interrupt Request Flag WSIy** Setting this bit clears bit INTSTATx.WSIy. Clearing this bit has no effect. Reading this bit returns always zero.
SDIy (y=0-5)	y+6	w	**Clear Interrupt Request Flag SDIy** Setting this bit clears bit INTSTATx.SDIy. Clearing this bit has no effect. Reading this bit returns always zero.
SOIy (y=0-5)	y+12	w	**Clear Interrupt Request Flag SOIy** Setting this bit clears bit INTSTATx.SOIy. Clearing this bit has no effect. Reading this bit returns always zero.
SCRIy (y=0-5)	y+18	w	**Clear Interrupt Request Flag SCRIy** Setting this bit clears bit INTSTATx.SCRIy. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
0	31:24	r	**Reserved** Read as 0; should be written with 0.

Interrupt Enable Register A x

The Interrupt Enable Register INTENA/B x contain control bits that enable the interrupt source of any interrupt of PSI5 channel x.

The Interrupt Status bits in register INTSTATA/Bx are set independently from the Interrupt Enable in Register INTENA/Bx.

INTENAx (x=0-3)

Interrupt Enable Register A x **(388_H+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							0								**NFI**
							r								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOI	**TOI**	**TSOI**	**TSI**	**TPOI**	**TPI**	**RMI**	**RUI**	**FWI**	**CRCI**	**MEI**	**NBI**	**TEI**	**RBI**	**RDI**	**RSI**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RSI	0	rw	**Enable Interrupt Request RSI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RDI	1	rw	**Enable Interrupt Request RDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RBI	2	rw	**Enable Interrupt Request RBI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TEI	3	rw	**Enable Interrupt Request TEI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
NBI	4	rw	**Enable Interrupt Request NBI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
MEI	5	rw	**Enable Interrupt Request MEII** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
CRCI	6	rw	**Enable Interrupt Request CRCI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source

Field	Bits	Type	Description
FWI	7	rw	**Enable Interrupt Request FWI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RUI	8	rw	**Enable Interrupt Request RUI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RMI	9	rw	**Enable Interrupt Request RMII** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TPI	10	rw	**Enable Interrupt Request TPI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TPOI	11	rw	**Enable Interrupt Request TPOI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TSI	12	rw	**Enable Interrupt Request TSI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TSOI	13	rw	**Enable Interrupt Request TSOI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TOI	14	rw	**Enable Interrupt Request TOI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TOOI	15	rw	**Enable Interrupt Request TOOI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
NFI	16	rw	**Enable Interrupt Request NFI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
0	31:17	r	**Reserved** Read as 0; should be written with 0.

Interrupt Enable Register B x

INTENBx (x=0-3)

Interrupt Enable Register B x **($39C_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SCRI5	SCRI4	SCRI3	SCRI2	SCRI1	SCRI0	SOI5	SOI4
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SOI3	SOI2	SOI1	SOI0	SDI5	SDI4	SDI3	SDI2	SDI1	SDI0	WSI5	WSI4	WSI3	WSI2	WSI1	WSI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
WSIy (y=0-5)	y	rw	**Enable Interrupt Request WSIy** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
SDIy (y=0-5)	y+6	rw	**Enable Interrupt Request SDIy** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
SOIy (y=0-5)	y+12	rw	**Enable Interrupt Request SOIy** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
SCRIy (y=0-5)	y+18	rw	**Enable Interrupt Request SCRIy** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
0	31:24	r	**Reserved** Read as 0; should be written with 0.

OCDS Control and Status

The OCDS Control and Status (OCS) register is cleared by Debug Reset. The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32 bit wide only and requires Supervisor Mode.

OCS
OCDS Control and Status **($3CC_H$)** **Debug Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		SUSSTA	SUS_P	SUS				0							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. The kernel continues when hard suspend is left. 2_H Soft suspend option A (Suspends after end of current send or receive transfers, if any are ongoing) **others**, reserved
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.

Field	Bits	Type	Description
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:0, 31:30	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access[1] for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B…EN31 -> TAG ID 011111B.

ACCEN0

Access Enable Register 0 ($3D0_H$) Application Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

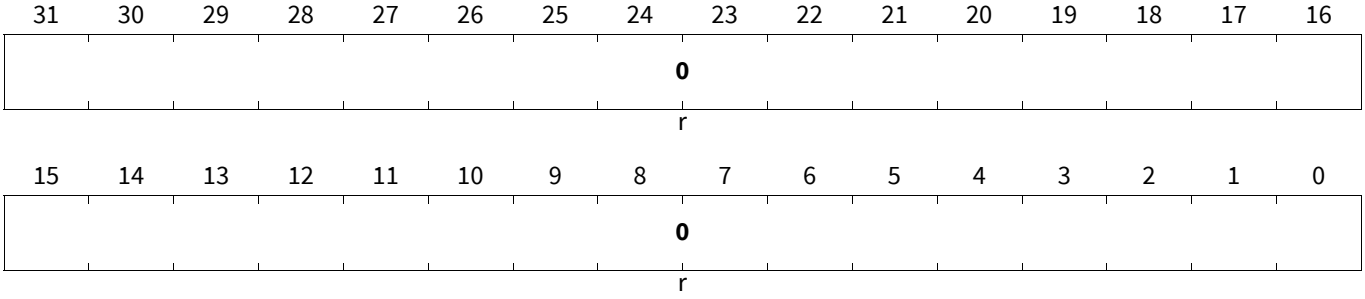
The Access Enable Register 1 controls write access[1] for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B…EN31 -> TAG ID 111111B.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

ACCEN1

Access Enable Register 1 **(3D4$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0

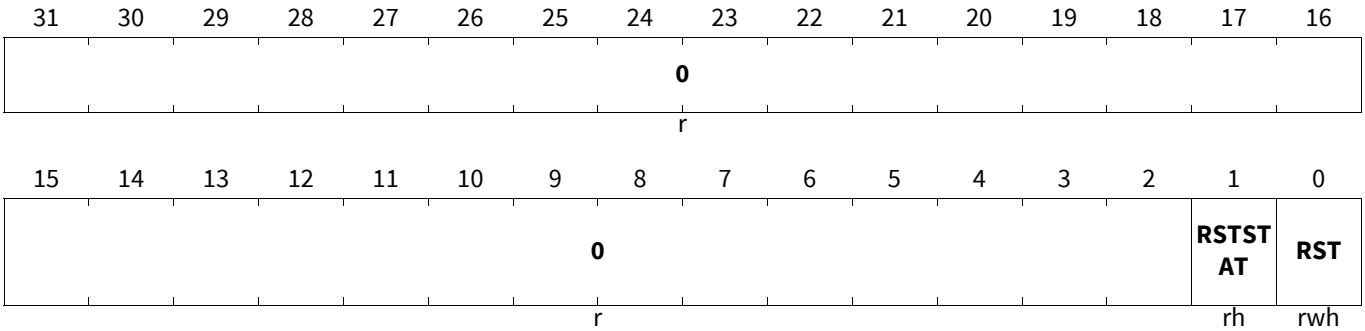
The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0

Kernel Reset Register 0 **(3D8$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested

Field	Bits	Type	Description
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **($3DC_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**RST**
r															rwh

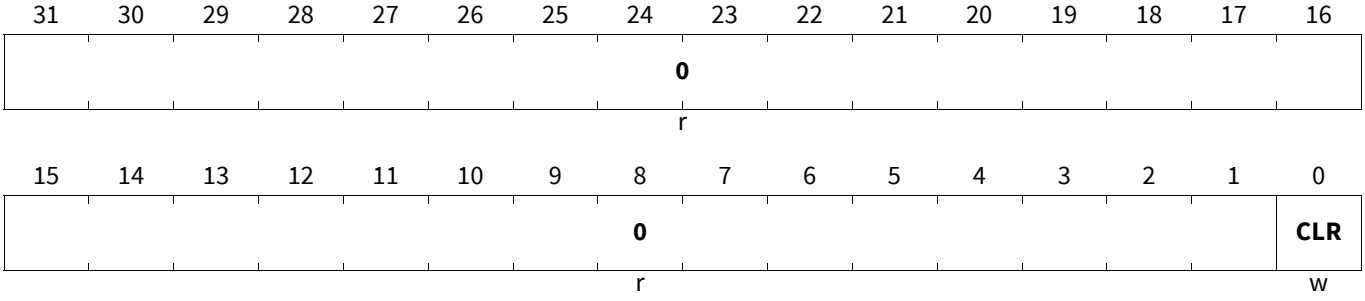
Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register (3E0$_H$) Application Reset Value: 0000 0000$_H$



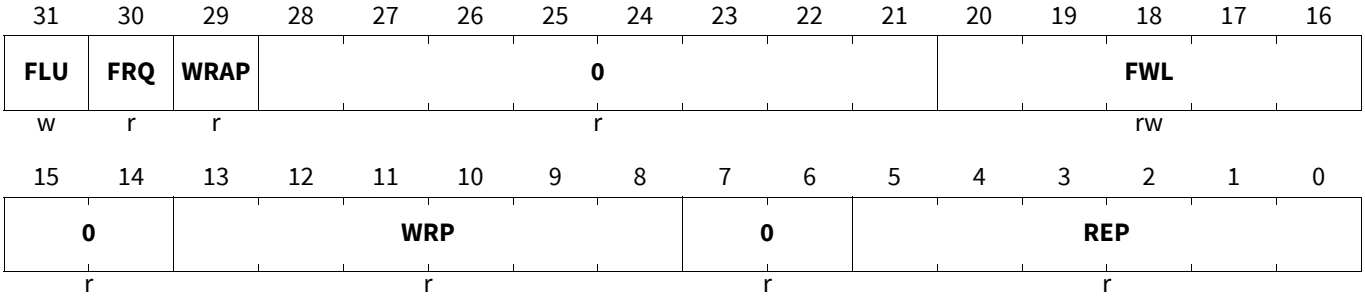
Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Receive FIFO Control Register x

RFCx contains control bits for the FIFO and Ring Buffer operation.

RFCx (x=0-3)

Receive FIFO Control Register x (3E4$_H$+x*4) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
REP	5:0	r	**FIFO Read Pointer** points to the buffer to be read next. Incremented after read. The last bit bit indicates if RDMxH or RDMxL is read. This LSB is ignored for FWL comparison WRP - REP collision (RBI, RUI)
WRP	13:8	r	**FIFO/Ring Buffer Write Pointer** points to the buffer written last. Incremented before write. The last bit indicates if RDMxH or RDMxL is written. This LSB is ignored for FWL comparison WRP - REP collision (RBI, RUI).

Field	Bits	Type	Description
FWL	20:16	rw	**FIFO Warning Level** Number of new entries at which the FIFO Warning Interrupt FWIx is set automatically. For calculation WRP and REP are used. I.e. only if the memory is read via RDFx, this works properly.
WRAP	29	r	**Write Pointer WRAP Indicator** If set, the Write Pointer is one wrap around ahead of the Read Pointer. It is cleared, if the Read Pointer wraps around too or when FLU is set by SW.
FRQ	30	r	**Flush Request** if set, FIFO read and write pointers are not incremented any more due to excessive RMI. (Too many overrun conditions). This bit is cleared when FLU is set.
FLU	31	w	**Flush** if set, FIFO read and write pointers are cleared. Bits and WRAP are cleared. SW needs to clear interrupts as needed. This bit can only be set and is cleared by HW. Always read as 0.
0	7:6, 15:14, 28:21	r	**Reserved** Read as 0; should be written with 0.

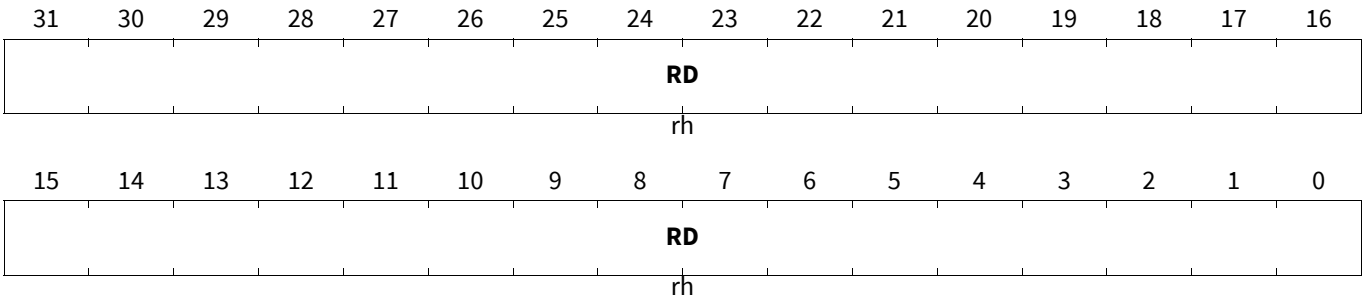
Receive Data FIFO x

The Receive Data FIFO RDFx for channel x shows the oldest data content of up to 32 received data frames. I.e. RFCx.REP is pointing at the buffer presented.

Do not set a debugger watch point on this register as any read access will issue a hard ware request for new data (move read pointer REP and present referring memory content in RDFx). A debugger watch point can be set on the RDML/Hx and RFCx to monitor the FIFO behavior.

RDFx (x=0-3)

Receive Data FIFO x **($3F8_H$+x*4)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RD	31:0	rh	**RD** Shows the content of RDML/Hxy; y = RFCx.REP[5:1]; L/H = RFCx.REP[0]. Reading this register triggers incrementation of REP and presentation of next FIFO value at next read access! Once a complete buffer is read (64 bit, 2 accesses to RDF) all overview flags for this very buffer are cleared automatically.

RSI Overview Register x

The RSI Overview Register RSIOVx contains flags to manage the receive memory of PSI5 channel x.

RSIOVx (x=0-3)

RSI Overview Register x **($40C_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSI31	**RSI30**	**RSI29**	**RSI28**	**RSI27**	**RSI26**	**RSI25**	**RSI24**	**RSI23**	**RSI22**	**RSI21**	**RSI20**	**RSI19**	**RSI18**	**RSI17**	**RSI16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSI15	**RSI14**	**RSI13**	**RSI12**	**RSI11**	**RSI10**	**RSI9**	**RSI8**	**RSI7**	**RSI6**	**RSI5**	**RSI4**	**RSI3**	**RSI2**	**RSI1**	**RSI0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RSIy (y=0-31)	y	rh	**RSI Flag of Buffer y** Copied from INTSTATAx at frame end while RFCx.WRP pointed to buffer y. This bit can be cleared by bit RSICLRx.RSIy. This bit can be set by bit RSISETx.RSIy.

RMI Overview Register x

The RMI Overview Register RMIOVx contains flags to manage the receive memory of PSI5 channel x.

RMIOVx (x=0-3)

RMI Overview Register x **(420_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMI31	**RMI30**	**RMI29**	**RMI28**	**RMI27**	**RMI26**	**RMI25**	**RMI24**	**RMI23**	**RMI22**	**RMI21**	**RMI20**	**RMI19**	**RMI18**	**RMI17**	**RMI16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMI15	**RMI14**	**RMI13**	**RMI12**	**RMI11**	**RMI10**	**RMI9**	**RMI8**	**RMI7**	**RMI6**	**RMI5**	**RMI4**	**RMI3**	**RMI2**	**RMI1**	**RMI0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RMIy (y=0-31)	y	rh	**RMI Flag of Buffer y** Copied from INTSTATAx at frame end while RFCx.WRP pointed to buffer y. This bit can be cleared by bit RMICLRx.RMIy. This bit can be set by bit RMISETx.RMIy.

NBI Overview Register x

The NBI Overview Register NBIOVx contains flags to manage the receive memory of PSI5 channel x.

NBIOVx (x=0-3)
NBI Overview Register x **(434_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NBI31	**NBI30**	**NBI29**	**NBI28**	**NBI27**	**NBI26**	**NBI25**	**NBI24**	**NBI23**	**NBI22**	**NBI21**	**NBI20**	**NBI19**	**NBI18**	**NBI17**	**NBI16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NBI15	**NBI14**	**NBI13**	**NBI12**	**NBI11**	**NBI10**	**NBI9**	**NBI8**	**NBI7**	**NBI6**	**NBI5**	**NBI4**	**NBI3**	**NBI2**	**NBI1**	**NBI0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NBIy (y=0-31)	y	rh	**NBI Flag of Buffer y** Copied from INTSTATAx at frame end while RFCx.WRP pointed to buffer y. This bit can be cleared by bit NBICLRx.NBIy. This bit can be set by bit NBISETx.NBIy.

TEI Overview Register x

The TEI Overview Register TEIOVx contains flags to manage the receive memory of PSI5 channel x.

TEIOVx (x=0-3)
TEI Overview Register x **(448_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEI31	**TEI30**	**TEI29**	**TEI28**	**TEI27**	**TEI26**	**TEI25**	**TEI24**	**TEI23**	**TEI22**	**TEI21**	**TEI20**	**TEI19**	**TEI18**	**TEI17**	**TEI16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEI15	**TEI14**	**TEI13**	**TEI12**	**TEI11**	**TEI10**	**TEI9**	**TEI8**	**TEI7**	**TEI6**	**TEI5**	**TEI4**	**TEI3**	**TEI2**	**TEI1**	**TEI0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TEIy (y=0-31)	y	rh	**TEI Flag of Buffer y** Copied from INTSTATAx at frame end while RFCx.WRP pointed to buffer y. This bit can be cleared by bit TEICLRx.TEIy. This bit can be set by bit TEISETx.TEIy.

CRCI Overview Register x

The CRCI Overview Register CRCIOVx contains flags to manage the receive memory of PSI5 channel x.

CRCIOVx (x=0-3)

CRCI Overview Register x **($45C_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCI31	CRCI30	CRCI29	CRCI28	CRCI27	CRCI26	CRCI25	CRCI24	CRCI23	CRCI22	CRCI21	CRCI20	CRCI19	CRCI18	CRCI17	CRCI16
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCI15	CRCI14	CRCI13	CRCI12	CRCI11	CRCI10	CRCI9	CRCI8	CRCI7	CRCI6	CRCI5	CRCI4	CRCI3	CRCI2	CRCI1	CRCI0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
CRCIy (y=0-31)	y	rh	**CRCI Flag of Buffer y** Copied from INTSTATAx at frame end while RFCx.WRP pointed to buffer y. This bit can be cleared by bit CRCICLRx.CRCIy. This bit can be set by bit CRCISETx.CRCIy.

RDI Overview Register x

The RDI Overview Register RDIOVx contains flags to manage the receive memory of PSI5 channel x.

RDIOVx (x=0-3)

RDI Overview Register x **(470_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDI31	RDI30	RDI29	RDI28	RDI27	RDI26	RDI25	RDI24	RDI23	RDI22	RDI21	RDI20	RDI19	RDI18	RDI17	RDI16
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDI15	RDI14	RDI13	RDI12	RDI11	RDI10	RDI9	RDI8	RDI7	RDI6	RDI5	RDI4	RDI3	RDI2	RDI1	RDI0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RDIy (y=0-31)	y	rh	**RDI Flag of Buffer y** Copied from INTSTATAx at frame end while RFCx.WRP pointed to buffer y. This bit can be cleared by bit RDICLRx.RDIy. This bit can be set by bit RDISETx.RDIy.

NFI Overview Register x

The NFI Overview Register NFIOVx contains flags to manage the receive memory of PSI5 channel x.

NFIOVx (x=0-3)

NFI Overview Register x **(484_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NFI31	**NFI30**	**NFI29**	**NFI28**	**NFI27**	**NFI26**	**NFI25**	**NFI24**	**NFI23**	**NFI22**	**NFI21**	**NFI20**	**NFI19**	**NFI18**	**NFI17**	**NFI16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NFI15	**NFI14**	**NFI13**	**NFI12**	**NFI11**	**NFI10**	**NFI9**	**NFI8**	**NFI7**	**NFI6**	**NFI5**	**NFI4**	**NFI3**	**NFI2**	**NFI1**	**NFI0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
NFIy (y=0-31)	y	rh	**NFI Flag of Buffer y** Copied from INTSTATAx at frame end while RFCx.WRP pointed to buffer y. This bit can be cleared by bit NFICLRx.NFIy. This bit can be set by bit NFISETx.NFIy.

MEI Overview Register x

The MEI Overview Register MEIOVx contains flags to manage the receive memory of PSI5 channel x.

MEIOVx (x=0-3)

MEI Overview Register x **(498_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEI31	**MEI30**	**MEI29**	**MEI28**	**MEI27**	**MEI26**	**MEI25**	**MEI24**	**MEI23**	**MEI22**	**MEI21**	**MEI20**	**MEI19**	**MEI18**	**MEI17**	**MEI16**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEI15	**MEI14**	**MEI13**	**MEI12**	**MEI11**	**MEI10**	**MEI9**	**MEI8**	**MEI7**	**MEI6**	**MEI5**	**MEI4**	**MEI3**	**MEI2**	**MEI1**	**MEI0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
MEIy (y=0-31)	y	rh	**MEI Flag of Buffer y** Copied from INTSTATAx at frame end while RFCx.WRP pointed to buffer y. This bit can be cleared by bit MEICLRx.MEIy. This bit can be set by bit MEISETx.MEIy.

RSI Overview Set Register x

The RSI Overview Set Register RSISETx contains bits to set the status flags of RSIOVx.

RSISETx (x=0-3)

RSI Overview Set Register x **($4AC_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSI31	RSI30	RSI29	RSI28	RSI27	RSI26	RSI25	RSI24	RSI23	RSI22	RSI21	RSI20	RSI19	RSI18	RSI17	RSI16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSI15	RSI14	RSI13	RSI12	RSI11	RSI10	RSI9	RSI8	RSI7	RSI6	RSI5	RSI4	RSI3	RSI2	RSI1	RSI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RSIy (y=0-31)	y	w	**Set RSI Flag of Buffer y** Setting this bit sets bit RSIOVx.RSIy Clearing this bit has no effect. Reading this bit returns always zero.

RMI Overview Set Register x

The RMI Overview Set Register RMISETx contains bits to set the status flags of RMIOVx.

RMISETx (x=0-3)

RMI Overview Set Register x **($4C0_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMI31	RMI30	RMI29	RMI28	RMI27	RMI26	RMI25	RMI24	RMI23	RMI22	RMI21	RMI20	RMI19	RMI18	RMI17	RMI16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMI15	RMI14	RMI13	RMI12	RMI11	RMI10	RMI9	RMI8	RMI7	RMI6	RMI5	RMI4	RMI3	RMI2	RMI1	RMI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RMIy (y=0-31)	y	w	**Set RMI Flag of Buffer y** Setting this bit sets bit RMIOVx.RMIy Clearing this bit has no effect. Reading this bit returns always zero.

NBI Overview Set Register x

The NBI Overview Set Register NBISETx contains bits to set the status flags of NBIOVx.

NBISETx (x=0-3)

NBI Overview Set Register x **($4D4_H$+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NBI31	**NBI30**	**NBI29**	**NBI28**	**NBI27**	**NBI26**	**NBI25**	**NBI24**	**NBI23**	**NBI22**	**NBI21**	**NBI20**	**NBI19**	**NBI18**	**NBI17**	**NBI16**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NBI15	**NBI14**	**NBI13**	**NBI12**	**NBI11**	**NBI10**	**NBI9**	**NBI8**	**NBI7**	**NBI6**	**NBI5**	**NBI4**	**NBI3**	**NBI2**	**NBI1**	**NBI0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
NBIy (y=0-31)	y	w	**Set NBI Flag of Buffer y** Setting this bit sets bit NBIOVx.NBIy Clearing this bit has no effect. Reading this bit returns always zero.

TEI Overview Set Register x

The TEI Overview Set Register TEISETx contains bits to set the status flags of TEIOVx.

TEISETx (x=0-3)

TEI Overview Set Register x **($4E8_H$+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEI31	**TEI30**	**TEI29**	**TEI28**	**TEI27**	**TEI26**	**TEI25**	**TEI24**	**TEI23**	**TEI22**	**TEI21**	**TEI20**	**TEI19**	**TEI18**	**TEI17**	**TEI16**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEI15	**TEI14**	**TEI13**	**TEI12**	**TEI11**	**TEI10**	**TEI9**	**TEI8**	**TEI7**	**TEI6**	**TEI5**	**TEI4**	**TEI3**	**TEI2**	**TEI1**	**TEI0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
TEIy (y=0-31)	y	w	**Set TEI Flag of Buffer y** Setting this bit sets bit TEIOVx.TEIy Clearing this bit has no effect. Reading this bit returns always zero.

CRCI Overview Set Register x

The CRCI Overview Set Register CRCISETx contains bits to set the status flags of CRCIOVx.

CRCISETx (x=0-3)

CRCI Overview Set Register x **($4FC_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCI31	**CRCI30**	**CRCI29**	**CRCI28**	**CRCI27**	**CRCI26**	**CRCI25**	**CRCI24**	**CRCI23**	**CRCI22**	**CRCI21**	**CRCI20**	**CRCI19**	**CRCI18**	**CRCI17**	**CRCI16**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCI15	**CRCI14**	**CRCI13**	**CRCI12**	**CRCI11**	**CRCI10**	**CRCI9**	**CRCI8**	**CRCI7**	**CRCI6**	**CRCI5**	**CRCI4**	**CRCI3**	**CRCI2**	**CRCI1**	**CRCI0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CRCIy (y=0-31)	y	w	**Set CRCI Flag of Buffer y** Setting this bit sets bit CRCIOVx.CRCIy Clearing this bit has no effect. Reading this bit returns always zero.

RDI Overview Set Register x

The RDI Overview Set Register RDISETx contains bits to set the status flags of RDIOVx.

RDISETx (x=0-3)

RDI Overview Set Register x **(510_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDI31	**RDI30**	**RDI29**	**RDI28**	**RDI27**	**RDI26**	**RDI25**	**RDI24**	**RDI23**	**RDI22**	**RDI21**	**RDI20**	**RDI19**	**RDI18**	**RDI17**	**RDI16**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDI15	**RDI14**	**RDI13**	**RDI12**	**RDI11**	**RDI10**	**RDI9**	**RDI8**	**RDI7**	**RDI6**	**RDI5**	**RDI4**	**RDI3**	**RDI2**	**RDI1**	**RDI0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RDIy (y=0-31)	y	w	**Set RDI Flag of Buffer y** Setting this bit sets bit RDIOVx.RDIy Clearing this bit has no effect. Reading this bit returns always zero.

NFI Overview Set Register x

The NFI Overview Set Register NFISETx contains bits to set the status flags of NFIOVx.

NFISETx (x=0-3)

NFI Overview Set Register x **(524_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NFI31	**NFI30**	**NFI29**	**NFI28**	**NFI27**	**NFI26**	**NFI25**	**NFI24**	**NFI23**	**NFI22**	**NFI21**	**NFI20**	**NFI19**	**NFI18**	**NFI17**	**NFI16**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NFI15	**NFI14**	**NFI13**	**NFI12**	**NFI11**	**NFI10**	**NFI9**	**NFI8**	**NFI7**	**NFI6**	**NFI5**	**NFI4**	**NFI3**	**NFI2**	**NFI1**	**NFI0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
NFIy (y=0-31)	y	w	**Set NFI Flag of Buffer y** Setting this bit sets bit NFIOVx.NFIy Clearing this bit has no effect. Reading this bit returns always zero.

MEI Overview Set Register x

The MEI Overview Set Register MEISETx contains bits to set the status flags of MEIOVx.

MEISETx (x=0-3)

MEI Overview Set Register x **(538_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEI31	**MEI30**	**MEI29**	**MEI28**	**MEI27**	**MEI26**	**MEI25**	**MEI24**	**MEI23**	**MEI22**	**MEI21**	**MEI20**	**MEI19**	**MEI18**	**MEI17**	**MEI16**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEI15	**MEI14**	**MEI13**	**MEI12**	**MEI11**	**MEI10**	**MEI9**	**MEI8**	**MEI7**	**MEI6**	**MEI5**	**MEI4**	**MEI3**	**MEI2**	**MEI1**	**MEI0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
MEIy (y=0-31)	y	w	**Set MEI Flag of Buffer y** Setting this bit sets bit MEIOVx.MEIy Clearing this bit has no effect. Reading this bit returns always zero.

RSI Overview Clear Register x

The RSI Overview Clear Register RSICLRx contains bits to Clear the status flags of RSIOVx.

RSICLRx (x=0-3)

RSI Overview Clear Register x **($54C_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSI31	RSI30	RSI29	RSI28	RSI27	RSI26	RSI25	RSI24	RSI23	RSI22	RSI21	RSI20	RSI19	RSI18	RSI17	RSI16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSI15	RSI14	RSI13	RSI12	RSI11	RSI10	RSI9	RSI8	RSI7	RSI6	RSI5	RSI4	RSI3	RSI2	RSI1	RSI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RSIy (y=0-31)	y	w	**Clear RSI Flag of Buffer y** Setting this bit clears bit RSIOVx.RSIy Clearing this bit has no effect. Reading this bit returns always zero.

RMI Overview Clear Register x

The RMI Overview Clear Register RMICLRx contains bits to Clear the status flags of RMIOVx.

RMICLRx (x=0-3)

RMI Overview Clear Register x **(560_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMI31	RMI30	RMI29	RMI28	RMI27	RMI26	RMI25	RMI24	RMI23	RMI22	RMI21	RMI20	RMI19	RMI18	RMI17	RMI16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMI15	RMI14	RMI13	RMI12	RMI11	RMI10	RMI9	RMI8	RMI7	RMI6	RMI5	RMI4	RMI3	RMI2	RMI1	RMI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RMIy (y=0-31)	y	w	**Clear RMI Flag of Buffer y** Setting this bit clears bit RMIOVx.RMIy Clearing this bit has no effect. Reading this bit returns always zero.

NBI Overview Clear Register x

The NBI Overview Clear Register NBICLRx contains bits to Clear the status flags of NBIOVx.

NBICLRx (x=0-3)

NBI Overview Clear Register x **(574_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NBI31	NBI30	NBI29	NBI28	NBI27	NBI26	NBI25	NBI24	NBI23	NBI22	NBI21	NBI20	NBI19	NBI18	NBI17	NBI16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NBI15	NBI14	NBI13	NBI12	NBI11	NBI10	NBI9	NBI8	NBI7	NBI6	NBI5	NBI4	NBI3	NBI2	NBI1	NBI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
NBIy (y=0-31)	y	w	**Clear NBI Flag of Buffer y** Setting this bit clears bit NBIOVx.NBIy Clearing this bit has no effect. Reading this bit returns always zero.

TEI Overview Clear Register x

The TEI Overview Clear Register TEICLRx contains bits to clear the status flags of TEIOVx.

TEICLRx (x=0-3)

TEI Overview Clear Register x **(588_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEI31	TEI30	TEI29	TEI28	TEI27	TEI26	TEI25	TEI24	TEI23	TEI22	TEI21	TEI20	TEI19	TEI18	TEI17	TEI16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEI15	TEI14	TEI13	TEI12	TEI11	TEI10	TEI9	TEI8	TEI7	TEI6	TEI5	TEI4	TEI3	TEI2	TEI1	TEI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
TEIy (y=0-31)	y	w	**Clear TEI Flag of Buffer y** Setting this bit clears bit TEIOVx.TEIy Clearing this bit has no effect. Reading this bit returns always zero.

CRCI Overview Clear Register x

The CRCI Overview Clear Register CRCICLRx contains bits to Clear the status flags of CRCIOVx.

CRCICLRx (x=0-3)

CRCI Overview Clear Register x **($59C_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCI31	**CRCI30**	**CRCI29**	**CRCI28**	**CRCI27**	**CRCI26**	**CRCI25**	**CRCI24**	**CRCI23**	**CRCI22**	**CRCI21**	**CRCI20**	**CRCI19**	**CRCI18**	**CRCI17**	**CRCI16**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCI15	**CRCI14**	**CRCI13**	**CRCI12**	**CRCI11**	**CRCI10**	**CRCI9**	**CRCI8**	**CRCI7**	**CRCI6**	**CRCI5**	**CRCI4**	**CRCI3**	**CRCI2**	**CRCI1**	**CRCI0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
CRCIy (y=0-31)	y	w	**Clear CRCI Flag of Buffer y** Setting this bit clears bit CRCIOVx.CRCIy Clearing this bit has no effect. Reading this bit returns always zero.

RDI Overview Clear Register x

The RDI Overview Clear Register RDICLRx contains bits to clear the status flags of RDIOVx.

RDICLRx (x=0-3)

RDI Overview Clear Register x **($5B0_H$+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDI31	**RDI30**	**RDI29**	**RDI28**	**RDI27**	**RDI26**	**RDI25**	**RDI24**	**RDI23**	**RDI22**	**RDI21**	**RDI20**	**RDI19**	**RDI18**	**RDI17**	**RDI16**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDI15	**RDI14**	**RDI13**	**RDI12**	**RDI11**	**RDI10**	**RDI9**	**RDI8**	**RDI7**	**RDI6**	**RDI5**	**RDI4**	**RDI3**	**RDI2**	**RDI1**	**RDI0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RDIy (y=0-31)	y	w	**Clear RDI Flag of Buffer y** Setting this bit clears bit RDIOVx.RDIy Clearing this bit has no effect. Reading this bit returns always zero.

NFI Overview Clear Register x

The NFI Overview Clear Register NFICLRx contains bits to clear the status flags of NFIOVx.

NFICLRx (x=0-3)

NFI Overview Clear Register x **(5C4$_H$+x*4)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NFI31	NFI30	NFI29	NFI28	NFI27	NFI26	NFI25	NFI24	NFI23	NFI22	NFI21	NFI20	NFI19	NFI18	NFI17	NFI16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NFI15	NFI14	NFI13	NFI12	NFI11	NFI10	NFI9	NFI8	NFI7	NFI6	NFI5	NFI4	NFI3	NFI2	NFI1	NFI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
NFIy (y=0-31)	y	w	**Clear NFI Flag of Buffer y** Setting this bit clears bit NFIOVx.NFIy Clearing this bit has no effect. Reading this bit returns always zero.

MEI Overview Clear Register x

The MEI Overview Clear Register MEICLRx contains bits to clear the status flags of MEIOVx.

MEICLRx (x=0-3)

MEI Overview Clear Register x **(5D8$_H$+x*4)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEI31	MEI30	MEI29	MEI28	MEI27	MEI26	MEI25	MEI24	MEI23	MEI22	MEI21	MEI20	MEI19	MEI18	MEI17	MEI16
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MEI15	MEI14	MEI13	MEI12	MEI11	MEI10	MEI9	MEI8	MEI7	MEI6	MEI5	MEI4	MEI3	MEI2	MEI1	MEI0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
MEIy (y=0-31)	y	w	**Clear MEI Flag of Buffer y** Setting this bit clears bit MEIOVx.MEIy Clearing this bit has no effect. Reading this bit returns always zero.

Receive Data Memory Low xy

The Receive Data Memory RDMLx for channel x shows the data content of up to 32 received data frames.

The internal receive buffer is always cleared (0x0000 0000$_H$) at each frame start. Thus unused nibbles are always read as zero.

RDMLxy (x=0-3;y=0-31)

Receive Data Memory Low xy **(600_H+x*100_H+y*8)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RD															
rh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD												CRC			CRCI
rh												rh			rh

Field	Bits	Type	Description
CRCI	0	rh	**CRC Error Flag** Copied from RDRLx at frame end.
CRC	3:1	rh	**CRC** Copied from RDRLx at frame end.
RD	31:4	rh	**RD** Copied from RDRLx at frame end.

Receive Data Memory High xy

The Receive Data Memory RDMHx for channel x shows the data content of up to 32 received data frames.

The internal receive buffer is always cleared (0x0000 0000_H) at each frame start. Thus unused nibbles are always read as zero.

RDMHxy (x=0-3;y=0-31)

Receive Data Memory High xy **(604_H+x*100_H+y*8)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMI	NFI	MEI	NBI	TEI	SC			TS							
rh	rh	rh	rh	rh	rh			rh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS															
rh															

Field	Bits	Type	Description
TS	23:0	rh	**Time Stamp** Copied from RDRHx at frame end.
SC	26:24	rh	**Slot Counter** Copied from RDRHx at frame end.
TEI	27	rh	**Time Slot Error Flag** Copied from RDRHx at frame end.
NBI	28	rh	**Number of bits Error Flag** Copied from RDRHx at frame end.

Field	Bits	Type	Description
MEI	29	rh	**Error in Messaging Bits Flag** Copied from RDRHx at frame end.
NFI	30	rh	**No Frame Received Flag** Copied from RDRHx at frame end.
RMI	31	rh	**Receive Memory Overflow Flag** Copied from INTSTATAx at frame end.

42.5 Safety Measures

Description of safety mechanisms and conditions of use.

42.6 IO Interfaces

The table below lists all the interfaces of the PSI5 to other modules.

Table 428 List of PSI5 Interface Signals

Interface Signals	I/O	Description
sx_fpi		**FPI slave interface**
sx_irq_psi5		**Interrupt requests**
sx_gtm2psi5		
TRIGO(7:0)	out	**PSI5 Service Request**
TRIG(5:0)	in	**GTM timer output vector - synchronized**
TX(3:0)	out	**TXD outputs (send data)** This is the TX (send) signal
RX0A	in	**RXD inputs (receive data) channel 0** This is the RX (receive) signal.
RX0B		
RX0C		
RX0D		
RX1A	in	**RXD inputs (receive data) channel 1** This is the RX (receive) signal.
RX1B		
RX1C		
RX1D		
RX2A	in	**RXD inputs (receive data) channel 2** This is the RX (receive) signal.
RX2B		
RX2C		
RX2D		
RX3A	in	**RXD inputs (receive data) channel 3** This is the RX (receive) signal.
RX3B		
RX3C		
RX3D		

42.7 Revision History

Table 429 Revision History

Reference	Change to Previous Version	Comment
V1.17.11		
	Revision History entries up to V1.17.10 removed.	
V1.17.12		
–	No functional change.	

43 Peripheral Sensor Interface with Serial PHY Connection (PSI5-S)

The PSI5-S IP-module performs communication according to the PSI5 specification. It communicates with the external world via one ASC interface for all channels. This ASC is built into the PSI5-S module. If the optional bidirectional mode is used, the commands to the external Physical Layer Interface (PHY) are transmitted via this ASC.

Figure 657 shows a global view of the PSI5-S interface.

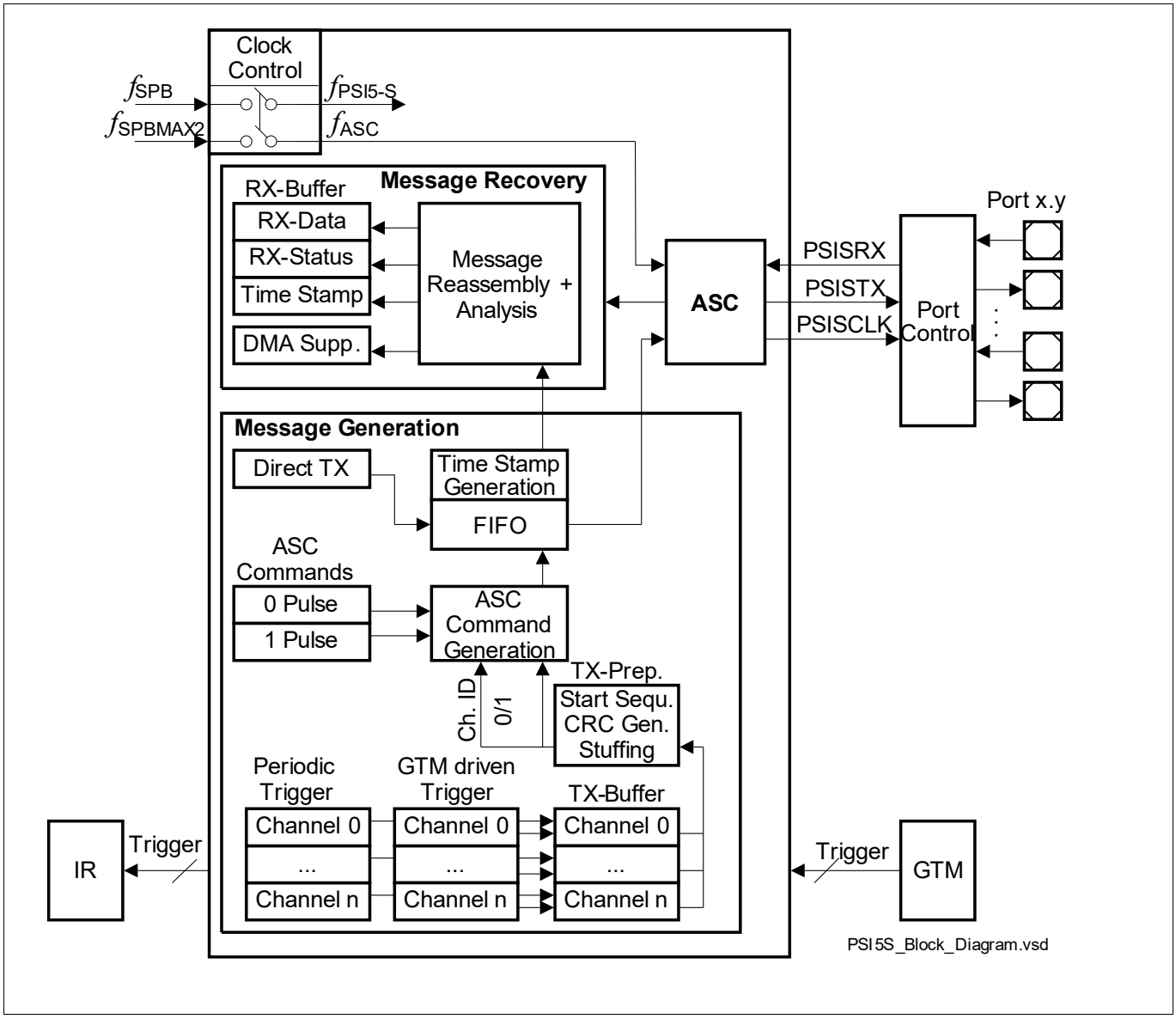


Figure 657 General Block Diagram of the PSI5-S Interface

43.1 Feature List

General Features

- ASC based communication with compatible PSI5-S PHY
- Baud rate Generator (up to 12,5) MBaud (@ 200 MHz ASC Sub Module Clock)
- Clock out generator to supply external PHY
- Conformance with PSI5 protocol specification
- Data rates of 125 kbit/s and 189 kbit/s supported (by external PHY already)
- 8 PSI5-S channels sharing one common ASC (inc. channel 0 frame 1 special use)
- Supports 6 sensor slots per channel
- Asynchronous and synchronous PSI5 data transmission modes
- 8 Interrupt or DMA triggers

Features of Message Recovery Block

- ASC format 10 Bit: 1 Start Bit, 8 Data bits, 1 Stop Bit (Up Stream)
- Each PSI5 Frame is transported in a Packet Frame consisting of 3 to 6 UART Frames transmitted back to back, i.e. with exactly one stop bit - no additional delay.
- Packet Frames are separated by a programmable idle time (1 … 16 idle bits)
- Assembly and decoding of Packet frames (envelope)
- Routing of PSI5 Frames to their referring channel receive buffer in system memory by hardware DMA support
- Checking CRC of Packet Frames (XCRC)
- Frames with XCRC NOK will be routed to channel 0 frame 1
- The external PHY sends all Frames of an Asynchronous Channel with FID = 1, all frames of this channel have the same length and structure as defined for FID 1. Thus it is sufficient to configure frame 1 for asynchronous channels.
- Configurable data length 8 … 28 bit + 3 bit PSI5 CRC or 1 bit PSI5 Parity
- CRC check of received PSI5 sensor data, CRC code still transparent
- No HW Support of extended serial data messaging according to SENT SAE J2716 JAN 2010. Messaging bits are transported fully transparent for SW decoding
- 24-Bit time stamp on sync pulse request via ASC (resolution: 1µs)
- Two independent time bases for Time Stamp: clocked by GTM (1 out of n GTM signals is selectable) or internal periodic trigger generator
- PHY answers to CPU commands with standard frames of fixed predefined length in channel 0 frame 0.
- Error bits in Packet Frame are presented fully transparent and an interrupt is issued.
- One Interrupt for error free Packet Frame, relevant errors are selectable
- One Interrupt for Packet Frame reception disregarding any error.

Features of Message Generation Block

- Support of ECU to Sensor communication
- One 32-bit send data register per channel for downstream data input
- Data is shifted out LSB first, MSB is filled with ‘0’ or ‘1’ with each shift (depending on bit coding method: Tooth Gap or Pulse Width)
- ASC format 11 Bit: 1 Start Bit, 8 Data bits, 1 Parity, 1 Stop Bit (Down Stream)

- Downstream commands of fixed format (3 MSB = Channel ID, 5 LSB = Command)
- All commands are randomly programmable and thus not interpreted by PSI5-S
- Downstream data transmission by 2 different ASC commands (short/no and long pulse)
- 8 common lines of FIFO for all channels, writable by
- PSI5 Message Generation
- Sync Pulse Generation (pre programmed Bytes for ‘0’ and ‘1’)
- CPU Commands
- Direct CPU Write Registers allow for insertion of commands into the FIFO for configuration and debugging
- 8 GTM inputs for Sync Pulse Triggering
- SW configurable staggering of Sync Pulse generation
- Generation of 3 or 6 bit CRC for downstream data
- Start sequence generator for downstream data (can be switched off)
- Bit stuffing generator for downstream data (can be switched off)
- CRC generator for downstream data (can be switched off)
- Internal Loop Back mode for check of CRC generator and SW development

43.2 Overview

The Peripheral Sensor Interface is an interface for automotive sensor applications. PSI5 is an open standard based on existing sensor interfaces for peripheral airbag sensors, already proven in millions of airbag systems.

PSI5 defines a current loop based serial communication link typically used to connect airbag sensors or other peripheral devices.

While the physical layer is done externally, this module manages communication with this PHY and the presentation of the data to the application. Note that there is no on chip PHY, i.e. the current to voltage and voltage to sync pulse translation is done externally.

Receive data on a PSI5-S channel can be set up according to the underlying application. In particular the number of bits forming one value is configured.

The message storage consists of a set of 3 32-bit registers for all channels: one PSI5-S Message Receive Data Register containing the received sensor data, one time stamp register containing the 24-bit value and one additional register containing status bits.

For synchronous communication mode, the module can consume sync pulses from the external GTM or generate periodic sync pulses by itself. It supports as well ECU to sensor communication by offering a 32-bit register, where messages can be set up by the CPU. All kinds of sync pulses are translated into a referring, programmable Byte to be sent via ASC.

The register set of the PSI5-S module can be accessed directly by the CPU for configuration, data read out and status query.

43.2.1 Definitions

SENT: Single Edge Nibble Transmission

CAN: Controller Area Network

LIN: Local Interconnect Network

ISO: International Organization for Standardization

PHY: Physical Layer Interface

ECU: Electronic Control Unit

FSM: Finite State Machine

ASIC: Application Specific Integrated Circuit

Nibble: Four Bit value between 0 and 15 = half a Byte = one character in hex (0 to F)

SOF: Start of Frame

EOF: End of Frame

STS: Start Sequence

AD: Address Data (e.g. of ECU to Sensor Frame)

SD: Send Data (Payload of ECU to Sensor Frame)

CRC: Cyclic Redundancy Check

UART: Universal Asynchronous Receiver Transmitter

43.3 Functional Description

PSI5-S is intended for use in applications where high resolution sensor data needs to be communicated from a sensor to an Engine Control Unit (ECU). It is intended as a replacement for the lower resolution methods of 10 bit A/D converters and PWM and as a simpler low cost alternative to CAN or LIN. The implementation assumes that the sensor is a smart sensor containing a microprocessor or dedicated logic device (ASIC) to create the signal.

Figure 658 shows a typical application in which a PSI5-S interface reads a sensor device.

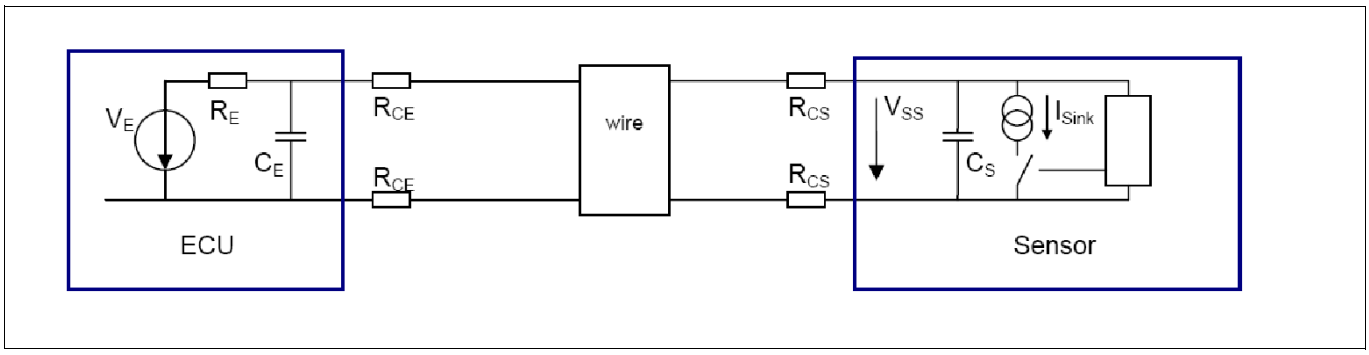


Figure 658 PSI5 to External Device Connection

PSI5-S communication is

- asynchronous, unidirectional from sensor to controller without any synchronization
- synchronous, bidirectional with a sync pulse from the ECU triggering messages
- synchronous, bidirectional, sync pulses additionally coding data from ECU to sensor

The sensor signal is transmitted as a series of current pulses in Manchester coding.

The sync pulses are transmitted by increasing the voltage of the current loop.

Figure 659 shows a typical application in which a PSI5 ECU reads multiple PSI sensor devices on a bus.

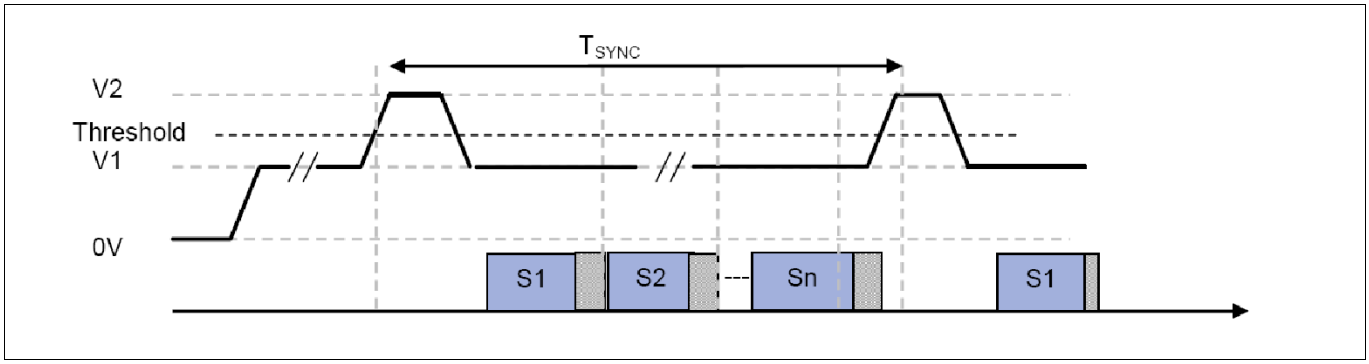


Figure 659 PSI5 Frames with Sync pulses

43.3.1 PSI5 ECU to Sensor Operation

ECU to Sensor communication is defined in PSI5 V2.0 2011-06. Data transmission and configuration of the sensor can be done by modulation of the Sync Pulses.

The Module supports two modes:

- Tooth Gap Method
- Pulse Width Method

With the Tooth Gap Method, a specific start condition and a start sequence switch the sensor in ECU to Sensor communication mode. In this mode, the sync pulses are no longer used as triggers for PSI5 Frames but are interpreted as data for the sensor. A pulse represents a 1 and a missing pulse represents a 0. This requires isochronous pulses.

With the Pulse Width Method a standard length pulse represents a 0, a longer pulse a 1. This way continuos data transmission can be achieved. The pulses can be non isochronous, e.g. dependent from the angle position of a rotating axis.

43.3.2 Frame Formats and Definitions

This section describes the frame formats and definitions of the PSI5-S protocol.

43.3.2.1 Communication between PSI5-S and PHY via UART

This chapter treats the communication on the UART connection.

43.3.2.1.1 “Packet Frames” received from PHY

Figure 660 and **Figure 661** show the layout and definitions of the “Packet Frames”. Packet Frames are 3 to 6 “UART Frames” carrying one “PSI5 Frame”. The format accepted by PSI5-S is explained here. A UART frame is one transmission unit on the ASC. On the receive path it consists of one Start Bit, 8 Data Bits and one Stop bit. Parity can be chosen optionally, e.g. for loop back tests.The figures show the minimum and maximum Packet Frame population with data. Where the figures show ‘Res0’ bits, these bits can be crowded by more data bits if configured so.

Detection of Frame Boundaries and Packet Frame Processing

- SOF (Start of Frame) is defined by a start bit detected after idle on ASC.
- Idle equals at least 2 Stop Bits.
  - The required idle time can be configured between 1 and 16 additional Stop Bits.
  - An idle state signal is provided internally by the ASC to the reassembly unit.

  - The idle signal starts on detection of the first stop bit and ends when the first slope of the next start bit has passed the 2 out of 3 majority vote input filter.
- If idle is detected before the minimum of 3 UART Frames was received, these UART Frames are copied to channel 0 frame 1 (non recoverable frames).
- Based on the ChID and FID received in the first UART frame, the module checks
  - If the referring channel is enabled and
  - If the FID is valid and configured (PDL > 0) and
  - What the number is of expected UART frames configured in UFC
  - If at least this number of UART frames is received without idle and
  - If the XCRC is correct
- If these checks pass
  - EOF is detected after the correct number of UART frames
  - Data is copied to RDR, interrupts, and the status are updated in RDS. In particular CRC/P check is executed. Time Stamp (TSM) and Target address (TAR) are updated. Depending from bit RCRAx.FIDS the FID might be manipulated.
  - The message recovery unit does not wait for idle to execute the steps above.
- If these checks fail (“non recoverable frames”)
  - EOF is detected after idle or a maximum of 6 UART Frames
  - The 1 to 6 UART Frames are stored in channel 0 frame 1.
- If more UART frames follow a passing or failing Packet Frame without idle,
  - EOF is detected after idle or a maximum of 6 UART Frames
  - The 1 to 6 UART Frames are stored in channel 0 frame 1
  - Reception is stopped
  - The next UART Frames following without idle time are ignored
  - Reception starts only after the idle time programmed in GCR.IDT is detected (“bubbling idiot” protection)

Non Recoverable Frame Presentation in RDR and RDS

- RDS.FID and ChID are derived from the first UART Frame received.
- RDR.RD[27:0] contains all bits of each UART Frame received after the header. I.e. 0 bits for 1 UART Frame,
  8 bits for 2 UART Frames,
  16 bits for 3 UART Frames,
  24 bits for 4 UART Frames,
  28 bits for 5 and 6 UART Frames.
- RDS.CRC is determined as follows:
  - If 6 UART Frames were received: bits [6:4] of the 5th UART Frame.
  - If 1 … 5 UART Frames where received: determined by PDL
  - PDL may not be reliable, but this way all data is covered either in the data field or in the XCRC field. In case PDL was correct, CRC is shown correctly.
- RDS.XCRC contains always the bits [7:2] of the last UART Frame received. Also if only 1 UART Frame was received.
- Note that XCRCI is always set as either XCRC was wrong or the number of UART frames received was wrong or it could not be determined.

Loop Back support

- Parity and number of stop bits need to be configured the same for sender and receiver in loop back mode. Parity control and polarity of parity can be configured separately each for TX and RX. This can be done before switching on loop back mode. Alternatively the parity control can be adjusted as well in one single register access together with switching the ASC part to loop back mode.
- Both ways are supported: add Parity bit to the receiver or use the transmitter without parity. (Standard use case is to use transmitter with parity)
- Data to be transmitted can be written directly to register CDW
- This way a synthetic Packet Frame can be sent to the reassembly unit for test purposes.
- The received loop back data is treated in the same way as the normal receive data. Good frames are received in their referring channel and slot. Frames with wrong XCRC, idle time or UFC are stored in channel 0 frame 1.

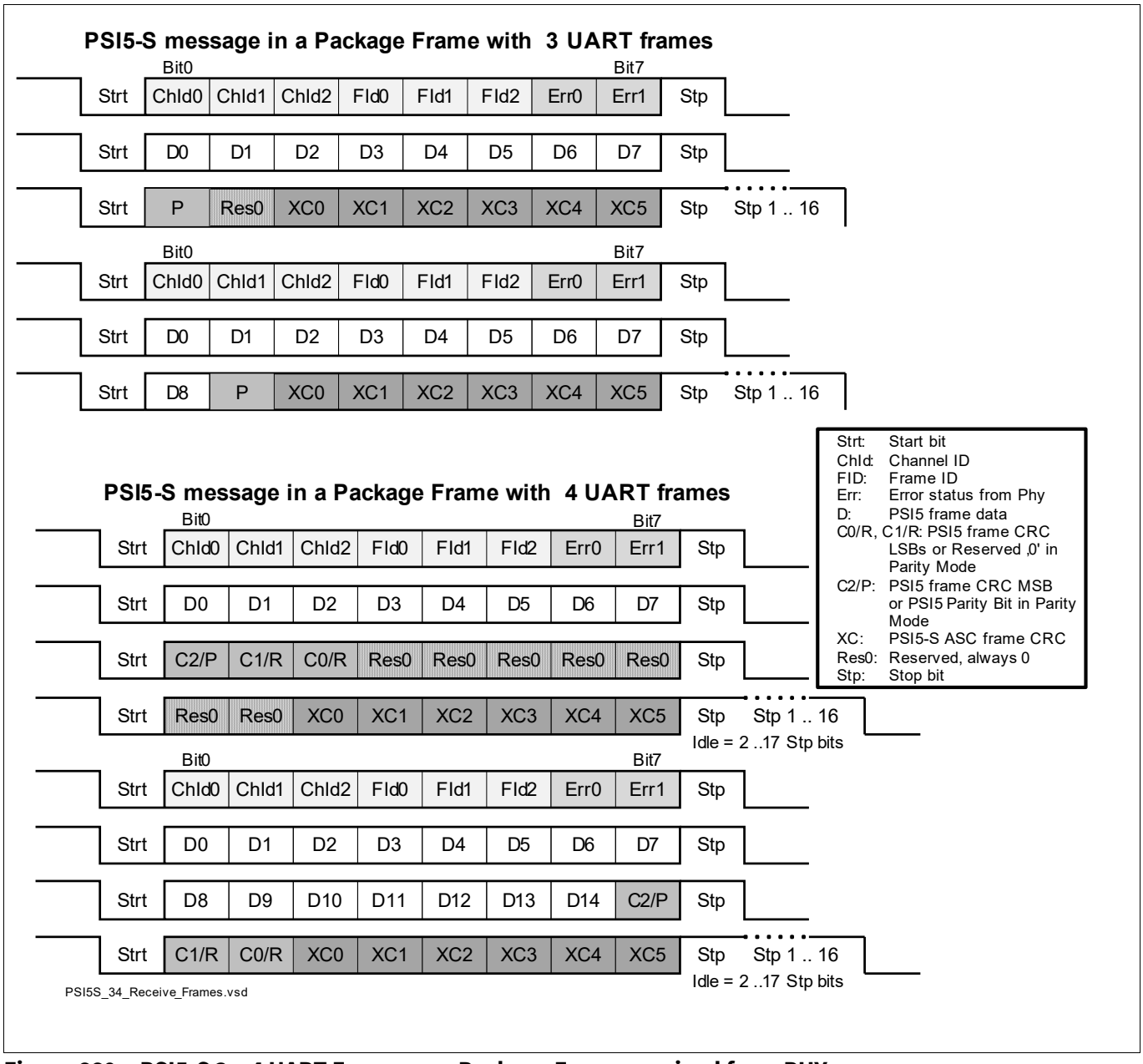


Figure 660 PSI5-S 3 + 4 UART Frames per Package Frame received from PHY

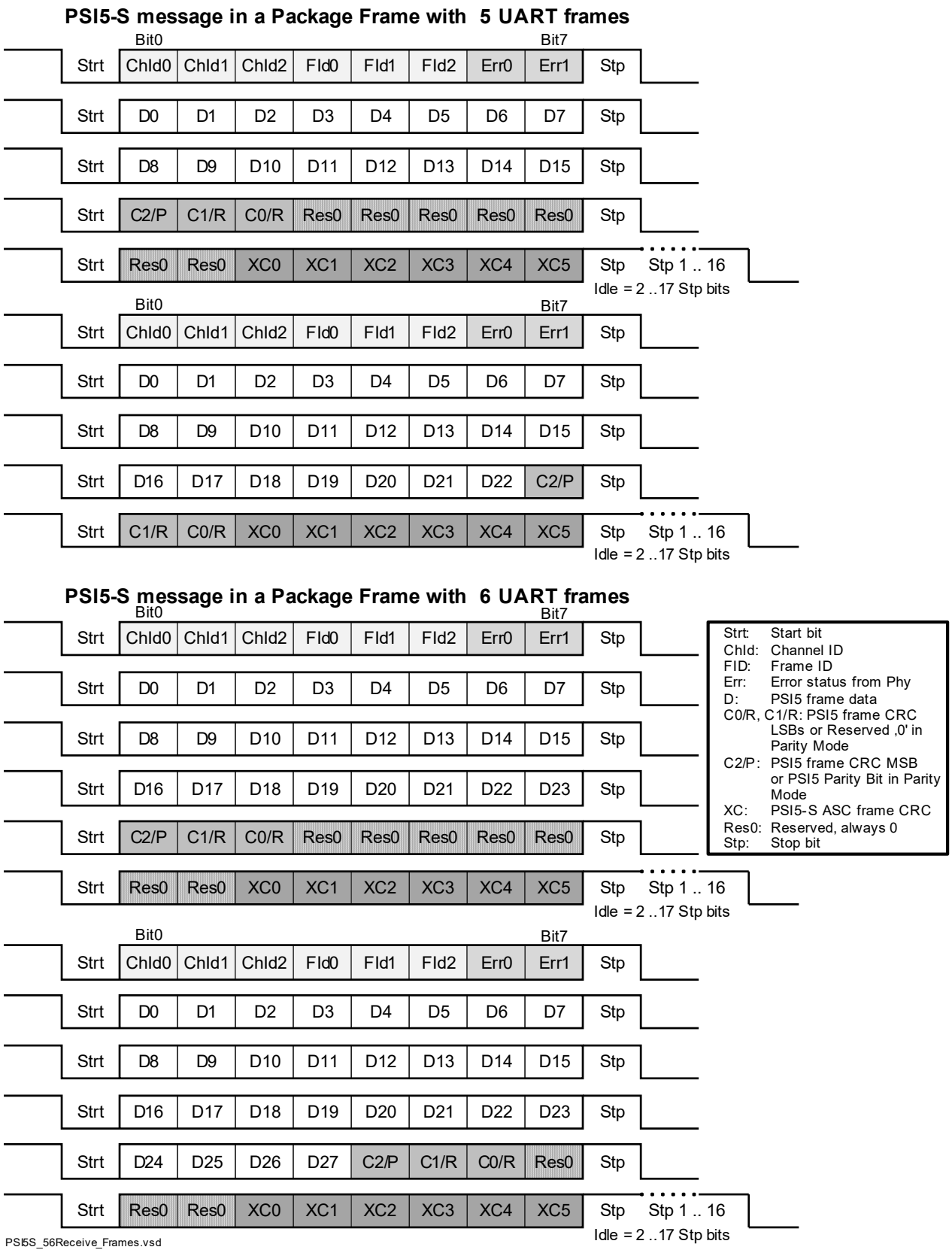


Figure 661 PSI5-S 5 + 6 UART Frames per Package Frame received from PHY

XCRC Content

XCRC is always sent at the end of the last UART Packet. For XCRC calculation, all bits of the preceding UART Frames that belong to the Packet Frame are relevant including potential stuffing bits (Res0). UART Start / Stop bits are excluded.

In detail the XCRC shall contain the header (ChID, FID, Err0, Err1), PSI5 frame data (D0 … Dx, PSI5 frame Parity Bit or CRC bits) and potential stuffing bits (Res0) between P/CRC and XCRC. Not included are start and stop bits of the UART packets and the 2 start bits (S0, S1) of the PSI5 frame.

The LSB of the first byte (ChId0) is fed first into the XCRC polynomial.

XCRC Calculation Method

XCRC equals the CRC defined in PSI5 Standard 2.0 “Frame format 4 ECU to Sensor Message CRC”, see **“ECU to Sensor Communication” on Page 11**.

43.3.2.1.2 PSI5-S UART Frames transmitted to PHY

Figure 662 shows the layout and definitions of the UART Frames sent by PSI5-S.

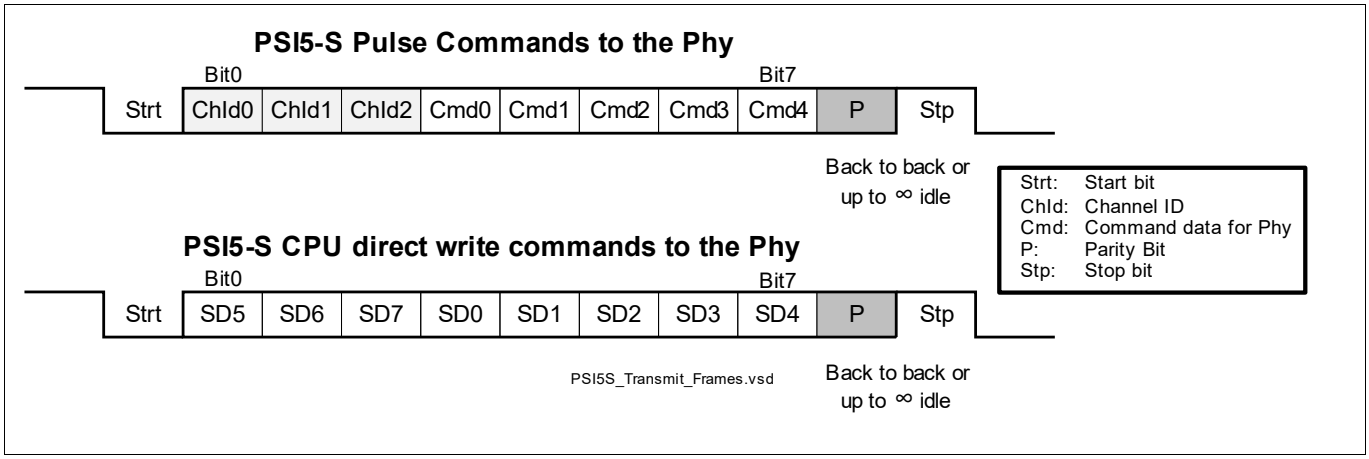


Figure 662 PSI5-S UART Frames transmitted to the PHY

43.3.2.2 Communication between PHY and Sensor (PSI5 Standard)

This chapter treats the frame formats on the PSI5 bus as short reference.

43.3.2.2.1 PSI5 Standard Frame Format

Figure 663 shows the principal layout and definitions of a PSI5 Frame. Note that the PSI5 standard specifies that the least significant bit is sent out first. See standard for CRC and Parity definition.

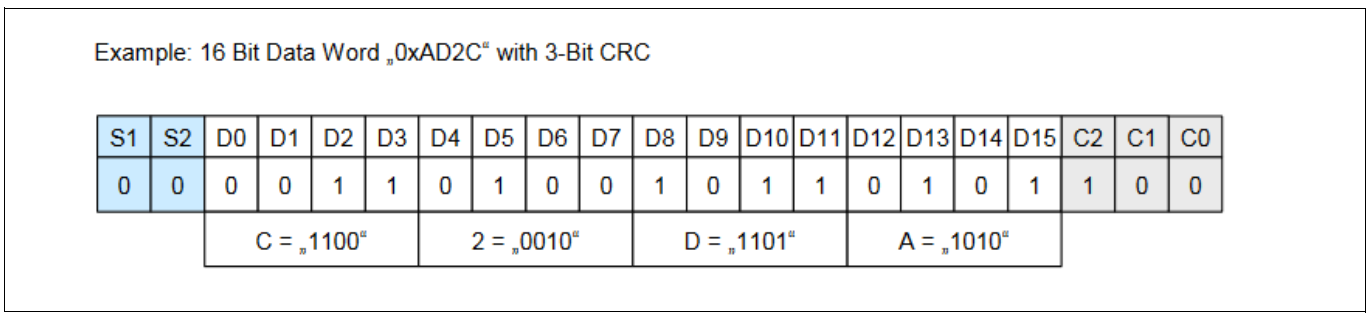


Figure 663 Standard PSI5 Frame Format

43.3.2.2.2 PSI5 Extended Frame Format

Figure 664 shows the layout and definitions of an Extended PSI5 Frame. Note that the PSI5 standard specifies that the least significant bit is sent out first.

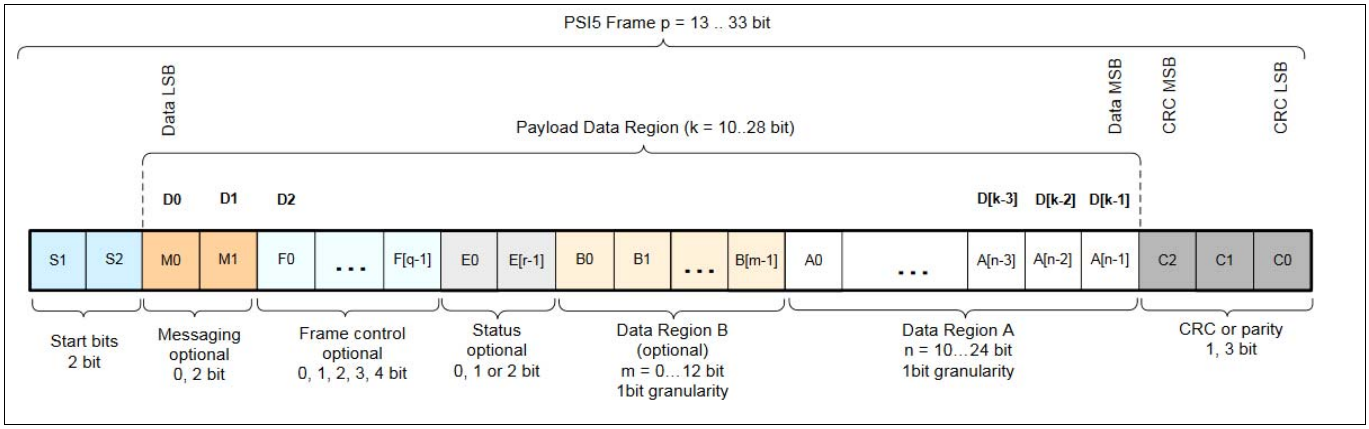


Figure 664 PSI5 Extended Frame Format

PSI5 Sensor to ECU Message CRC

The transmission of PSI5 data is checked by the following protection modes. The transmission error detection must be selectable:

1) 1-bit even parity

2) 3-bit CRC

The applied generator polynomial of the CRC is g(x) = 1 + x + x3 with a binary start value (seed) “111”. The transmitter shall extend the data bits by three zeros (as MSBs). This augmented data word shall be fed (LSB first) into the shift registers of the CRC check. Start bits shall be ignored in this check. When the last zero of the augmentation is pending on the input adder, the shift registers contain the CRC checksum. These three check bits shall be transmitted in reverse order (MSB first: C2, C1, C0)

43.3.2.2.3 Sync Pulses

Sync Pulses are used for two different purposes:

- Triggering a data frame for data acquisition from a sensor or
- ECU to sensor communication.

Synchronous Transmission

In the “synchronous” mode, the sensor (slave) starts to transfer a complete data frame only after a sync pulse is transmitted via ASC to the external PHY. The external PHY translates this ASC command into a voltage increase. The sensor then initiates a measurement and starts to calculate the new output data value. The data follows in a standard PSI5 Frame, starting with the 2 start bits, data and Parity or CRC. The timing diagram in **Figure 665** visualizes a synchronous transmission

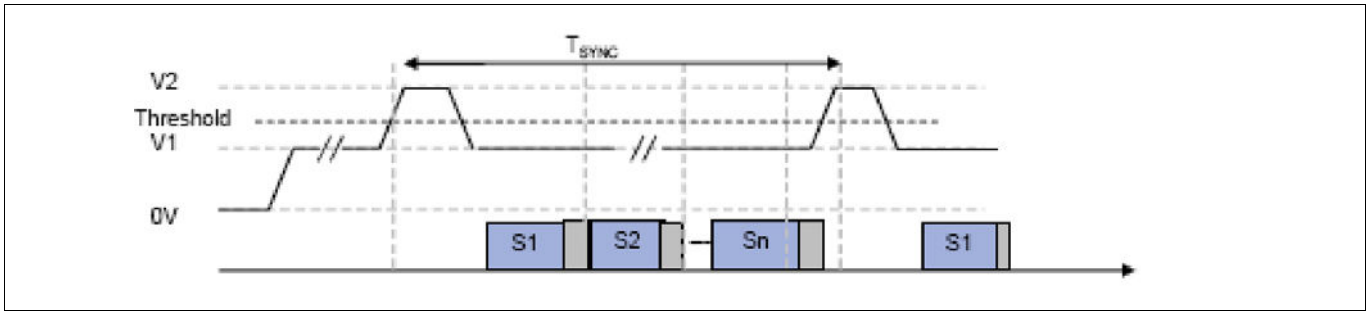


Figure 665 Synchronous Transmission

ECU to Sensor Communication

PSI5 V2.0 2011-06 defines 4 principle frame formats. Please refer to this standard document for more details. As the main target application is power train, only frame format 4 is cited here. This is the format for the power train sub standard.

Two methods for the ECU to Sensor Communication are supported:
Tooth Gap Method (see **Figure 666**) and
Pulse Width Method (see **Figure 667** and **Figure 668**) .

Tooth Gap Method

uses usually 31 sync pulses as start condition and start bits (0 1 0) to signal the start of the ECU to Sensor Communication. From this start on, the sync pulses are interpreted by the sensor as data. See standard for CRC and Parity definition.

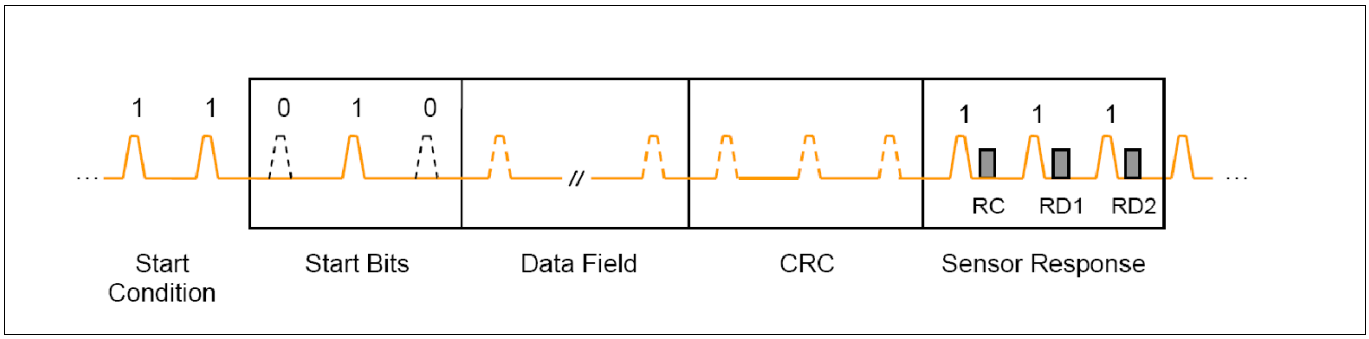


Figure 666 ECU to Sensor (Tooth Gap Method)

Pulse Width Method

allows for modulation of the sync pulse duration. This way bidirectional communication is possible continuously.

Sync Bits of frame format 4

A consecutive train of seven ones within the transmitted data could be misinterpreted as a frame start condition. Therefore a “sync bit” (logical 0) is inserted after every six bits of the sensor address and the payload data. With respect to the data transmission, these sync bits are treated like stuffing bits, which are removed at the receiver side.

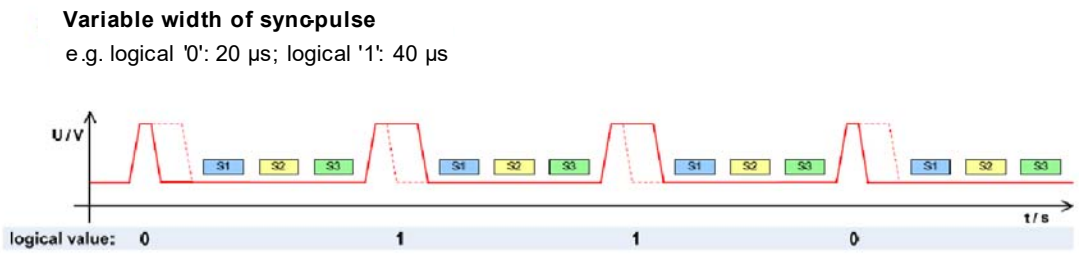


Figure 667 ECU to Sensor, Bit transmission (Pulse Width Method)

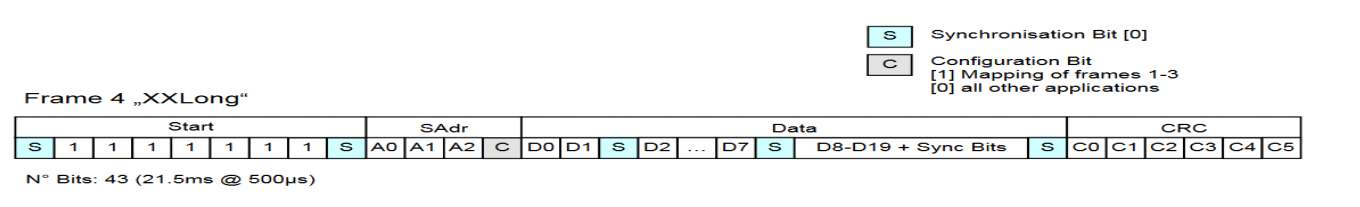


Figure 668 ECU to Sensor, Frame format 4 (power train)

Frame format 4 ECU to Sensor Message CRC

The ECU-to-Sensor Frame has a 6-bit CRC.
The generator polynomial of the CRC is
g(x)=x6 + x4 +x3 +1
with a binary CRC initialization value “010101”.
The transmitter extends the data bits by six zeros (as MSBs).
This augmented data word shall be fed (LSB first) into the shift registers of the CRC check. Start bits and sync bits are ignored in this check.

When the last zero of the augmentation is pending on the input adder, the shift registers contain the CRC checksum.

These six check bits shall be transmitted LSB first [C0, C1 … C5].This CRC value is computed as a function of the contents of all address and data bits and Sync bits (“0” bits). For purposes of the CRC calculation, the bits shall be ordered: m = [A0 A1 … A3 D0 D1 0 D2 … D19 0].

In this implementation the number of elements is configurable. While the standard is 4 address and 20 data bits, the module allows longer or shorter messages too.

43.3.3 Clock Generation

The PSI5-S module provides 2 central Time Stamp Counters and the central Time Stamp Clock f_{TS} to each channel. See **Figure 669**. The clock of each Time Stamp Counter is individually selectable: either f_{TS} or any of the GTM inputs can be selected to drive them.

This chapter shows in detail how f_{TS} is adjusted.

For details on the principles of a fractional divider, please refer to the SCU chapter of section “Clock Control”.

The PSI5-S module provides 2 internal clock signals centrally

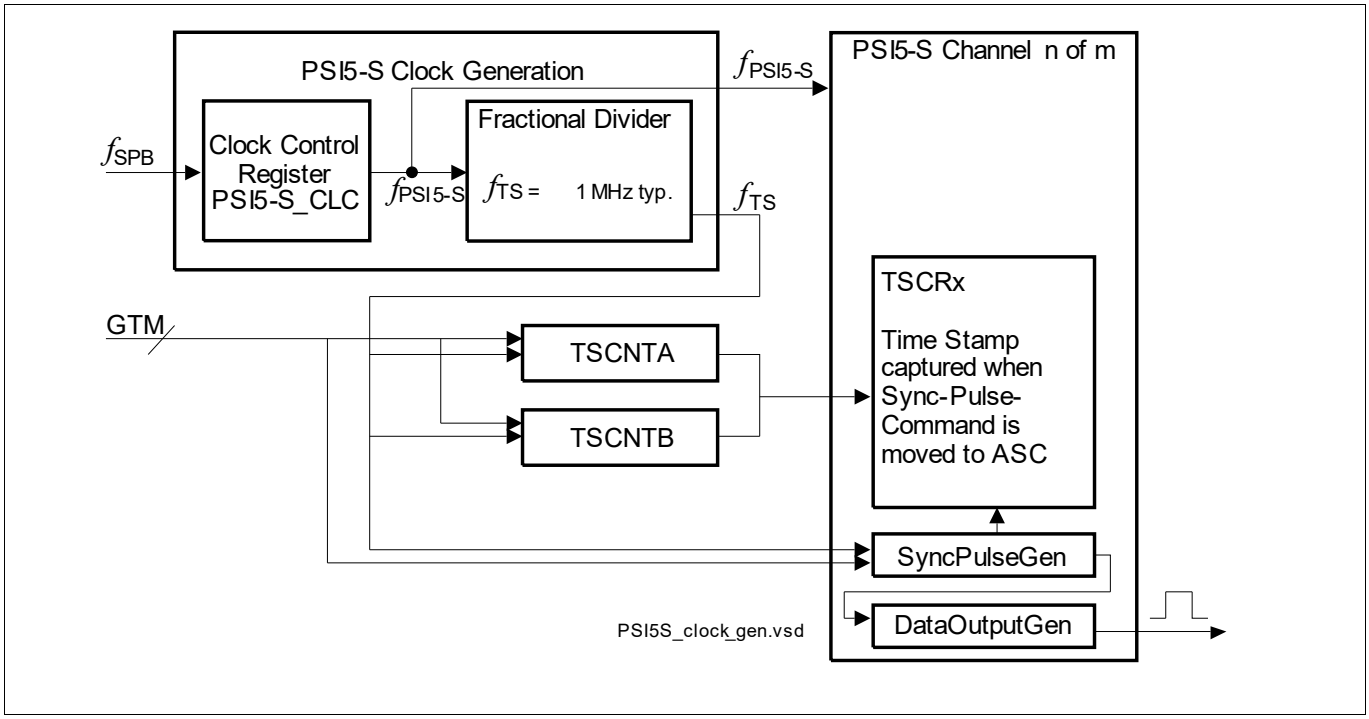


Figure 669 PSI5-S Module Clock Generation

- f_{PSI5-S}
  This is the module clock that is used inside the PSI5-S kernel for control purposes such as clocking of control logic and register operations. The frequency of f_{PSI5-S} is controlled by register FDR and derived from the system clock frequency f_{SPB}. The clock control register CLC makes it possible to enable/disable f_{PSI5-S} under certain conditions.

The following two formulas define the frequency of f_{PSI5-S}:

$f_{PSI5-S} = f_{SPB} / (1024 - PSI5\text{-}S\_FDR.STEP)$; FDR.DM = 01B (43.1)

$f_{PSI5-S} = f_{SPB} \times PSI5\text{-}S\_FDR.STEP / 1024$ with STEP = 0 … 1023; FDRT.DM = 10B (43.2)

- f_{TS}
  This clock is usually used to drive the central Time Stamp Counters.
  It can be selected to drive the global sync pulse time base and
  inside the PSI5-S channels to drive the local sync pulse generators and watch dog timers. The fractional divider register FDRT controls the frequency of f_{TS}.This is usually adjusted to 1 MHz / 1 µs period time.

The following two formulas define the frequency of f_{TS}:

$f_{TS} = f_{PSI5-S} / (1024 - PSI5\text{-}S\_FDRT.STEP)$; FDRT.DM = 01B (43.3)

$f_{TS} = f_{PSI5-S} \times PSI5\text{-}S\_FDRT.STEP / 1024$ with STEP = 0 … 1023; FDRT.DM = 10B (43.4)

43.3.3.1 Overview on Clocks in the System

To reduce system cost, the external PHY can be supplied with a clock from this device. EXTCLK1/2 (SCU), CMU_ECK (GTM) or by PSISCLK (PSI5-S). The range is limited to 20 … 25 MHz maximum for proper clock transmission. The PHY needs to provide 3 clocks from this: Sample clock for 189 kHz, and 125 kHz and the ASC oversampling clock.

A divider by 2 is added between the actual fractional divider block and the output signal $f_{PSISCLK}$ in order to achieve a duty cycle of 50% (plus tolerances due to fractional divider jitter). This is why the formulas contain the division by 2 at the end.

- $f_{PSISCLK}$
  This clock is used to drive the external PHY.
  The fractional divider register **FDO** controls the frequency of $f_{PSISCLK}$.

The following formula defines the frequency of $f_{PSISCLK}$:

$$f_{PSISCLK} = f_{ASC} / (2048 - PSI5\text{-}S\_FDO.STEP) / 2; DM = 01B \quad (43.5)$$

$$f_{PSISCLK} = f_{ASC} \times PSI5\text{-}S\_FDO.STEP / 2048 / 2 \text{ with } STEP = 0 \ldots 2047; DM = 10B \quad (43.6)$$

In order to relax the requirements with respect to ASC communication while allowing for high bit rates, some additional provisions have been made. The ASC Sub Module is supplied with a higher frequency (f_{ASC}) to allow for very high precision at high baud rates (4 - 12,5 M). As a second measure to achieve this, the fractional divider in ASC is longer than in the standard module. **Figure 670** illustrates the situation on PCB. Refer to the ASC sub chapter for baud rate calculations.

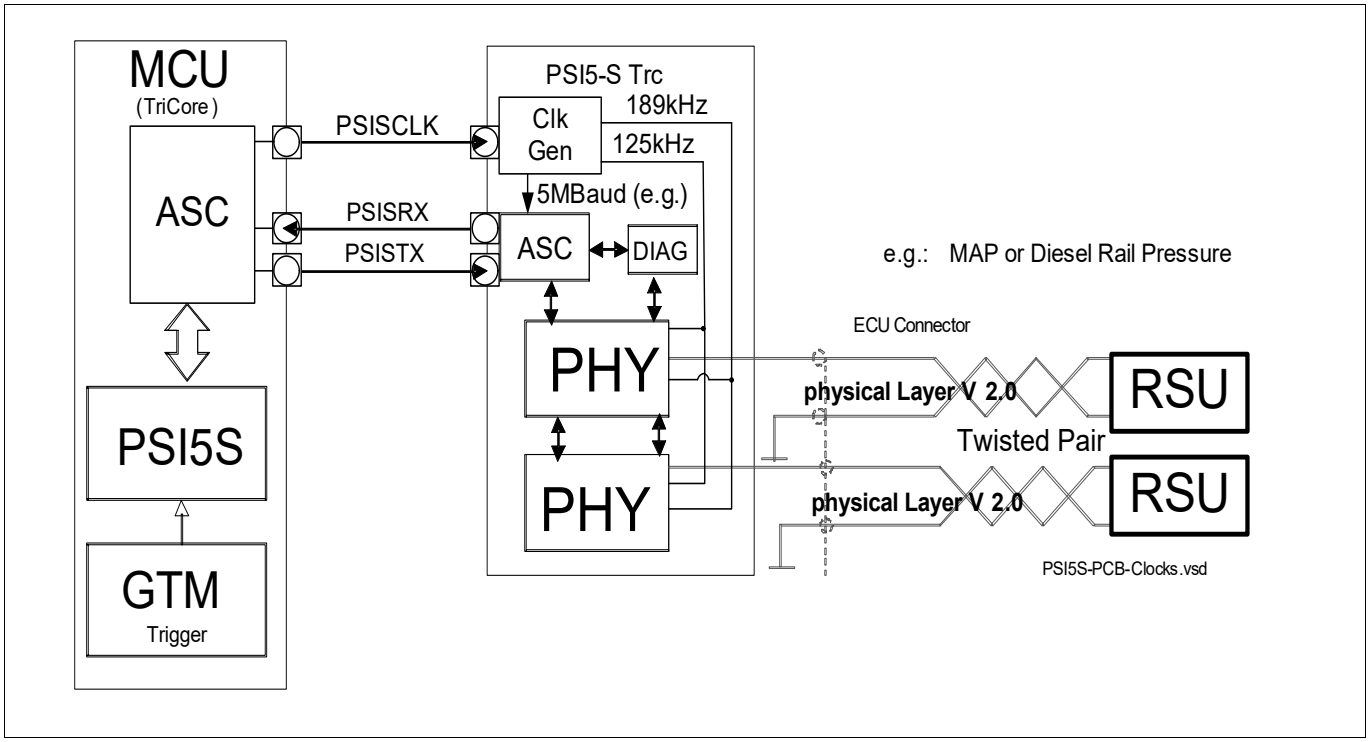


Figure 670 PSI5-S Clock Requirements of PHY

43.3.4 Time Stamp Generation

Two time bases can be selected for Time Stamping:

Any of TSCNTA and TSCNTB provide one Time Stamp Counter. Each of them can be configured to be driven by either f_{TS} or by an external timer, i.e. one of the GTM inputs. The frequency of f_{TS} is controllable by the fractional divider in register FDRT.

All of them can be cleared by SW single or synchronously.

The last sync pulse sent on a channel is stored in the sync pulse time stamp capture register TSCRx. Note that if RCRAx.TSTS is set, the Sync Pulses do not trigger the Time stamp capture. In this case, the Packet Frame reception triggers the capture. Non recoverable Packet Frames will be stored with the current value of TSCNTA/B directly copied to TSM. See RCRAx on **Page 60** and on for more details.

For each channel the capture time (TSCRx) is stored.

Figure 671 shows the time stamp generation:

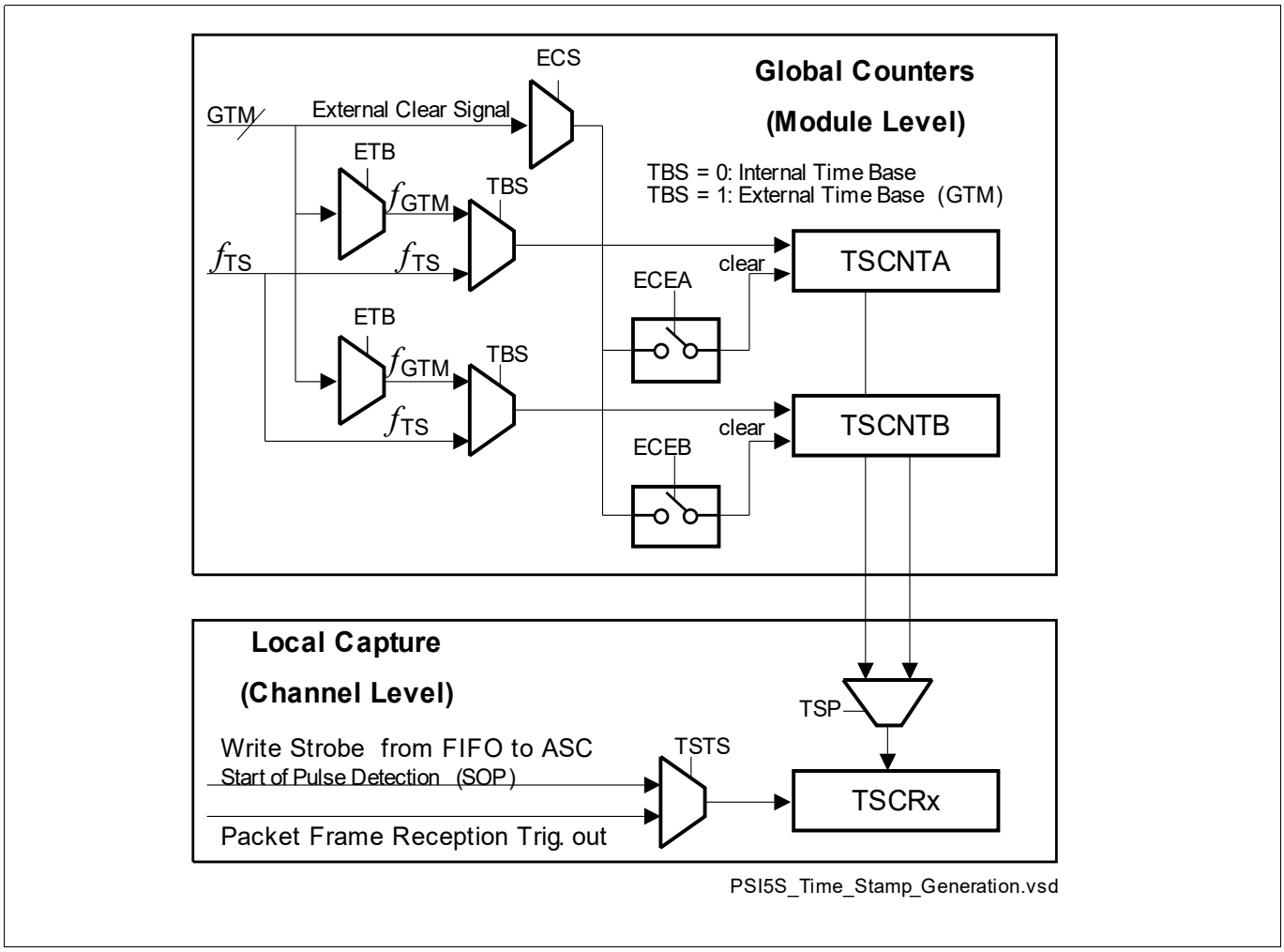


Figure 671 Time Stamp Generation

43.3.5 Watch Dog Timers

The value entered in WDTx.WDL defines the time in multiples of TTS (defined in FDRT) until expiry of the watch dog.

There are two modes of operation that can be selected with bit RCRAx.WDMS.

In normal operation an interrupt (TEI) is issued if the distance between two RDIs is longer than specified in WDL. In this mode, The internal watch dog timer is started automatically when the channel is enabled.

In synchronous mode, TEI is issued if the watch dog timer expires without reception of CHCI in time. I.e. the time from issuing the sync pulse to reception of the last expected frame configured in NFC.NFx was too long.

In both modes, only completely and correctly received Packet Frames on channel x can qualify, i.e. Packet Frames with XCRC checked ok. Otherwise the channel number can not be determined - the referring WDLx not be selected.

The internal watch dog timer is compared to WDL. A match

- triggers a TEI.
- restarts the watch dog timer. I.e. it clears the internal watch dog timer and it counts on from zero. TEI is repeated if no recoverable frames are received.

If no watch dog is needed, WDL is cleared, the internal watch dog timer is stopped and no check is performed.

Asynchronous mode of the Watch Dog Timers:

If RCRAx.WDMS is cleared the internal watch dog timer y is restarted on reception of a new recoverable frame (RDIx) for the referring channel. It is not restarted by a sync pulse and not stopped by a certain number of received Packet Frames.

The example below shows the use of the watch dog. The sync pulse is completely ignored by the watch dog system.

F2 is the only frame coming too late. All others start before the expiry of the watch dog.

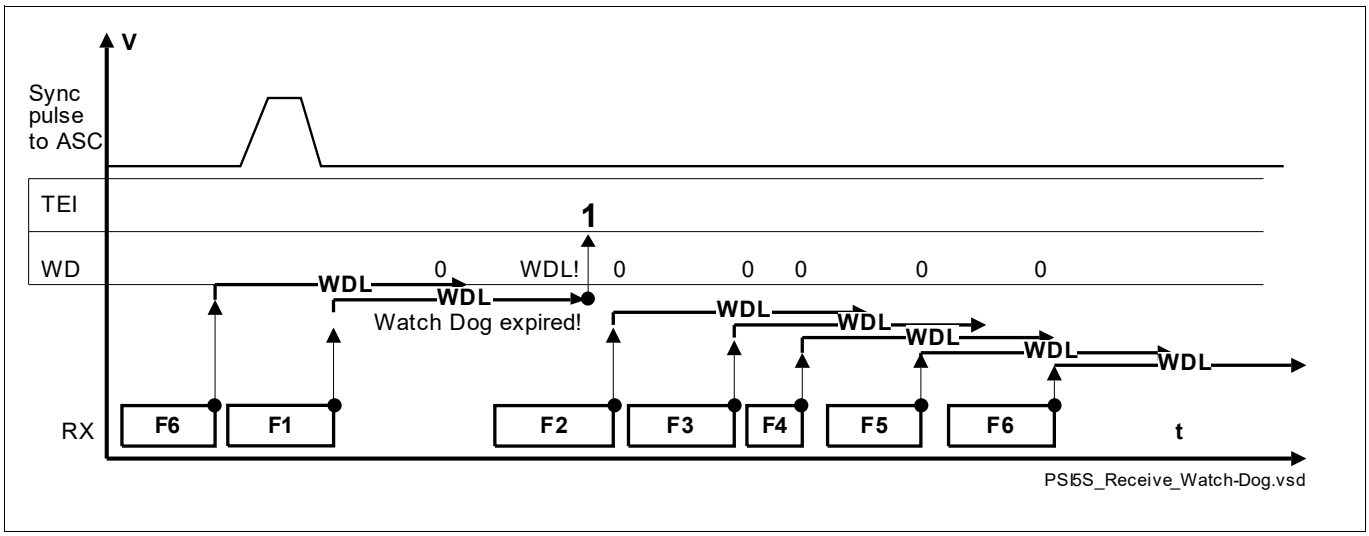


Figure 672 Watch Dog Timers

Synchronous mode of the Watch Dog Timers:

If RCRAx.WDMS is set the WDT is restarted on sync pulse and stopped at reception of the last frame configured in NFx. Register NFCx contains the bit fields NFx which configure the number of Packet Frames expected after the sync pulse. FCNT.FCx is the counter that is compared with NFx.

The Watch Dog Timer is restarted (cleared and set to run), each time a Sync Pulse is transferred for the referring channel from FIFO to ASC. The Watch Dog Timer of this channel is forced to stop if the number of a received Packet Frames (counted in FCNT.FCx matches the number configured in NFC.NFx. This supports in particular angle synchronous channels where the distance of packets can vary strongly with e.g. the roundings per minute of an axis.

The WDT can not stop in time if a non recoverable (and thus not countable) Packet Frame is received! If more than the configured number of Packet Frames are received, no dedicated interrupt is issued for this but the frame is still received correctly.

RCRAx.WDMS should be 0 on asynchronous channels. Otherwise, the internal Watch Dog Timer will be stopped after reception of the number of configured frames but never be restarted due to the lack of a Sync Pulse!

If a sensor is not present in a special configuration, NFx must be reduced accordingly! E.g. if out of F1 … F6 the F2 is missing, only 5 frames are configured.

Please refer to register **“NFC” on Page 58** for detailed description.

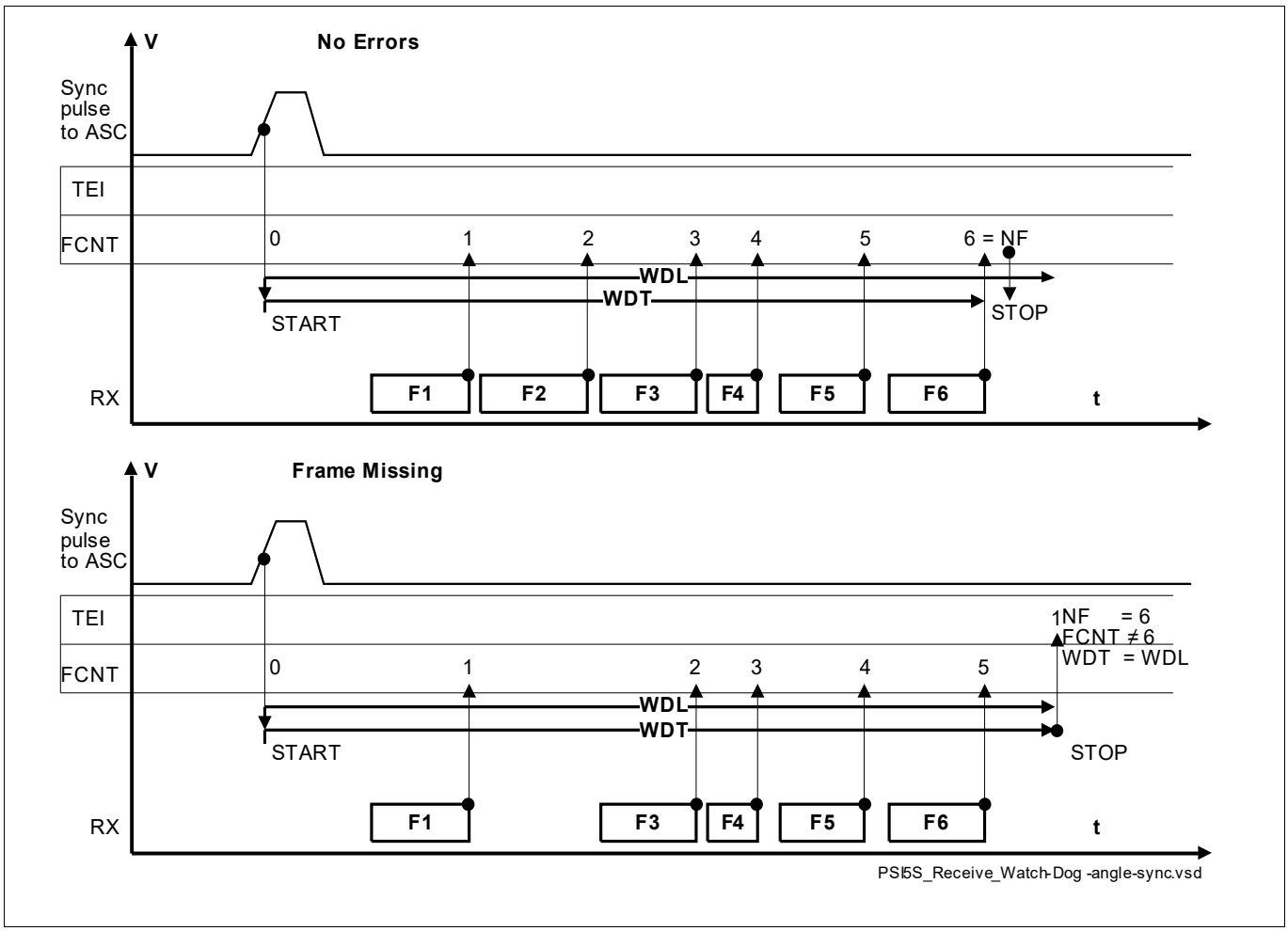


Figure 673 Watch Dog Timers in synchronous mode

43.3.6 Send Data

This chapter contains the details of how data to be sent is prepared. The scope extends to:

- Periodic Triggers
- Angle Synchronous Triggers
- ECU to Sensor communication

43.3.6.1 Channel Trigger

The Channel Trigger Value Register CTVx contains a two cell reset counter. It allows for setting up periodic sync pulses for channel x.

The Channel Trigger Value CTV is the compare value which restarts the counter CTC on equality and only on equality. At exactly this time, a sync pulse is triggered.
The compare on equality allows to preset CTC and thus configure an offset between the CTVx.CTCs. This allows for staggering periodic sync pulses that must have frequencies with common multiples (e.g. same frequency, double etc.).

In this document OFFSET = 65535 - “CTC preset value”

I.e. CTC preset is > CTV so that the the time to roll over is the offset.

Of course CTC preset can be < CTV so that this CTC has some advance.

Note that CTC can be written only if it is disabled by clearing GCR.ETCx.

Staggering the pulses might be required if the current source for the external PHYs can provide only limited current so that not all channels can be provided with the additional current that is required to apply the voltage increase representing a proper sync pulse.

For a synchronization of all counters CTVx.CTC, bits GCR.ETC0 … 7 can be used.

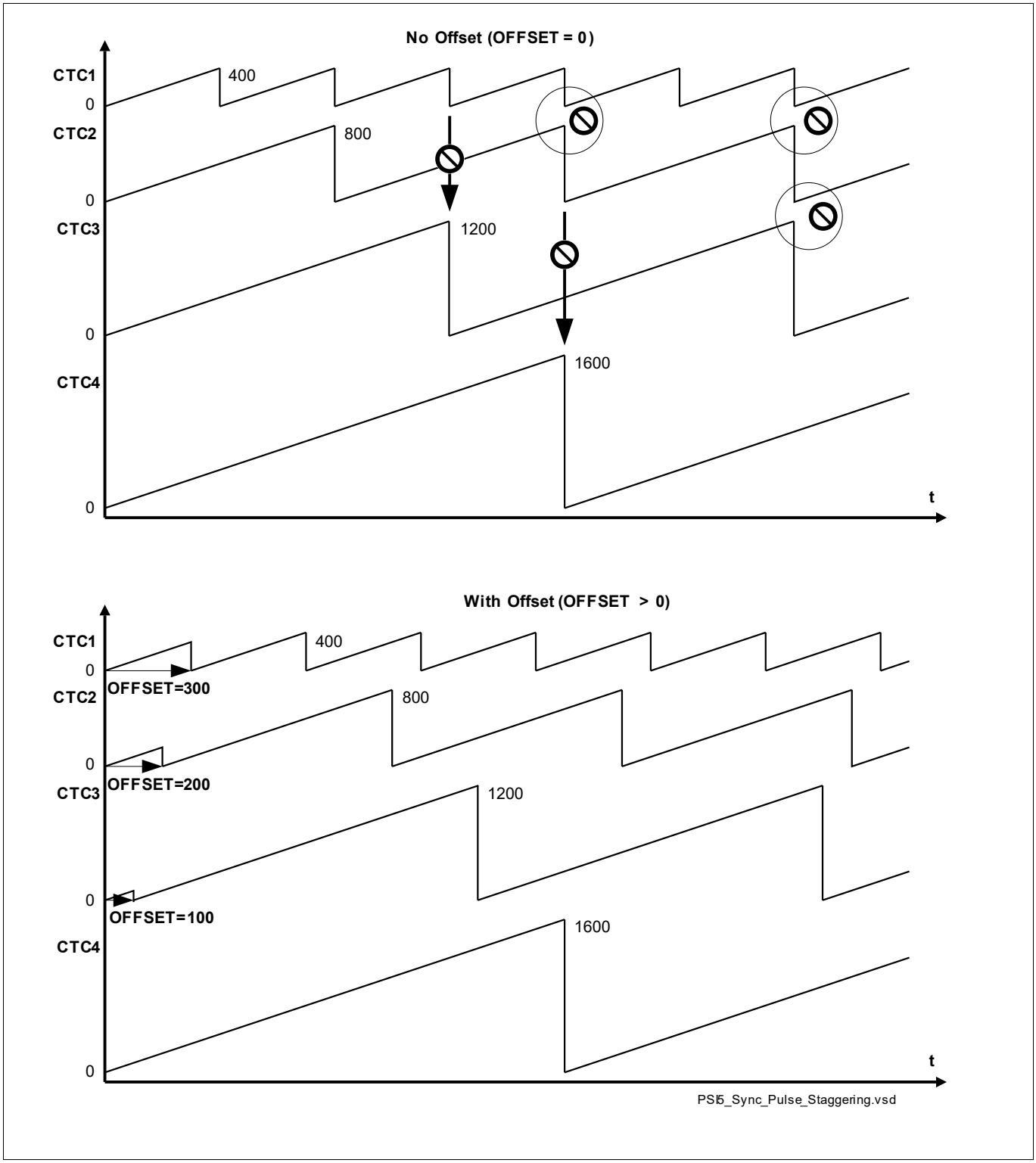


Figure 674 PSI5-S Pulse Staggering

43.3.6.2 Sync Pulse Control

The sync pulse generation provides the following functionality:

- periodic sync pulses, driven by a common module time base (PTE is set)
- angle synchronous sync pulses, driven by GTM (ETE is set)
- sync pulses for ECU to Sensor communication (Tooth Gap Method, EPS[0]=0)
- sync pulses for ECU to Sensor communication (PWM Method, EPS[0]=1)

It is possible to switch on and off the periodic trigger or the angle synchronous trigger at any time. The pulses will be modulated according to the selected standard. EPS selects between PWM method and Tooth Gap Method with pulse dropping.

An ECU to Sensor communication can be stopped at any point in time by setting the flush bit. This will clear the send register and flag TPF.

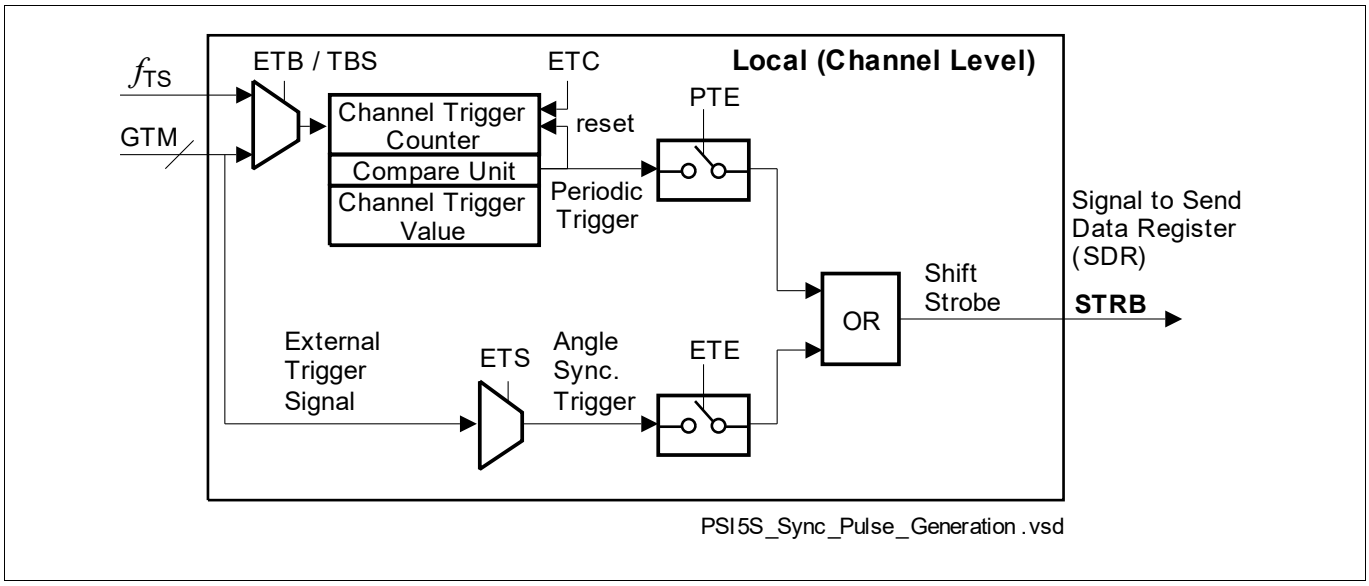


Figure 675 PSI5-S Sync Pulse Generation

43.3.6.3 Send Data Preparation

Figure 676 shows the mechanisms how ECU to sensor communication data is treated:

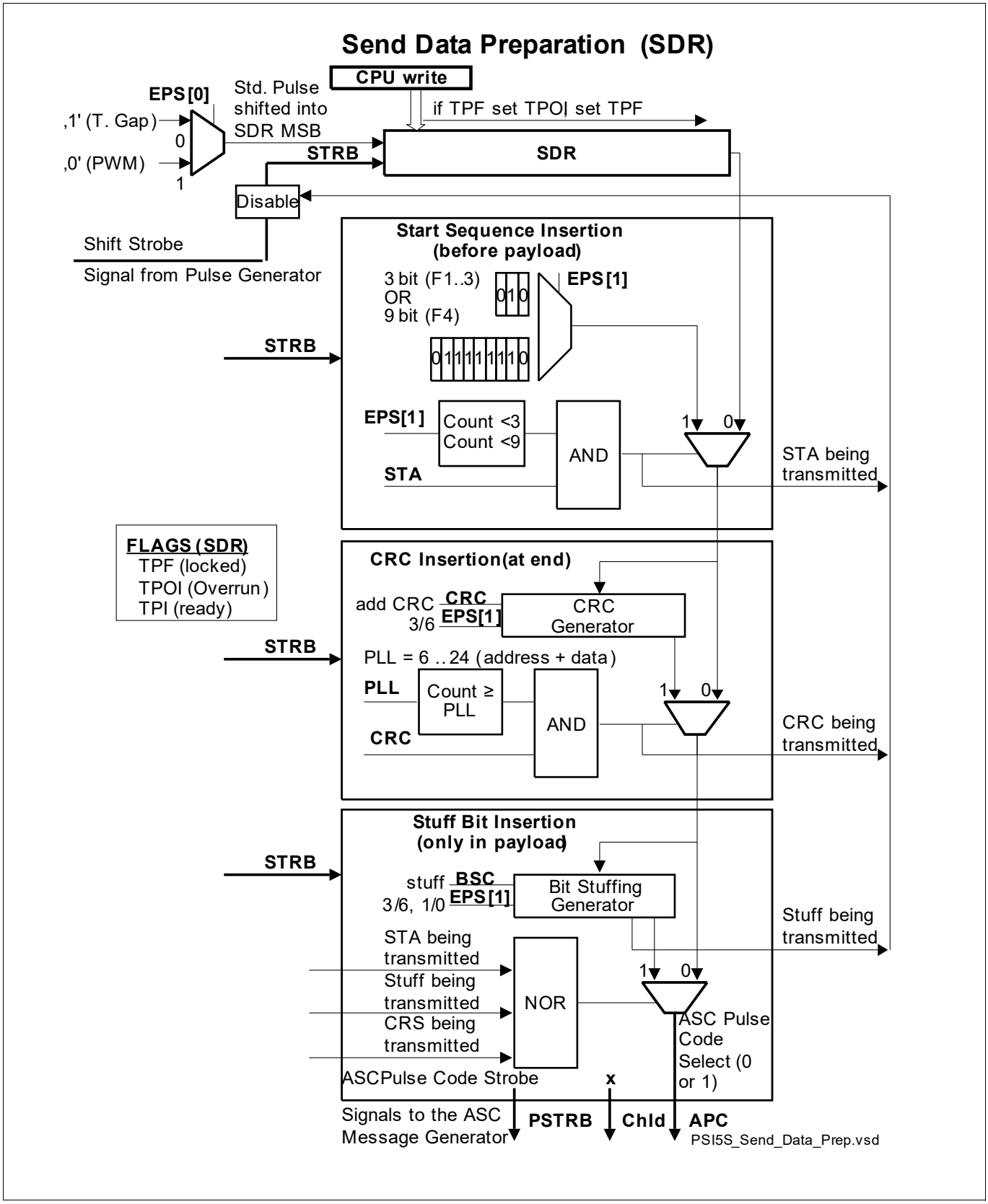


Figure 676 PSI5-S Send Data Preparation

43.3.7 Message Generation

Figure 677 shows the message generation logic. The generated ASC commands are sorted into an internal FIFO of 8 entries of 9 bits width. The MSB stores the information, if a command is generated from the sync pulse generation (ASC Pulse Code Strobe issued the write into FIFO) or if the CPU wrote a command. This is important for the Time Stamp generation. Usually the Time Stamp is captured when a Sync Pulse Command leaves the FIFO i.e. is copied to the ASC TX buffer. Commands stored in the FIFO can only trigger a Time Stamp capture if they are marked as Sync Pulse in bit 8. Nevertheless, the CPU can set MSB (bit 8) in Register CDW if the command is to be treated like a Sync Pulse Command and a Time Stamp should be captured. Note that if RCRAx.TSTS is set, the Sync Pulses do not trigger the Time stamp capture. In this case, the Packet Frame reception triggers the capture and bit 9 is ignored.
There is no further arbitration of the CPU and Message Generation Unit. The PHY answers CPU commands with standard frames of fixed predefined length in channel 0 frame 0.

Interrupt FOI is issued if a transfer to the FIFO was generated by the message generation unit or by CDW (CPU direct write register) while the FIFO was full. In a correct setup, this will never be the case as the bandwidth of the ASC is assumed to be by far higher than the write bandwidth in order to have short delays. In most cases too many write accesses to CDW will be the root cause. This exceptional condition can be handled either by waiting until the FIFO has transferred at least one command or by a module reset by SW. It should not occur in the application but FOI should help SW generation and debugging.

The transfer path of a command to the ASC could add jitter to sync pulses.

In case several channels request a pulse at exactly the same time, the channel with the lower channel number is served first and CDW has highest priority.

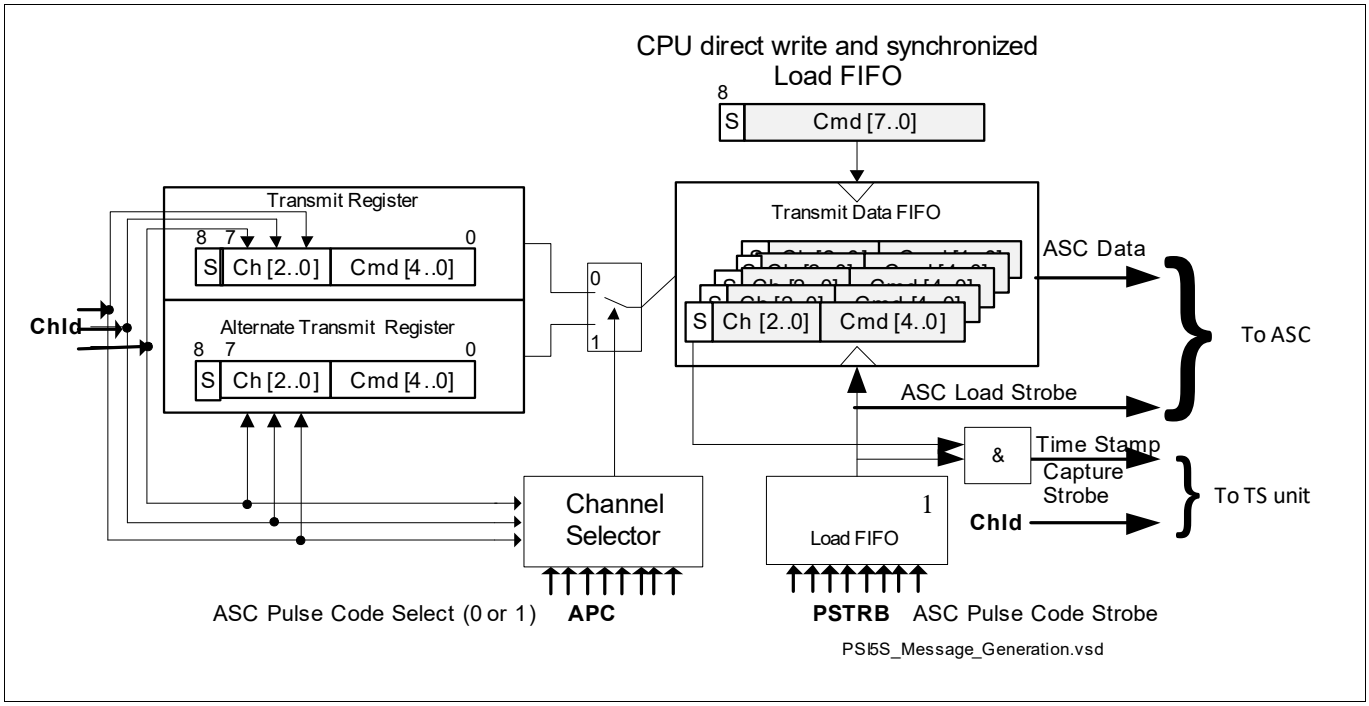


Figure 677 PSI5-S Message Generation Logic

43.3.8 DMA Support

The received frames come in sequentially and are XCRC checked. The complete reassembled frame including Channel Id (ChId), Frame ID (FId) and status are available at RDR and RDS. TSM shows the latest time stamp of the currently received channel.

RDR, RDS and TSM are located in 3 sequentially addressed registers. This way, they are easily readable by DMA. See **Figure 678**.

43.3.8.1 Single DMA, 8 dedicated DMAs

With one single DMA channel, one common frame buffer in the system memory can be build up. All frames can be collected from RDR, RDS and TSM. The DMA source address is programmed to be always RDS, block transfer is set to 3 words with auto address increment. The target is set to be automatically incremented. This results in one buffer in system memory with all frames from all channels.

Individual buffers for each channel can be build up if more DMA channels are used. Each channel can generate its own receive triggers RDI and RSI. Thus individual DMAs can be set up for each channel similarly to the single DMA use case but with individual target addresses for each channel.

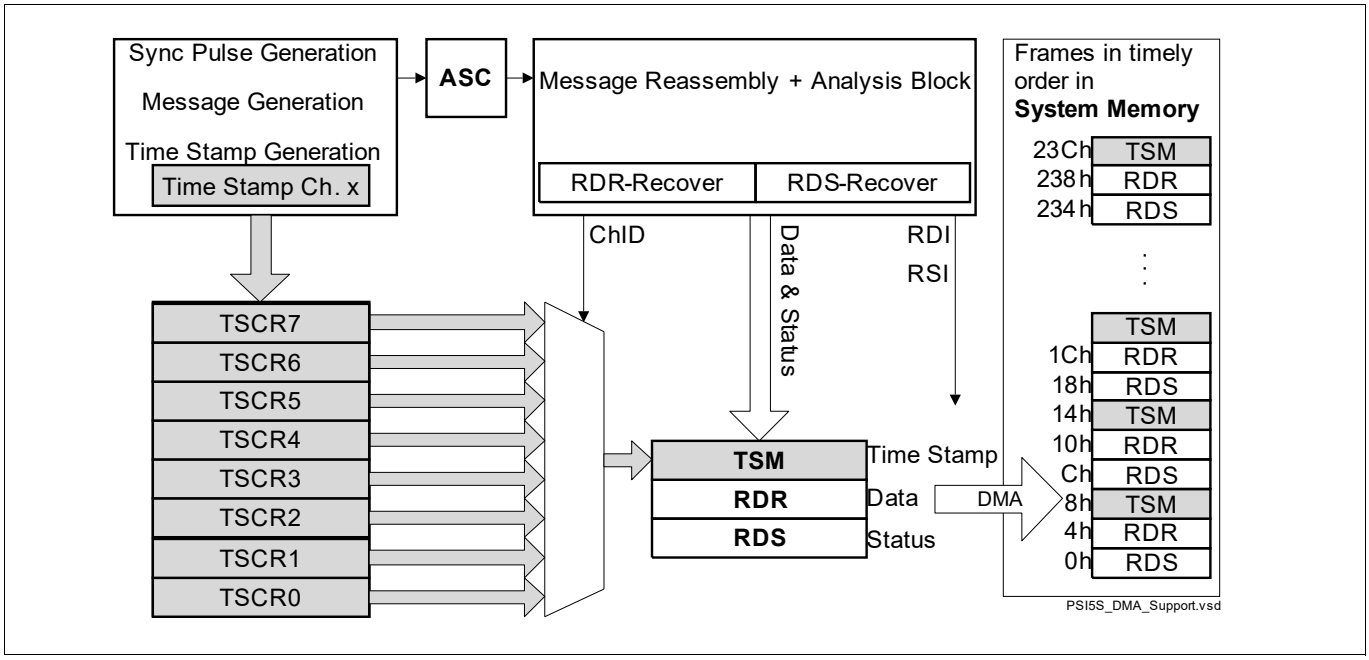


Figure 678 PSI5-S Single DMA Support

43.3.8.2 Two daisy chained DMAs

The PSI5-S supports sorting by channel with only two DMA channels. Register TAR contains the target address that is to be copied by the first DMA into the target address register of the second DMA. Source and target of the first DMA are fixed (DMA register SADR and DADR). The target address offered in TAR is calculated by PSI5-S by adding an offset value to the programmable base address defined in BAR. This allows for locating the channel buffers at any location in system memory.

The resulting base + offset is presented in TAR. The target address depends on:

- BAR (base address)
- ChID: offset of channel number currently received (channel number x 6 (buffer lines) x 3 (words per frame) x 4 Bytes per word = 72 Bytes x channel number. This allows to receive 6 frames per channel which equals the maximum number of PSI5 Frames per Sync Pulse.
- FID: offset of frame ID currently received. (3 words i.e. 12 Bytes per frame)
  Note that the frame ID used for this sorting depends from RCRAx.FIDS! In particular in asynchronous mode, FIDS needs to be set so that the FID from the Packet Frame is replaced by a rolling number.
- Note that the message reassembly unit forces non recoverable messages to be stored at the address for ChID = 0, FID = 1 but with original ChID and FID! I.e. RDS contains the CHID and FID as received in the first UART Frame for better debugging!
- The source address of the second DMA needs to be programmed to be fixed to the address of RDS. Use shadow control to “wrap around” the source address. (DMA_MExADICR.SHCT must be set to 0x0101b).
  Block transfer with address auto incrementing must be selected and the number of transactions programmed to 3 (see DMA chapter, DMA_MExCHCR.BLKM) 32 bit (the data width is defined in DMA_MExCHCR.CHDW) accesses.
  Use shadow control to “wrap around” the source address. (DMA_MExADICR.SHCT must be set to 0x0101b).

The second DMA must be of the next lower priority than the first. Selecting hardware request input via CHCRxz.PRSEL allows for daisy chaining the two DMA channels. See **Figure 679**.

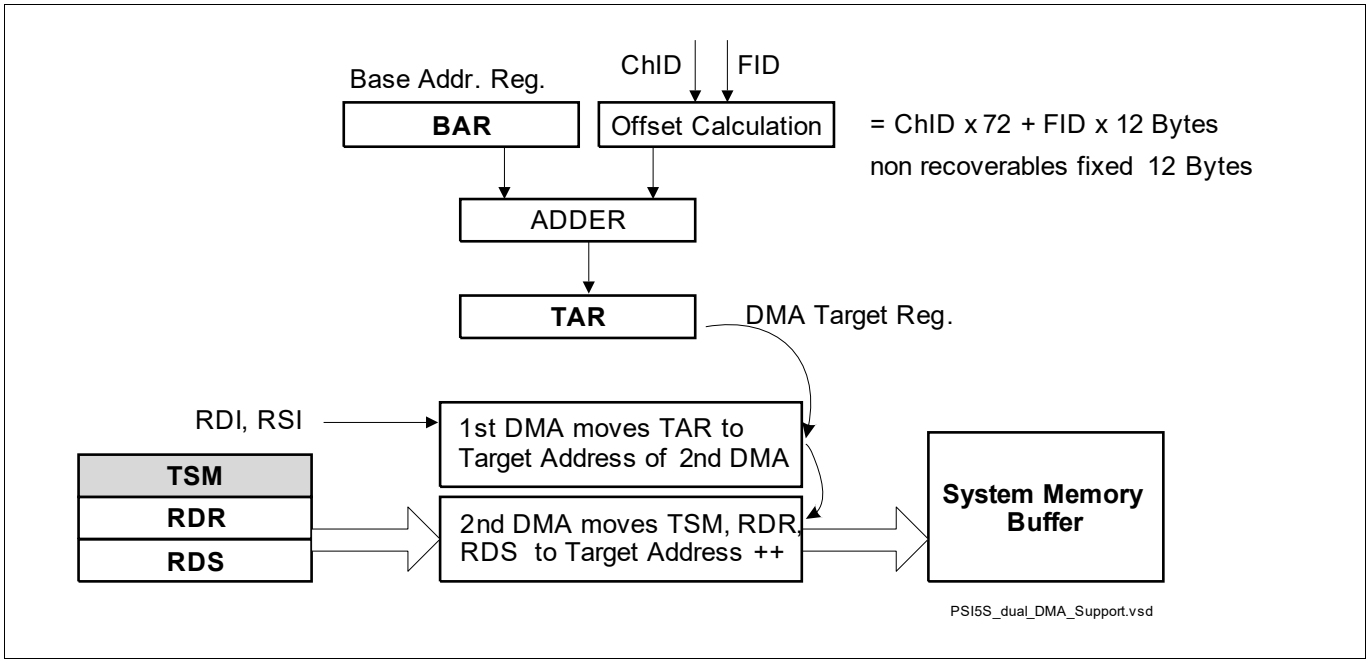


Figure 679 PSI5-S Dual DMA Support (daisy chained)

Table 430 shows the detailed mapping of the received frames resulting from the described calculation rules:

Table 430 Frame Mapping

Offset (Dec)	Offset (Hex)	Channel/Frame	Content
0	0_H	Channel 0 frame 0	Status (Reserved for Transceiver Messages)
4	4_H	Channel 0 frame 0	Data (Reserved for Transceiver Messages)
8	8_H	Channel 0 frame 0	Time Stamp (Reserved for Transceiver Messages)
12	C_H	Channel 0 frame 1	Status (Reserved for non recoverable Messages)
16	10_H	Channel 0 frame 1	Data (Reserved for non recoverable Messages)
20	14_H	Channel 0 frame 1	Time Stamp (Reserved for non recoverable Messages)
24	18_H	Channel 0 frame 2	Status
28	$1C_H$	Channel 0 frame 2	Data
32	20_H	Channel 0 frame 2	Time Stamp
36	24_H	Channel 0 frame 3	Status
40	28_H	Channel 0 frame 3	Data
44	$2C_H$	Channel 0 frame 3	Time Stamp
48	30_H	Channel 0 frame 4	Status

Table 430 Frame Mapping (cont'd)

Offset (Dec)	Offset (Hex)	Channel/Frame	Content
52	34_H	Channel 0 frame 4	Data
56	38_H	Channel 0 frame 4	Time Stamp
60	$3C_H$	Channel 0 frame 5	Status
64	40_H	Channel 0 frame 5	Data
68	44_H	Channel 0 frame 5	Time Stamp
72	48_H	Channel 1 frame 0	Status
76	$4C_H$	Channel 1 frame 0	Data
80	50_H	Channel 1 frame 0	Time Stamp
84	54_H	Channel 1 frame 1	Status
88	58_H	Channel 1 frame 1	Data
92	$5C_H$	Channel 1 frame 1	Time Stamp
96	60_H	Channel 1 frame 2	Status
100	64_H	Channel 1 frame 2	Data
104	68_H	Channel 1 frame 2	Time Stamp
108	$6C_H$	Channel 1 frame 3	Status
112	70_H	Channel 1 frame 3	Data
116	74_H	Channel 1 frame 3	Time Stamp
120	78_H	Channel 1 frame 4	Status
124	$7C_H$	Channel 1 frame 4	Data
128	80_H	Channel 1 frame 4	Time Stamp
132	84_H	Channel 1 frame 5	Status
136	88_H	Channel 1 frame 5	Data
140	$8C_H$	Channel 1 frame 5	Time Stamp
144	90_H	Channel 2 frame 0	Status
148	94_H	Channel 2 frame 0	Data
152	98_H	Channel 2 frame 0	Time Stamp
156	$9C_H$	Channel 2 frame 1	Status
160	$A0_H$	Channel 2 frame 1	Data
164	$A4_H$	Channel 2 frame 1	Time Stamp
168	$A8_H$	Channel 2 frame 2	Status
172	AC_H	Channel 2 frame 2	Data
176	$B0_H$	Channel 2 frame 2	Time Stamp
180	$B4_H$	Channel 2 frame 3	Status
184	$B8_H$	Channel 2 frame 3	Data
188	BC_H	Channel 2 frame 3	Time Stamp
192	$C0_H$	Channel 2 frame 4	Status
196	$C4_H$	Channel 2 frame 4	Data
200	$C8_H$	Channel 2 frame 4	Time Stamp

Table 430 Frame Mapping (cont'd)

Offset (Dec)	Offset (Hex)	Channel/Frame	Content
204	CC_H	Channel 2 frame 5	Status
208	$D0_H$	Channel 2 frame 5	Data
212	$D4_H$	Channel 2 frame 5	Time Stamp
216	$D8_H$	Channel 3 frame 0	Status
220	DC_H	Channel 3 frame 0	Data
224	$E0_H$	Channel 3 frame 0	Time Stamp
228	$E4_H$	Channel 3 frame 1	Status
232	$E8_H$	Channel 3 frame 1	Data
236	EC_H	Channel 3 frame 1	Time Stamp
240	$F0_H$	Channel 3 frame 2	Status
244	$F4_H$	Channel 3 frame 2	Data
248	$F8_H$	Channel 3 frame 2	Time Stamp
252	FC_H	Channel 3 frame 3	Status
256	100_H	Channel 3 frame 3	Data
260	104_H	Channel 3 frame 3	Time Stamp
264	108_H	Channel 3 frame 4	Status
268	$10C_H$	Channel 3 frame 4	Data
272	110_H	Channel 3 frame 4	Time Stamp
276	114_H	Channel 3 frame 5	Status
280	118_H	Channel 3 frame 5	Data
284	$11C_H$	Channel 3 frame 5	Time Stamp
288	120_H	Channel 4 frame 0	Status
292	124_H	Channel 4 frame 0	Data
296	128_H	Channel 4 frame 0	Time Stamp
300	$12C_H$	Channel 4 frame 1	Status
304	130_H	Channel 4 frame 1	Data
308	134_H	Channel 4 frame 1	Time Stamp
312	138_H	Channel 4 frame 2	Status
316	$13C_H$	Channel 4 frame 2	Data
320	140_H	Channel 4 frame 2	Time Stamp
324	144_H	Channel 4 frame 3	Status
328	148_H	Channel 4 frame 3	Data
332	$14C_H$	Channel 4 frame 3	Time Stamp
336	150_H	Channel 4 frame 4	Status
340	154_H	Channel 4 frame 4	Data
344	158_H	Channel 4 frame 4	Time Stamp
348	$15C_H$	Channel 4 frame 5	Status
352	160_H	Channel 4 frame 5	Data

Table 430 Frame Mapping (cont'd)

Offset (Dec)	Offset (Hex)	Channel/Frame	Content
356	164_H	Channel 4 frame 5	Time Stamp
360	168_H	Channel 5 frame 0	Status
364	$16C_H$	Channel 5 frame 0	Data
368	170_H	Channel 5 frame 0	Time Stamp
372	174_H	Channel 5 frame 1	Status
376	178_H	Channel 5 frame 1	Data
380	$17C_H$	Channel 5 frame 1	Time Stamp
384	180_H	Channel 5 frame 2	Status
388	184_H	Channel 5 frame 2	Data
392	188_H	Channel 5 frame 2	Time Stamp
396	$18C_H$	Channel 5 frame 3	Status
400	190_H	Channel 5 frame 3	Data
404	194_H	Channel 5 frame 3	Time Stamp
408	198_H	Channel 5 frame 4	Status
412	$19C_H$	Channel 5 frame 4	Data
416	$1A0_H$	Channel 5 frame 4	Time Stamp
420	$1A4_H$	Channel 5 frame 5	Status
424	$1A8_H$	Channel 5 frame 5	Data
428	$1AC_H$	Channel 5 frame 5	Time Stamp
432	$1B0_H$	Channel 6 frame 0	Status
436	$1B4_H$	Channel 6 frame 0	Data
440	$1B8_H$	Channel 6 frame 0	Time Stamp
444	$1BC_H$	Channel 6 frame 1	Status
448	$1C0_H$	Channel 6 frame 1	Data
452	$1C4_H$	Channel 6 frame 1	Time Stamp
456	$1C8_H$	Channel 6 frame 2	Status
460	$1CC_H$	Channel 6 frame 2	Data
464	$1D0_H$	Channel 6 frame 2	Time Stamp
468	$1D4_H$	Channel 6 frame 3	Status
472	$1D8_H$	Channel 6 frame 3	Data
476	$1DC_H$	Channel 6 frame 3	Time Stamp
480	$1E0_H$	Channel 6 frame 4	Status
484	$1E4_H$	Channel 6 frame 4	Data
488	$1E8_H$	Channel 6 frame 4	Time Stamp
492	$1EC_H$	Channel 6 frame 5	Status
496	$1F0_H$	Channel 6 frame 5	Data
500	$1F4_H$	Channel 6 frame 5	Time Stamp
504	$1F8_H$	Channel 7 frame 0	Status

Table 430 Frame Mapping (cont'd)

Offset (Dec)	Offset (Hex)	Channel/Frame	Content
508	$1FC_H$	Channel 7 frame 0	Data
512	200_H	Channel 7 frame 0	Time Stamp
516	204_H	Channel 7 frame 1	Status
520	208_H	Channel 7 frame 1	Data
524	$20C_H$	Channel 7 frame 1	Time Stamp
528	210_H	Channel 7 frame 2	Status
532	214_H	Channel 7 frame 2	Data
536	218_H	Channel 7 frame 2	Time Stamp
540	$21C_H$	Channel 7 frame 3	Status
544	220_H	Channel 7 frame 3	Data
548	224_H	Channel 7 frame 3	Time Stamp
552	228_H	Channel 7 frame 4	Status
556	$22C_H$	Channel 7 frame 4	Data
560	230_H	Channel 7 frame 4	Time Stamp
564	234_H	Channel 7 frame 5	Status
568	238_H	Channel 7 frame 5	Data
572	$23C_H$	Channel 7 frame 5	Time Stamp

43.3.8.3 Interrupts for DMA support

Interrupts for Single DMA usage

Interrupts RDI or RSI can be used as trigger for the DMA transfer from TSM/RDS/RDR.

Interrupts for two daisy chained DMA usage

If one of the buffers is full, this is signalled to SW by interrupt CHCI. The "status buffer full" is defined as follows: current number of received frames counted in FCNT.FCx equals the number of expected frames configured in NFC.NFx. In synchronous mode this is the case, if all expected PSI5 frames after a Sync Pulse are received by PSI5-S. The number of expected frames is looked up in register NFC.NFx for channel x. In Asynchronous mode, NFC.NFx should be programmed to 6 while RCRAx.FIDS is set. This way the FID for an asynchronous channel is a rolling number 0 … 5 and the 6 buffer locations are used completely before CHCI is issued. See **Figure 680**.

There is a delay between CHCI and the completion of the DMA transfer of the last set of RDS, RDR and TSM to the system memory! SW needs special checks to grant data consistency, e.g. evaluating the DMA interrupts or comparison of the Time Stamps in the System Memory!

CHCI is not granting that all data is completely moved to system memory by DMA!

Improved Data Consistency with Packet Frame Count (PFC)

For data consistency, RDR, RDS and TSM are tagged with a Packet Frame count value. For each channel there is a separate internal packet counter. This allows to read a triplet of RDR, RDS and TSM from the system memory without looking at interrupts. If the PFC is identical all three values belong together. Otherwise they need to be read again until PFC is identical.

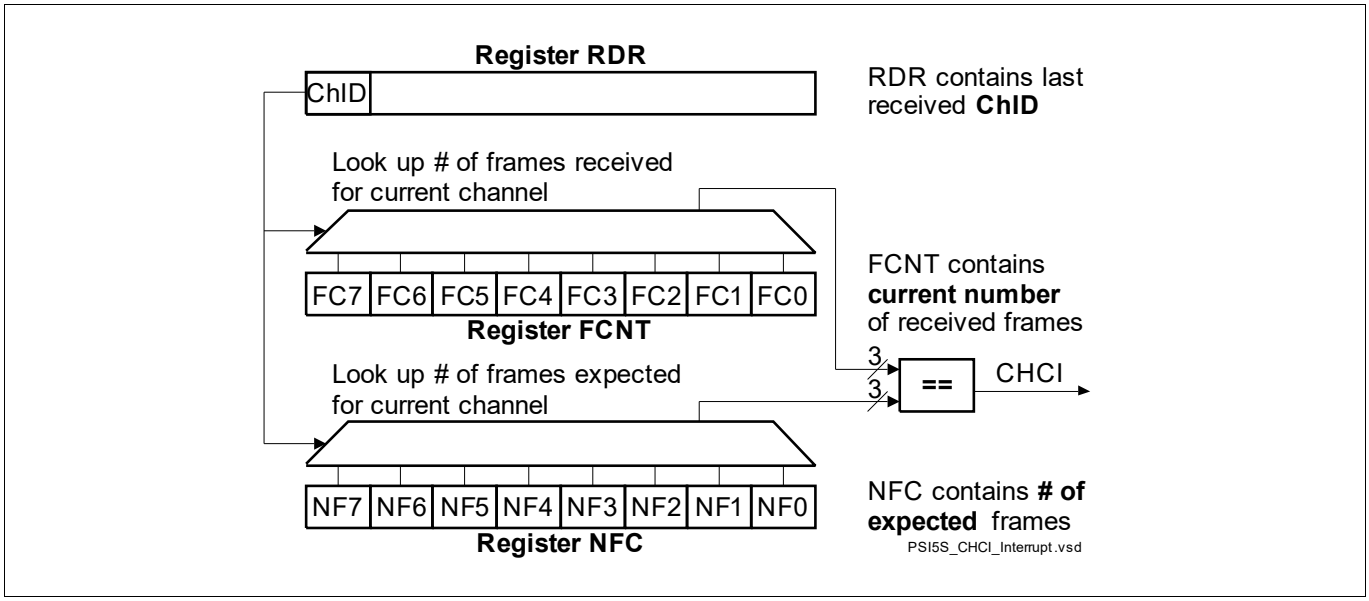


Figure 680 PSI5-S Generation of Interrupt CHCI

43.3.9 Error Detection Capabilities

Each PSI5-S channel can detect and signal the following error conditions:

Protocol Level:

- Packet Frame CRC Error (XCRC)
- PSI5 Frame Checksum error (CRCI)
- Frame not sent in time / UART Frames missing (TEI)
- Error Bits Set in Packet Frame (HDI)
- Errors Signalled by ASC Sub Module (i.e. PE, FE, OE)

Transfer Management Level:

- Receive Data Buffer Overrun (RBI)
- ECU to Sensor Data Buffer Underrun (TBI)

43.3.10 Special use of Channel 0

Note that data that is received in incomplete or too long Packet Frames is always copied - as is - to Channel 0 Frame ID 1. This can happen, if UART Frames are lost, the PHY sends non allowed data or idle before the actual end of the Packet Frame. See **““Packet Frames” received from PHY” on Page 5**.

This is the only actual reservation / limitation in functionality of Channel 0. I.e. all frames can be received and sorted correctly in Channel 0 as long as the Packet Frame is configured with: ChID = 0. It is recommended not to use ChID = 0 AND FID = 1 for normal sensor frames but to keep FID = {2, 3 ,4 ,5}.

The target application assumes, that the PHY sends messages for its own house keeping with ChID = 0, FID = 0. This is assumed in this document and e.g. shown in **Table 430 “Frame Mapping” on Page 25**. Nevertheless, a different application may not have this functionality or may use a different ChID / FID configuration.

43.3.11 ASC Kernel Description

Figure 681 shows a global view of the ASC interface. The ASC is fully integrated into the PSI5-S module. It can be switched to dedicated internal use of the PSI5-S module or to ASC only mode where all PSI5 functionality is disabled. In the last case, the whole PSI5-S module represents one additional ASC in the system.

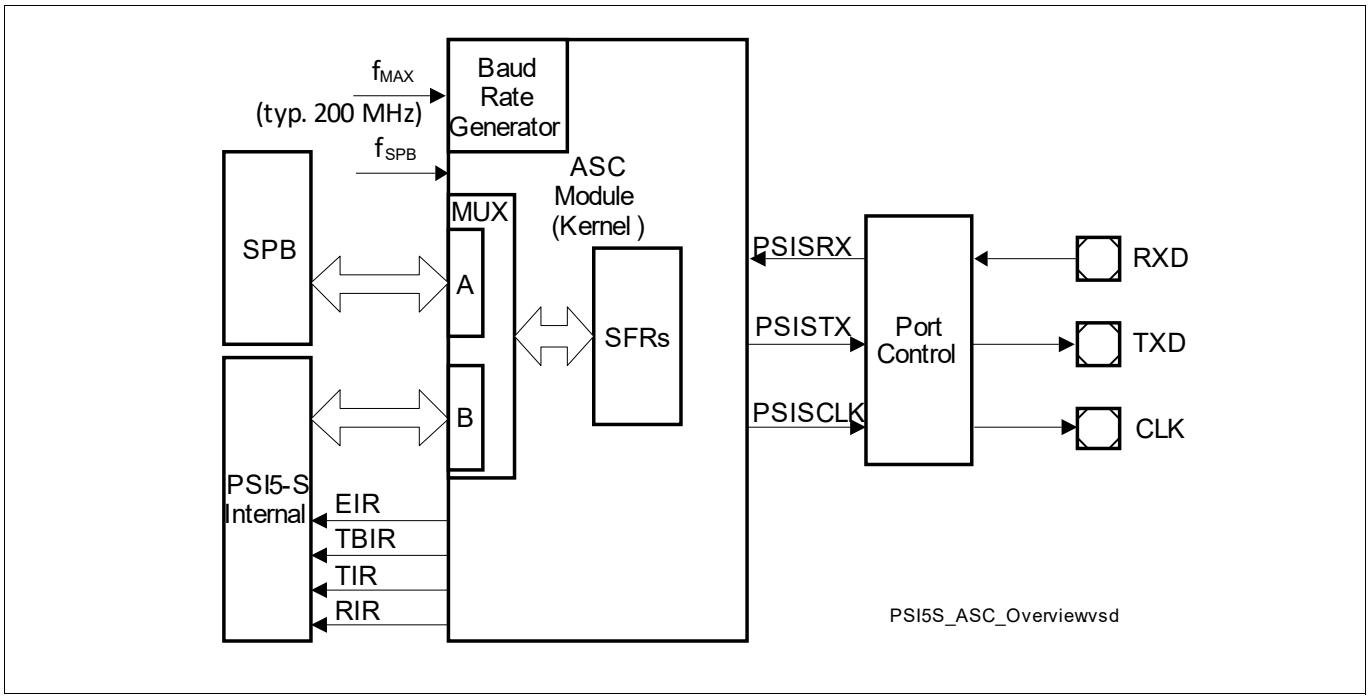


Figure 681 General Block Diagram of the ASC Interface

The ASC module communicates with the external world via two I/O lines. The RXD line is the receive data input signal (and also output signal in Synchronous Mode), and TXD is the transmit output signal.

Clock control, address decoding, and interrupt service request control are managed outside the ASC module kernel.

43.3.11.1 Overview

The ASC provides serial communication between this device and other micro controllers, microprocessors, or external peripherals.

The ASC supports full-duplex asynchronous communication and half-duplex synchronous communication. In Synchronous Mode, data is transmitted or received synchronous to a shift clock that is generated by the ASC internally. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation, and the number of stop bits can be selected. Parity, framing, and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data is double-buffered in ASC only mode. During internal PSI5-S mode, double-buffering is switched off for the transmitter. The receiver stays double buffered. For multiprocessor communication, a mechanism is included to distinguish address Bytes from data Bytes. Testing is supported by a loop-back option. A 13-bit baud rate generator provides the ASC with a separate serial clock signal, which can be accurately adjusted by a prescaler implemented as fractional divider.

Peripheral Sensor Interface with Serial PHY Connection (PSI5-S)

Features

- Full-duplex asynchronous operating modes
  - 8-bit or 9-bit data frames, LSB first
  - Parity-bit generation/checking
  - Separate Parity control for receive and transmit
  - One or two stop bits
  - Baud rate from 12,5 Mbit/s to 3 bit/s (@ 200 MHz module clock)
  - Multiprocessor mode for automatic address/data Byte detection
  - Loop-back capability
- Half-duplex 8-bit synchronous operating mode
  - Baud rate from 25 Mbit/s to 2034,5 bit/s (@ 200 MHz module clock)
- Double-buffered transmitter (in ASC-only mode)/receiver
- Interrupt generation
  - On a transmit buffer empty condition
  - On a transmit last bit of a frame condition
  - On a receive buffer full condition
  - On an error condition (frame, parity, overrun error)
- Implementation features
  - Connections to DMA Controller
  - Connections from GTM

43.3.11.2 General Operation

The ASC supports full-duplex asynchronous communication and half-duplex synchronous communication. In Synchronous Mode, data is transmitted or received synchronously to a shift clock generated by the microcontroller. In Asynchronous Mode, 8-bit or 9-bit data transfer, parity generation, and the number of stop bits can be selected. Parity, framing, and overrun error detection are provided to increase the reliability of data transfers. Transmission and reception of data are double-buffered. For multiprocessor communication, a mechanism is included to distinguish address Bytes from data Bytes. Testing is supported by a loop-back option. A 13-bit baud rate generator provides the ASC with a separate serial clock signal, which can be accurately adjusted by a prescaler implemented as fractional divider.

A transmission is started by writing to the Transmit Buffer Register, TBUF. Only the number of data bits determined by the selected operating mode will actually be transmitted; that is, bits written to positions 9 through 15 of register TBUF are always insignificant. Data transmission is double-buffered, so a new character may be written to TBUF before the transmission of the previous character is complete. This allows a back-to-back transmission of characters to take place without gaps.

Data reception is enabled by the receiver enable bit CON.REN. After a reception has been completed, the received data and, if provided by the selected operating mode, the received parity bit can be read from the (read-only) receive buffer register RBUF. Unused bits in the upper half of RBUF that are not required in the selected operating mode will be read as zeros.

Data reception is double-buffered, so that reception of a second character may already begin before the previously received character has been read out of the receive buffer register. During the exclusive use by the PSI5-S, double buffering is switched off. The PSI5-S internal output FIFO is used instead. As commands for sync pulses are time stamped inside PSI5-S as they leave the internal FIFO, the double buffering would add unwanted jitter. In all modes, receive buffer overrun error detection can be selected through bit CON.OEN. When enabled, the overrun error status flag CON.OE and the error interrupt request line EIR will be activated when the receive buffer register has not been read by the time reception of a second character is complete. In this case, the previously received character in the receive buffer is overwritten.

The loop-back option (selected by bit CON.LB) allows the data currently being transmitted to be received simultaneously in the receive buffer. This may be used to test serial communication routines at an early stage without having to provide an external network. In loop-back mode, the alternate input/output function of port pins is not required.

43.3.11.3 Asynchronous Operation

Asynchronous Mode supports full-duplex communication, in which both transmitter and receiver use the same data frame format and have the same baud rate. Data is transmitted on pin TXD and received on pin RXD. **Figure 682** shows the block diagram of the ASC when operating in Asynchronous Mode.

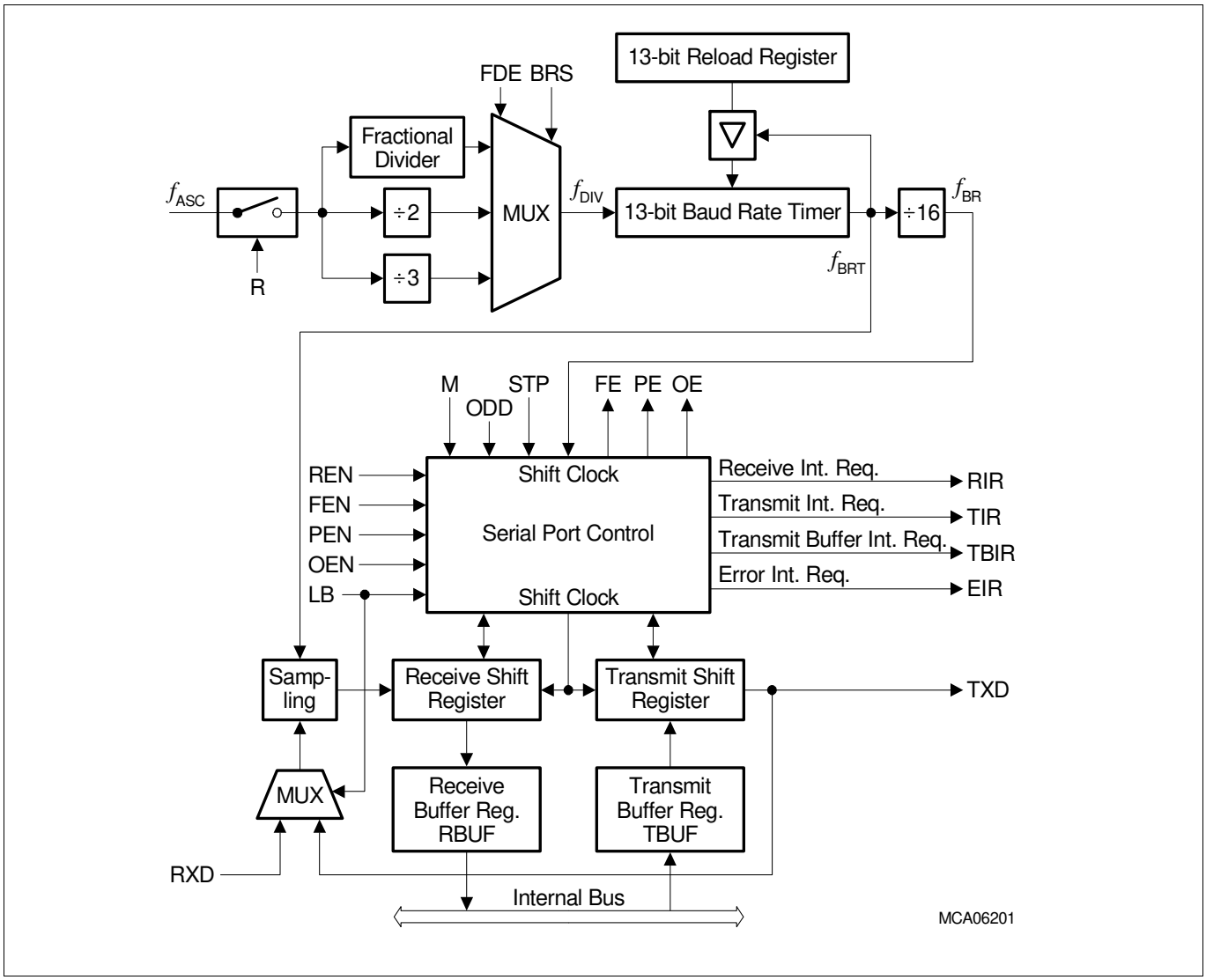


Figure 682 Asynchronous Mode of the ASC

43.3.11.3.1 Asynchronous Data Frames

Asynchronous data frames can consist of 8-bit or 9-bit data frames.

8-bit Data Frames

The 8-bit data frames consist of either eight data bits D7 … D0 (CON.M = 001_B), or of seven data bits D6 … D0 plus an automatically generated parity bit (CON.M = 011_B). Parity may be odd or even, depending on bit CON.ODD. An even parity bit will be set if the modulo-2 sum of the seven data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit CON.PEN (always OFF in 8-bit data mode). The parity error flag CON.PE will be set, along with the error interrupt request flag, if a wrong parity bit is received. The received parity bit itself will be stored in RBUF too.

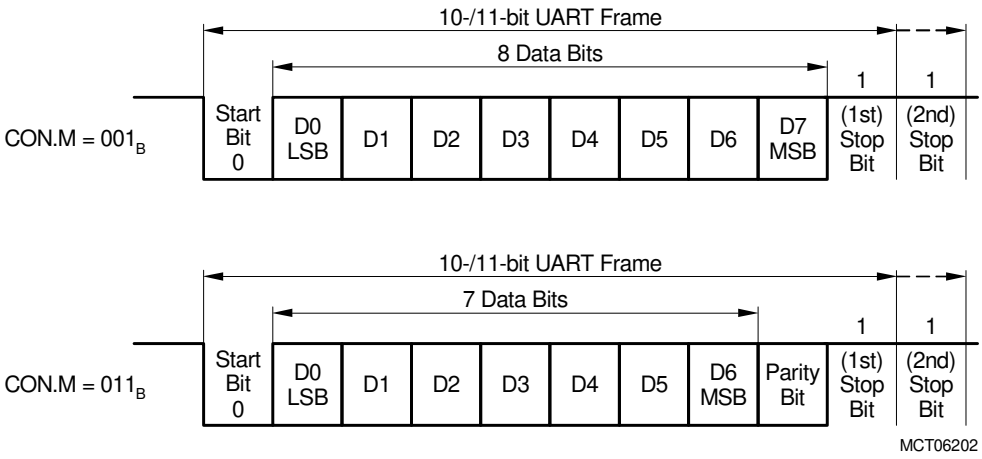


Figure 683 Asynchronous 8-bit Frames

9-bit Data Frames

The 9-bit data frames consist of nine data bits D8 … D0 (CON.M = 100_B), or of eight data bits D7 … D0 plus an automatically generated parity bit (CON.M = 111_B) or of eight data bits D7 … D0 plus wake-up bit (CON.M = 101_B). Parity may be odd or even, depending on bit CON.ODD. An even parity bit will be set if the modulo-2-sum of the eight data bits is 1. An odd parity bit will be cleared in this case. Parity checking is enabled via bit CON.PEN (always OFF in 9-bit data and wake-up mode). The parity error flag CON.PE will be set along with the error interrupt request flag if a wrong parity bit is received. The received parity bit itself will be stored in RBUF too.

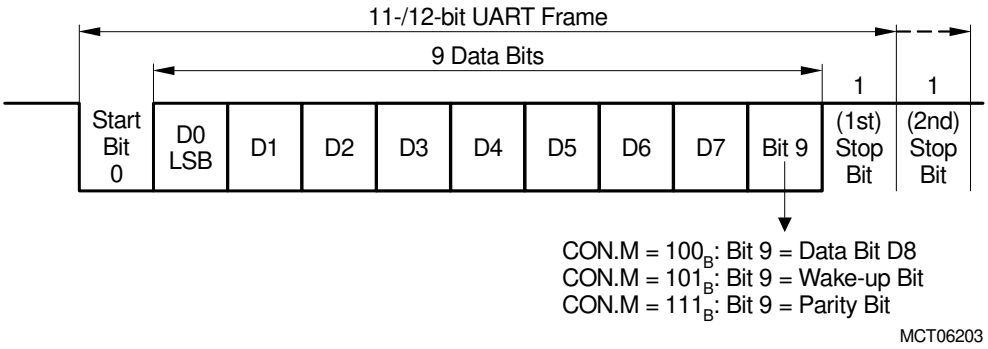


Figure 684 Asynchronous 9-bit Frames

In Wake-up Mode (CON.M = 101_B), received frames are transferred to the receive buffer register only if the 9^{th} bit (the wake-up bit) of the frame is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.

This feature can be used to control communication in multi-processor systems, for example:
When the master processor aims to transmit a block of data to one of several slaves, it first sends out an address ‘Byte’ (in this case, a ‘Byte’ consists of nine bits) that identifies the target slave. An address ‘Byte’ differs from a data ‘Byte’ in that the additional 9^{th} bit is a 1 for an address ‘Byte’ but is a 0 for a data ‘Byte’, so, no slave will be interrupted by a data ‘Byte’. An address ‘Byte’ will interrupt all slaves (operating in 8-bit data + wake-up bit mode), so each slave can examine the eight LSBs of the received character (the address). The addressed slave will switch to 9-bit data mode (for example, by clearing bit CON.M[0]), which enables it to also receive the data Bytes

that will be coming (having the wake-up bit cleared). The slaves that were not being addressed remain in 8-bit data + wake-up bit mode, ignoring the following data ‘Bytes’.

43.3.11.3.2 Asynchronous Transmission

Asynchronous transmission begins when the next overflow of the divide-by-16 baud rate timer (transition of the baud rate clock f_{BR}) occurs, if bit CON.R is set and data has been loaded into TBUF. The transmitted data frame consists of three elements:

1. The start bit
2. The data field (8 or 9 bits, LSB first, including a parity bit, if selected)
3. The delimiter (1 or 2 stop bits)

Data transmission is double-buffered. When the transmitter is idle, the transmit data loaded into TBUF is immediately moved to the transmit shift register; thus, freeing TBUF for the next transmit data to be loaded. This is indicated by the transmit buffer interrupt request line TBIR being activated. TBUF may then be loaded with the next transmit data while transmission of the previous one continues.

The Transmit Interrupt Request line TIR will be activated before the last bit of a frame is transmitted, that is, before the first or the second stop bit is shifted out of the transmit shift register.

Note: A dedicated GPIO device pin which is connected to the module output pin TXD must be configured by software as alternate data output for asynchronous transmission.

43.3.11.3.3 Asynchronous Reception

Asynchronous reception is initiated by a falling edge (1-to-0 transition) on pin RXD, on the condition that bits CON.R and CON.REN are set. The receive data input pin RXD is sampled at sixteen times the rate of the selected baud rate. A majority decision of the 7^{th}, 8^{th} and 9^{th} sample determines the effective sampled bit value. This avoids erroneous results that may be caused by noise.

If the detected value is not a 0 when the start bit is sampled, the receive circuit is reset and waits for the next 1-to-0 transition at pin RXD. If the start bit proves valid, the receive circuit continues sampling and shifts the incoming data frame into the receive shift register.

When the last stop bit has been received, the contents of the receive shift register are transferred to the Receive Data Buffer Register RBUF. Simultaneously, the receive interrupt request line RIR is activated after the 9^{th} sample in the last stop bit time-slot (as programmed), regardless whether valid stop bits have been received or not. The receive circuit then waits for the next start bit (1-to-0 transition) at the receive data input line.

Note: A dedicated GPIO pin that is connected to the module input pin RXD must be configured by software as input for asynchronous reception.

Asynchronous reception is stopped by clearing bit CON.REN. A currently received frame is completed including generation of the receive interrupt request and an error interrupt request, if appropriate. Start bits that follow this frame will not be recognized.

Note: In wake-up mode, received frames are transferred to the receive buffer register only if the 9^{th} bit (the wake-up bit) is 1. If this bit is 0, no receive interrupt request will be activated and no data will be transferred.

43.3.11.3.4 RXD/TXD Data Path Selection in Asynchronous Modes

The data paths for the serial input and output data in Asynchronous Modes are affected by control bit CON.LB (loop-back) as shown in **Figure 685**.

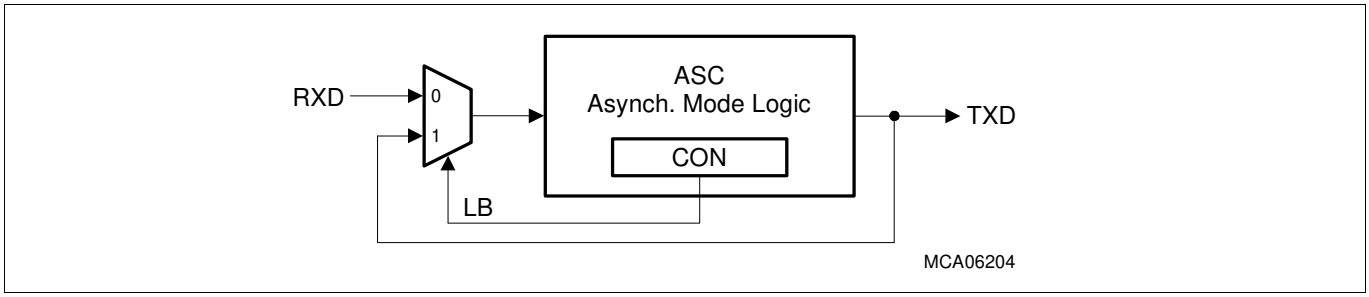


Figure 685 RXD/TXD Data Path Selection in Asynchronous Modes

43.3.11.4 Synchronous Operation

Synchronous Mode supports half-duplex communication, usable for simple I/O expansion via shift registers. Data is transmitted and received via pin RXD while pin TXD outputs the shift clock. These signals are typically connected as alternate functions with GPIO port pins. Synchronous Mode is selected with CON.M = 000_B.

Eight data bits are transmitted or received synchronous to a shift clock generated by the internal baud rate generator. The shift clock is active only as long as data bits are transmitted or received.

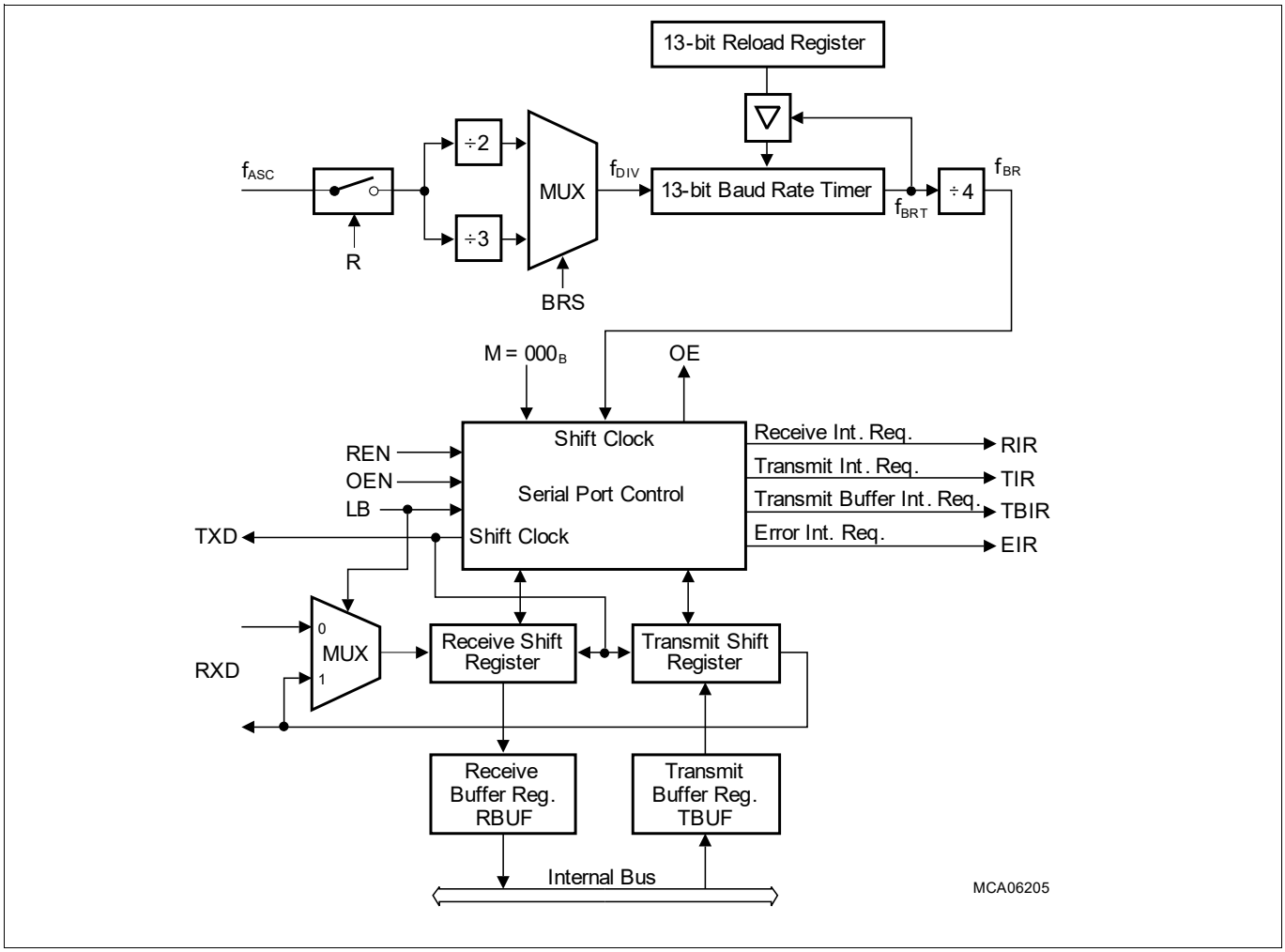


Figure 686 Synchronous Mode of Serial Channel ASC

43.3.11.4.1 Synchronous Transmission

Synchronous transmission begins after data has been loaded into TBUF, provided that CON.R is set and CON.REN = 0 (half-duplex, no reception), with one exception: in asynchronous Loop-back Mode (bit CON.LB set), CON.REN must be set for reception of the transmitted Byte. In synchronous Loop-back Mode, the transmitted Byte can still be received if CON.REN=0. Data transmission is double-buffered. When the transmitter is idle, the transmit data loaded into TBUF is immediately moved to the transmit shift register, thus freeing TBUF for the next data to be sent. This is indicated by the transmit buffer interrupt request line TBIR being activated. TBUF may now be loaded with the next data, while transmission of the previous one continues. The data bits are transmitted synchronously with the shift clock. After the bit time for the 8^{th} data bit, both TXD and RXD will be set to high level, the transmit interrupt request line TIR is activated, and serial data transmission stops.

Note: The dedicated GPIO device pins that are connected to TXD and RXD must be configured by software as alternate data outputs in order to provide the shift clock and the output data during synchronous transmission.

43.3.11.4.2 Synchronous Reception

Synchronous reception is initiated by setting bit CON.REN = 1. If bit CON.R = 1, the data applied at RXD is clocked into the receive shift register synchronously to the clock which is output at TXD. After the 8^{th} bit has been shifted in, the contents of the receive shift register are transferred to the receive data buffer RBUF, the receive interrupt request line RIR is activated, the receiver enable bit CON.REN is reset, and serial data reception stops.

Synchronous reception is stopped by clearing bit CON.REN. Any Byte that is currently being received is completed, including the generation of the receive interrupt request and an error interrupt request, if appropriate. Writing to the transmit buffer register while a reception is in progress has no effect on reception and will not start a transmission.

If a previously received Byte has not been read out of the receive buffer register by the time the reception of the next Byte is complete, both the error interrupt request line EIR and the overrun error status flag CON.OE will be activated/set, provided that the overrun check has been enabled by bit CON.OEN.

Note: The dedicated GPIO device pin that is connected to TXD must be configured by software as alternate data output in order to provide the shift clock. The dedicated GPIO device pin that is connected to RXD must be configured by software as input during synchronous reception.

43.3.11.4.3 Synchronous Timing

Figure 687 shows timing diagrams of the ASC Synchronous Mode data reception and data transmission. In idle state, the shift clock is at high level. With the beginning of a synchronous transmission of a data Byte, the data is shifted out at RXD with the falling edge of the shift clock. If a data Byte is received through RXD, data is latched with the rising edge of the shift clock.

One shift clock cycle (f_{BR}) delay is inserted between two consecutive receive or transmit data Bytes.

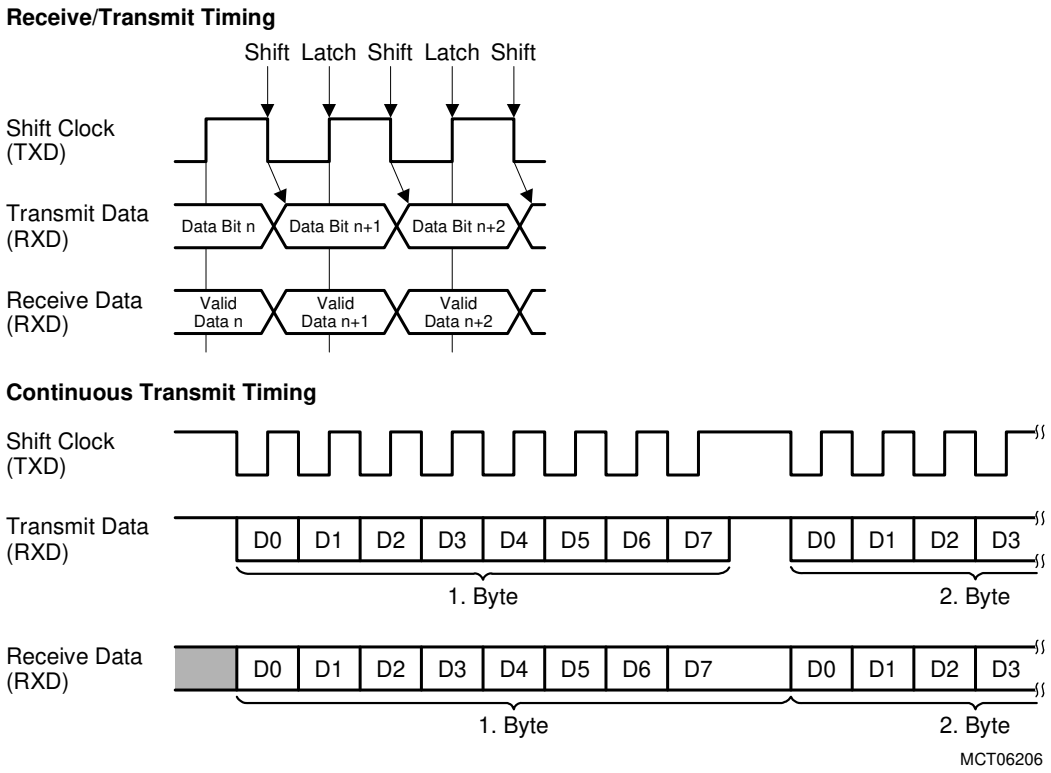


Figure 687 ASC Synchronous Mode Waveforms

43.3.11.5 Baud Rate Generation

The ASC has its own dedicated 13-bit baud rate generator with 13-bit reload capability, allowing baud rate generation independent of other timers.

The baud rate generator is clocked with a clock (f_{DIV}) which is derived via a prescaler from the ASC module clock f_{ASC}. The baud rate timer is counting downwards and can be started or stopped through the baud rate generator run bit CON.R. Each underflow of the timer generates one clock pulse to the serial channel. The timer is reloaded with the value stored in its 13-bit reload register at each underflow. The resulting clock f_{BRT} is again divided by a factor for the baud rate clock (÷ 16 in asynchronous operating modes and ÷ 4 in synchronous operating mode). The prescaler is selected by the bits CON.BRS and CON.FDE. In the asynchronous operating modes, a fractional divider prescaler unit is available (in addition to the two fixed dividers) that allows selection of prescaler divider ratios of n/2048 with n = 0-2047. Therefore, the baud rate of ASC is determined by the module clock, the content of register FDV, the reload value in register BG, and the operating mode (asynchronous or synchronous).

Register BG is the dual-function baud rate generator/reload register. Reading BG returns the contents of the timer in bit field BR_VALUE (bits 31:13 return zero), while writing to BG always updates the reload register (bits 31:13 are insignificant).

An auto-reload of the timer with the contents of the reload register is performed each time BG is written to. However, if CON.R = 0 at the time the write operation to BG is performed, the timer will not be reloaded until the first instruction cycle after CON.R = 1. For a clean baud rate initialization, BG should only be written if CON.R = 0. If BG is written with CON.R = 1, an unpredictable behavior of the ASC may occur during running transmit or receive operations.

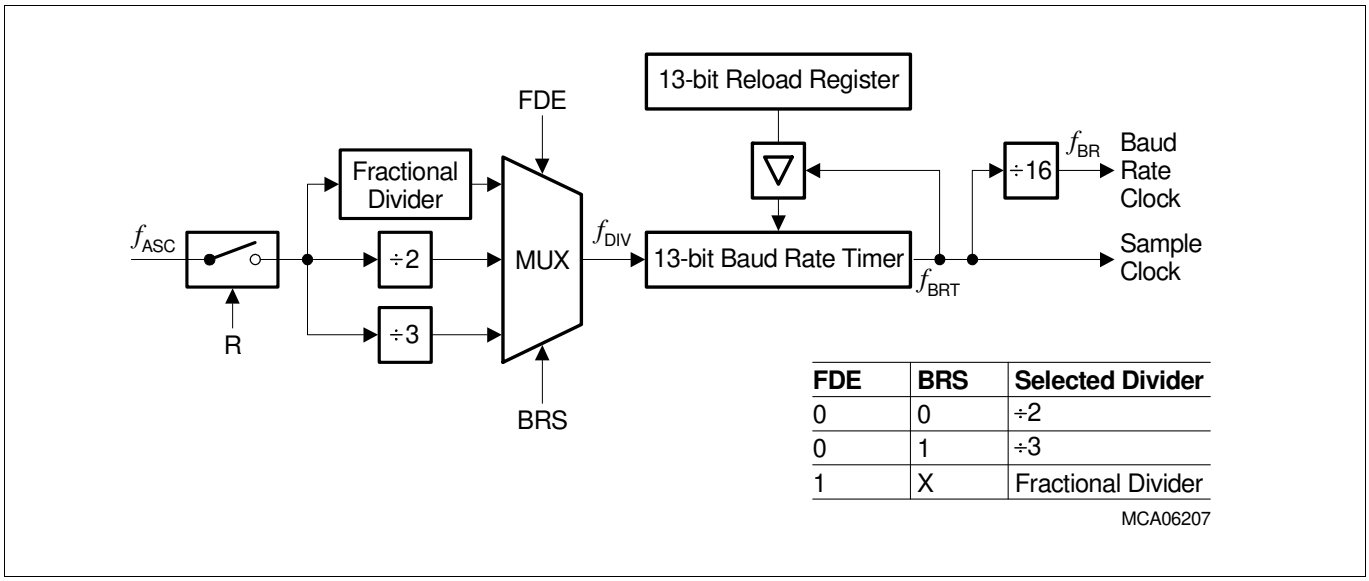
43.3.11.5.1 Baud Rates in Asynchronous Mode

For asynchronous operation, the baud rate generator provides a clock f_{BRT} with sixteen times the rate of the established baud rate. Every received bit is sampled on the 7th, 8th and 9th cycle of this clock. The clock divider circuitry, which generates the input clock f_{DIV} for the 13-bit baud rate timer, is extended by a fractional divider circuitry that allows the adjustment of more accurate baud rates and the extension of the baud rate range.

The baud rate of the baud rate generator depends on the settings of the following bits and register values:

- Input clock f_{ASC}
- Selection of the baud rate timer input clock f_{DIV} by bits CON.FDE and CON.BRS
- If bit CON.FDE = 1 (fractional divider): value of register FDV
- Value of the 13-bit reload register BG

The output clock of the baud rate timer with the reload register is the sample clock in the asynchronous operating modes of the ASC. For baud rate calculations, this baud rate clock f_{BR} is derived from the sample clock f_{BRT} by a division of sixteen.



FDE	BRS	Selected Divider
0	0	÷2
0	1	÷3
1	X	Fractional Divider

Figure 688 ASC Baud Rate Generator Circuitry in Asynchronous Modes

Using the Fixed Input Clock Divider

The baud rate for asynchronous operation of the serial channel ASC when using the fixed input clock divider ratios (CON.FDE = 0) and the required BG reload value for a given baud rate can be determined by the following formulas:

Table 431 Asynchronous Baud Rate Formulas using the Fixed Input Clock Dividers

FDE	BRS	BG	Formula
0	0	0 … 8191	Baud rate = f_{ASC} / (32 x (BG + 1)) BG = f_{ASC} / (32 x Baud rate) - 1
	1		Baud rate = f_{ASC} / (48 x (BG + 1)) BG = f_{ASC} / (48 x Baud rate) - 1

BG represents the content of the reload register bit field BG.BR_VALUE, taken as an unsigned 13-bit integer.

The maximum baud rate that can be achieved for the asynchronous operating modes when using the two fixed clock dividers and a module clock of 200 MHz is 6.25 Mbit/s.

Using the Fractional Divider

When the fractional divider is selected, the input clock f_{DIV} for the baud rate timer is derived from the module clock f_{ASC} by a programmable fractional divider. If CON.FDE = 1, the fractional divider is activated. It divides f_{ASC} by a fraction of n/2048 for any value of n from 0 to 2047. If n = 0, the divider ratio is 1, which means that $f_{DIV} = f_{ASC}$. In general, the fractional divider allows the baud rate to be programmed with much better accuracy than with the two fixed prescaler divider stages.

Note: In fractional divider mode, the clock f_{DIV} can have a maximum period jitter of one f_{ASC} clock period.

Table 432 Asynchronous Baud Rate Formulas using the Fractional Input Clock Divider

FDE	BRS	BG	FDV	Formula
1	–	0 … 8191	1 … 2047	Baud rate = (FDV / 2048) x (f_{ASC} / (16 x (BG + 1)))
			0	Baud rate = f_{ASC} / (16 x (BG + 1))

BG represents the content of the reload register bit field BG.BR_VALUE, taken as an unsigned 13-bit integer. FDV represents the contents of the fractional divider register bit field FDV.FD_VALUE, taken as an unsigned 11-bit integer.

43.3.11.5.2 Baud Rates in Synchronous Mode

For synchronous operation, the baud rate generator provides a clock f_{BRT} that runs with four times the established baud rate (see **Figure 689**).

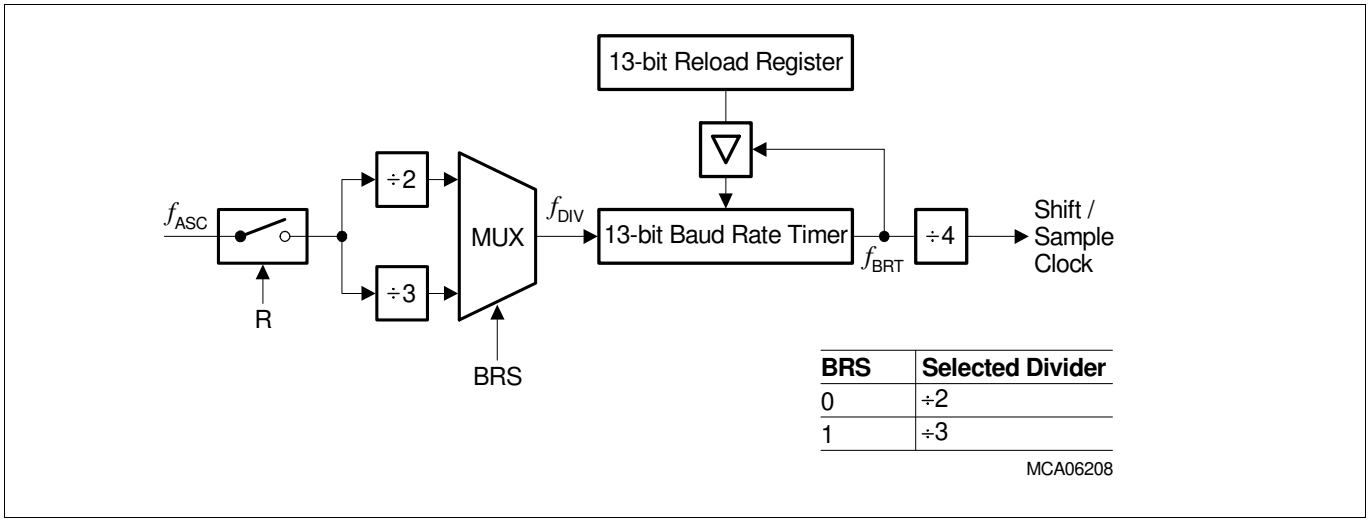


Figure 689 ASC Baud Rate Generator Circuitry in Synchronous Mode

The baud rate for synchronous operation of the serial channel ASC can be determined by the formulas as shown in **Table 433**.

Table 433 Synchronous Baud Rate Formulas

BRS	BG	Formula
0	0 … 8191	Baud rate = f_{ASC} / (8 x (BG + 1)) BG = f_{ASC} / (8 x Baud rate) - 1
1		Baud rate = f_{ASC} / (12 x (BG + 1)) BG = f_{ASC} / (12 x Baud rate) - 1

BG represents the content of the reload register bit field BG.BR_VALUE, taken as an unsigned 13-bit integer.

43.3.11.6 Hardware Error Detection Capabilities

To improve the reliability of serial data exchange, the serial channel ASC provides an error interrupt request flag that indicates the presence of an error and three (selectable) error status flags in register CON that indicate which error has been detected during reception. Upon completion of a reception, the error interrupt request line EIR will be activated simultaneously with the receive interrupt request line RIR, if one or more of the following conditions are met:

- If the framing error detection enable bit CON.FEN is set and any of the expected stop bits is not high, the framing error flag CON.FE is set, indicating that the error interrupt request is due to a framing error (asynchronous operating modes only).
- If the parity error detection enable bit CON.PEN is set in the modes where a parity bit is received and the parity check on the received data bits proves false, the parity error flag CON.PE is set, indicating that the error interrupt request is due to a parity error (asynchronous operating modes only).
- If the overrun error detection enable bit CON.OEN is set and the last character received was not read out of the receive buffer by software or DMA transfer at the time the reception of a new frame is complete, the overrun error flag CON.OE is set indicating that the error interrupt request is due to an overrun error (Asynchronous and Synchronous Modes).

43.3.11.7 Interrupts

Four interrupt sources are provided for serial channel ASC. Line TIR indicates a transmit interrupt, TBIR indicates a transmit buffer interrupt, RIR indicates a receive interrupt, and EIR indicates an error interrupt of the serial channel. The interrupt output lines TBIR, TIR, RIR, and EIR are synchronized to the PSI5-S kernel and routed by the INPs to the trigger output lines.

The cause of an error interrupt request EIR (framing, parity, overrun error) can be identified by the error status flags CON.FE, CON.PE, and CON.OE.

Note: By contrast to the error interrupt request line EIR, the error status flags CON.FE/CON.PE/CON.OE are not reset automatically but must be cleared by software.

For normal operation (that is, other than error interrupt), the ASC provides three interrupt requests to control data exchange via this serial channel:

- TBIR is activated when data is moved from TBUF to the transmit shift register.
- TIR is activated before the last bit of an asynchronous frame is transmitted, or after the last bit of a synchronous frame has been transmitted.
- RIR is activated when the received frame is moved to RBUF.

While the task of the receive interrupt handler is quite clear, the transmitter is serviced by two interrupt handlers. This provides advantages for the servicing software.

For single transfers, it is sufficient to use the transmitter interrupt (TIR), which indicates that the previously loaded data has been transmitted, except for the last bit of an asynchronous frame.

For multiple back-to-back transfers, it is necessary to load the following piece of data at least before the last bit of the previous frame has been transmitted. In Asynchronous Mode, this leaves just one bit-time for the handler to respond to the transmitter interrupt request; in Synchronous Mode, it is entirely impossible.

Using the Transmit Buffer Interrupt (TBIR) to reload transmit data provides the time necessary to transmit a complete frame for the service routine, as TBUF may be reloaded while the previous data is still being transmitted.

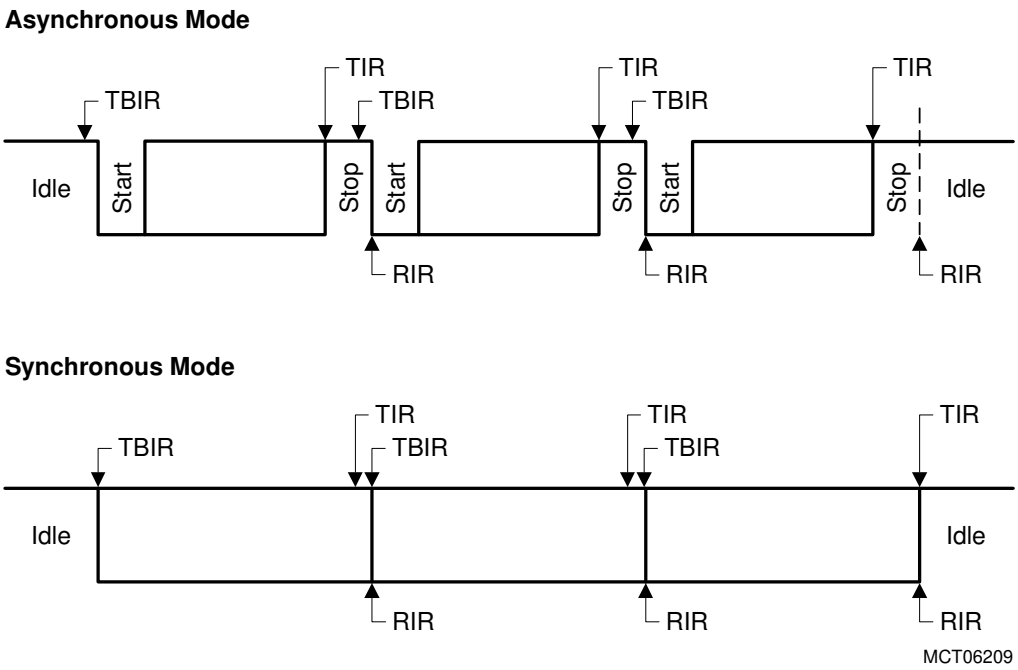


Figure 690 ASC Interrupt Generation

As shown in **Figure 690**, TBIR is an early trigger for the reload routine, while TIR indicates the completed transmission. Software using handshake should, therefore, rely on TIR at the end of a data block to ensure that all data has been transmitted.

43.3.12 Interrupts

8 Interrupt sources are available for the PSI5-S module. For each trigger source of a channel x one interrupt can be selected in register INPx.

RDI indicates a receive data interrupt. It is activated when a received frame is moved to Receive Data Register RDR.

RSI indicates a receive frame success interrupt, i.e. non of the interrupts/errors XCRCI, CRCI, TEI, PE, FE, OE, RBI or HDI was detected. The referring interrupt must be enabled in GCR to be considered for RSI. Both RDI and RSI will be issued together in normal use cases where reception is correct.

RBI indicates a receive buffer overrun interrupt. It is activated when a new frame is transferred to a Receive Data Register RDR while the old value was still not read by the host (“overwrite”), i.e. the kernel wants to set interrupt RDI and finds RDI already set.

TPI indicates a transmit interrupt. It is activated when data is completely moved from a SDR to ASC (TPI). Thus it can be used to trigger the next write access to SDR.

TPOI indicates a transfer (send) register (SDR) overrun interrupt. It is set if data is written to SDR while the referring register is locked (occupied), signalled by TPF. The data written is ignored in this case. FOI indicates that the output FIFO was overrun.

In addition the protocol error interrupts are available:

CRCI, XCRC. If CRC interrupt is activated, data is to be treated as invalid according to standard V2.0 2011-06. Note that the wrong data is still available in all buffers. For XCRCI, these frames are always stored in Channel 0 Frame 1.

TEI is issued by the watch dog timer if the timely distance between two frames is too long.

CHCI indicates that the last frame expected for a channel is received.

The interrupt request or the corresponding interrupt set bit (in register INTSET) can trigger the interrupt generation at the selected interrupt node. The service request pulse is generated independently from the interrupt flag in register INTSTATx. The interrupt flag can be cleared by software by writing to the corresponding bit in register INTCLR.

If more than one interrupt source is connected to the same interrupt node pointer (in register INPx all node pointer point to the same line), the requests are combined to one common line.

43.3.13 Trigger Outputs

Any interrupt source can be used as trigger output TRIGO outside the module. Each TRIGO is connected to the interrupt router.

43.4 Registers

This section describes the kernel registers of the PSI5-S module. All PSI5-S kernel register names described in this section will be referenced in other parts of this User's Manual by the module name prefix "PSI5S_" for the PSI5-S interface.

All registers in the PSI5-S address spaces are reset with the application reset with one exception: OCS is reset with debug reset only. (Definition see SCU section "Reset Operation").

The complete and detailed address map of the PSI5-S module is described below.

x can take the values 0 … 7 (PSI5-S channels)

List of Access Protection Abbreviations

U - User Mode

SV - Supervisor Mode

BE - Bus Error

nBE - no Bus Error

P - Access Protection, as defined by the ACCEN Register

E - ENDINIT

SE - Safety ENDINIT

Table 434 Register Overview - PSI5S (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	Clock Control Register	0000_H	SV,U	SV,E,P	Application Reset	52
ID	Module Identification Register	0008_H	SV,U	BE	Application Reset	52
FDR	PSI5-S Fractional Divider Register	$000C_H$	SV,U	U,SV,P	Application Reset	53
FDRT	Fractional Divider Register for Time Stamp	0010_H	SV,U	U,SV,P	Application Reset	54
TSCNTA	Time Stamp Count Register A	0014_H	SV,U	U,SV,P	Application Reset	55
TSCNTB	Time Stamp Count Register B	0018_H	SV,U	U,SV,P	Application Reset	56

Table 434 Register Overview - PSI5S (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
GCR	Global Control Register	$001C_H$	SV,U	U,SV,P	Application Reset	57
NFC	Number of Frames Control Register	0020_H	SV,U	U,SV,P	Application Reset	58
FCNT	Frame Counter Register	0024_H	SV,U	U,SV,P	Application Reset	59
IOCR	Input and Output Control Register	0028_H	SV,U	U,SV,P	Application Reset	60
RCRAx	Receiver Control Register Ax	0030_H+x *4	SV,U	U,SV,P	Application Reset	60
RCRBx	Receiver Control Register Bx	0050_H+x *4	SV,U	U,SV,P	Application Reset	62
WDTx	Watch Dog Timer Register x	0070_H+x *4	SV,U	U,SV,P	Application Reset	63
TSCRx	Capture Register TSCRx	0090_H+x *4	SV,U	BE	Application Reset	63
RDS	Receive Status Register	$00B0_H$	SV,U	BE	Application Reset	64
RDR	Receive Data Register	$00B4_H$	SV,U	BE	Application Reset	66
TSM	Time Stamp Mirror Register	$00B8_H$	SV,U	BE	Application Reset	66
TAR	Target Address Register	$00D0_H$	SV,U	BE	Application Reset	67
BAR	Base Address Register	$00D4_H$	SV,U	U,SV,P	Application Reset	68
PGCx	Pulse Generation Control Register x	$00F0_H$+x *4	SV,U	U,SV,P	Application Reset	68
CTVx	Channel Trigger Value Register x	0110_H+x *4	SV,U	U,SV,P	Application Reset	69
SCRx	Send Control Register x	0130_H+x *4	SV,U	U,SV,P	Application Reset	70
SDRx	Send Data Register x	0150_H+x *4	SV,U	U,SV,P	Application Reset	72
CDW	CPU Direct Write Register	0170_H	SV,U	U,SV,P	Application Reset	72
CON	Control Register	0210_H	SV,U	U,SV,P	Application Reset	73
BG	Baud Rate Timer/Reload Register	0214_H	SV,U	U,SV,P	Application Reset	75

Table 434 Register Overview - PSI5S (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
FDV	Fractional Divider Register	0218_H	SV,U	U,SV,P	Application Reset	76
FDO	Fractional Divider for Output CLK Register	$021C_H$	SV,U	U,SV,P	Application Reset	76
TBUF	Transmit Buffer Register	0220_H	SV,U	U,SV,P	Application Reset	77
RBUF	Receive Buffer Register	0224_H	SV,U	BE	Application Reset	78
WHBCON	Write Hardware Bits Control Register	0250_H	SV,U	U,SV,P	Application Reset	78
INTSTATx	Interrupt Status Register x	0260_H+x *4	SV,U	BE	Application Reset	80
INTSETx	Interrupt Set Register x	0280_H+x *4	SV,U	U,SV,P	Application Reset	82
INTCLRx	Interrupt Clear Register x	$02A0_H$+x *4	SV,U	U,SV,P	Application Reset	83
INTENx	Interrupt Enable Register x	$02C0_H$+x *4	SV,U	U,SV,P	Application Reset	84
INPx	Interrupt Node Pointer Register x	$02E0_H$+x *4	SV,U	U,SV,P	Application Reset	85
INTOV	Interrupt Overview Register	0300_H	SV,U	BE	Application Reset	87
INTSTATG	Interrupt Status Register Global	0304_H	SV,U	BE	Application Reset	88
INTSETG	Interrupt Set Register Global	0308_H	SV,U	U,SV,P	Application Reset	89
INTCLRG	Interrupt Clear Register Global	$030C_H$	SV,U	U,SV,P	Application Reset	90
INTENG	Interrupt Enable Register Global	0310_H	SV,U	U,SV,P	Application Reset	91
INPG	Interrupt Node Pointer Register Global	0314_H	SV,U	U,SV,P	Application Reset	92
OCS	OCDS Control and Status	$03CC_H$	U,SV	SV,P	Debug Reset	93
ACCEN0	Access Enable Register 0	$03D0_H$	U,SV	SV,SE	Application Reset	94
ACCEN1	Access Enable Register 1	$03D4_H$	U,SV	SV,SE	Application Reset	95
KRST0	Kernel Reset Register 0	$03D8_H$	U,SV	SV,E,P	Application Reset	95

Table 434 Register Overview - PSI5S (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
KRST1	Kernel Reset Register 1	$03DC_H$	U,SV	SV,E,P	Application Reset	96
KRSTCLR	Kernel Reset Status Clear Register	$03E0_H$	U,SV	SV,E,P	Application Reset	97

43.4.1 Module Control

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock, sleep mode and disable mode for the module.

Note: After a hardware reset operation, the clocks $f_{PSI5\text{-}S}$ and the ASC clock f_{ASC} are switched off and the PSI5-S module and the included ASC module are disabled (DISS set).

CLC
Clock Control Register (0000_H) Application Reset Value: $0000\ 0003_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS	0	DISS	DISR
r												rw	r	rh	rw

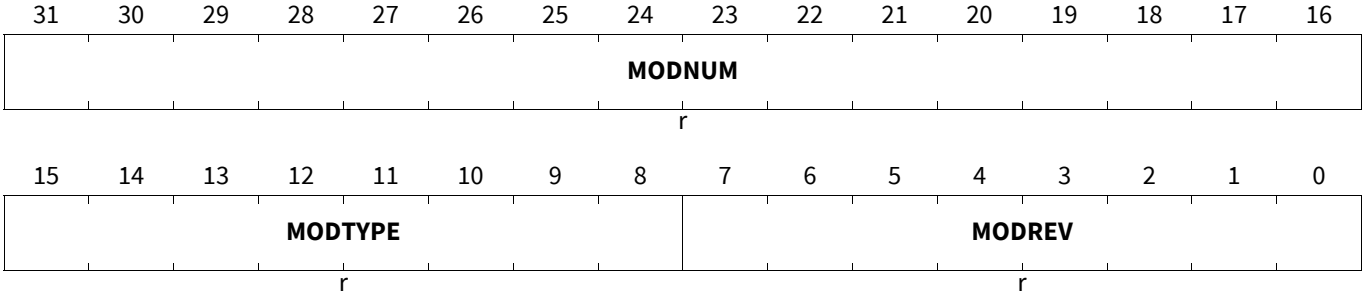
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. *Note: This bit disables the kernel clocks $f_{PSI5\text{-}S}$ and the ASC clock f_{ASC}.*
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
EDIS	3	rw	**External Sleep Mode Request Disable Bit** Used to control module's sleep mode. *Note: If this bit is cleared the kernel clocks $f_{PSI5\text{-}S}$ and the ASC clock f_{ASC} are disabled during System Sleep Mode.*
0	2, 31:4	r	**Reserved** Read as 0; should be written with 0.

Module Identification Register

The PSI5-S Module Identification Register ID contains read-only information about the module version.

ID

Module Identification Register **(0008$_H$)** **Application Reset Value: 00D3 C0XX$_H$**



Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01_H (first revision).
MODTYPE	15:8	r	**Module Type** This bit field defines the module as a 32-bit module: $C0_H$
MODNUM	31:16	r	**Module Number Value** This bit field defines the module identification number for the PSI5-S: $00D3_H$

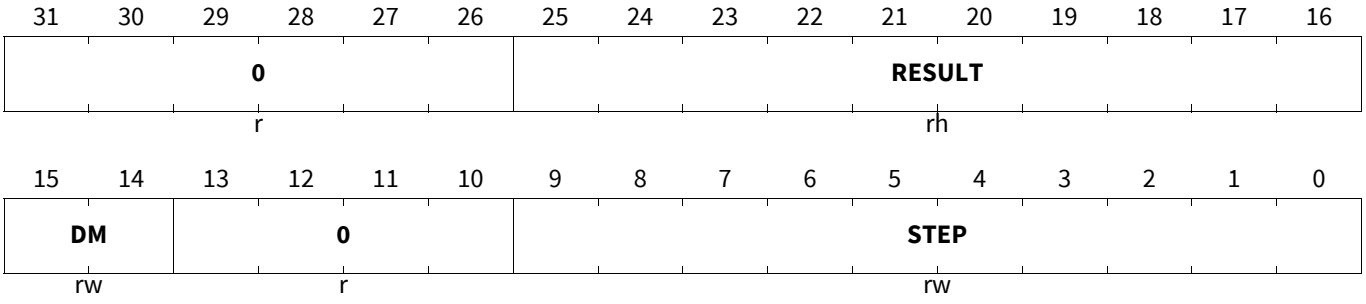
PSI5-S Fractional Divider Register

The Fractional Divider Register controls the clock $f_{PS5\text{-}S}$.

This bit field only controls the kernel clock $f_{PSI5\text{-}S}$ and not the sampling clock f_{ASC}.

FDR

PSI5-S Fractional Divider Register **(000C$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
STEP	9:0	rw	**Step Value** Reload or addition value for RESULT.

Field	Bits	Type	Description
DM	15:14	rw	**Divider Mode** DM selects normal or fractional divider mode. 00_B Fractional divider is switched off; no output clock is generated. The Reset External Divider signal is 1. RESULT is not updated (default after System Reset). 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.
RESULT	25:16	rh	**Result Value** Bit field for the addition result.
0	13:10, 31:26	r	**Reserved** Read as 0; should be written with 0.

Fractional Divider Register for Time Stamp

The PSI5-S Module Time Stamp Predivider Register contains the pre divider that defines the time resolution of the Time Stamp Registers TSCNTA/B and the Sync Pulse Time Base Counter SBC. It divides $f_{PSI5\text{-}S}$ by a factor as given in **Equation (43.3)** and **Equation (43.4)**. It contains as well the bits for reset control of the Time Stamp Counters.

FDRT

Fractional Divider Register for Time Stamp (0010_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**ECEB**	**ECEA**	**ECS**			**RESULT**									
r	rw	rw	rw			rh									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		**0**				**STEP**									
rw		r				rw									

Field	Bits	Type	Description
STEP	9:0	rw	**Step Value** Reload or addition value for RESULT.
DM	15:14	rw	**Divider Mode** DM selects normal or fractional divider mode. 00_B Fractional divider is switched off; no output clock is generated. 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated. RESULT is not updated.
RESULT	25:16	rh	**Result Value** Bit field for the addition result.

Field	Bits	Type	Description
ECS	28:26	rw	**External Time Stamp Clear Source Select** Selects the external trigger line that clears the global Time Stamp Counters TSCNTA/B.CTS if this is enabled by ECEA/ECEB. 000_B TRIG0 001_B TRIG1 … 111_B TRIG7
ECEA	29	rw	**External Time Stamp Clear Enable A** Enables the external trigger line selected by ECS to clear the global Time Stamp Counter TSCNTA.CTS on rising edge of the external trigger. 0_B disabled 1_B enabled
ECEB	30	rw	**External Time Stamp Clear Enable B** Enables the external trigger line selected by ECS to clear the global Time Stamp Counter TSCNTB.CTS on rising edge of the external trigger. 0_B disabled 1_B enabled
0	13:10, 31	r	**Reserved** Read as 0; should be written with 0.

Time Stamp Count Register A

This PSI5-S Module Time Stamp Counter Register contains read-only information about the current time given in clock cycles of f_{TS} or f_{TRIGx} since last reset while module was clocked. It also contains the control bits for selection of input clock and reset of this counter.

TSCNTA

Time Stamp Count Register A **(0014_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLRB	**CLRA**	**TBEB**	**TBEA**	**TBS**	**ETB**			**CTS**							
w	w	rw	rw	rw		rw					r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTS															
r															

Field	Bits	Type	Description
CTS	23:0	r	**Current Time Stamp for the Module** This bit field shows the current time stamp.
ETB	26:24	rw	**External Time Base Select** Selects the external clock line for counter CTS if TBS is set. 000_B TRIG0 001_B TRIG1 … 111_B TRIG7

Field	Bits	Type	Description
TBS	27	rw	**Time Base Select** This bit selects the clock source for CTS 0_B Internal, CTS counts in clock cycles of f_{TS} 1_B External, CTS counts in clock cycles received on external clock line selected by ETB
TBEA	28	rw	**Time Base Enable TSCNTA** This bit starts/stops TSCNTA.CTS 0_B CTS stopped (w/o clear of CTS) 1_B CTS started (w/o clear of CTS)
TBEB	29	rw	**Time Base Enable TSCNTB** This bit starts/stops TSCNTB.CTS 0_B CTS stopped (w/o clear of CTS) 1_B CTS started (w/o clear of CTS)
CLRA	30	w	**Clear Time Stamp Counter A** This bit clears TSCNTA.CTS. TSCNTA.CTS counts on, starting from 0.
CLRB	31	w	**Clear Time Stamp Counter B** This bit clears TSCNTB.CTS. TSCNTB.CTS counts on, starting from 0.

Time Stamp Count Register B

This PSI5-S Module Time Stamp Counter Register contains read-only information about the current time given in clock cycles of f_{TS} or f_{TRIGx} since last reset while module was clocked. It also contains the control bits for selection of input clock and reset of this counter.

TSCNTB

Time Stamp Count Register B **(0018_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				TBS	ETB			CTS							
r				rw	rw			r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTS															
r															

Field	Bits	Type	Description
CTS	23:0	r	**Current Time Stamp for the Module** This bit field shows the current time stamp.
ETB	26:24	rw	**External Time Base Select** Selects the external clock line for counter CTS. 000_B TRIG0 001_B TRIG1 … 111_B TRIG7

Field	Bits	Type	Description
TBS	27	rw	**Time Base Select** This bit selects the clock source for CTS 0_B Internal, CTS counts in clock cycles of f_{TS} 1_B External, CTS counts in clock cycles received on external clock line selected by ETB
0	31:28	r	**Reserved** Read as 0; should be written with 0.

Global Control Register

The Global Control Register defines module wide settings.

The first 8 bits define, which error flags are regarded at RSI. RSI indicates that the referring frame is free of the selected errors. Each of these errors can be selected: CRCI, XCRCI, TEI, PE, FE, OE, RBI, HDI.

GCR
Global Control Register **($001C_H$)** **Application Reset Value: $0000\ 001F_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ASC		**0**			**IDT**			**CEN7**	**CEN6**	**CEN5**	**CEN4**	**CEN3**	**CEN2**	**CEN1**	**CEN0**
rw		r			rw			rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETC7	**ETC6**	**ETC5**	**ETC4**	**ETC3**	**ETC2**	**ETC1**	**ETC0**	**HDI**	**RBI**	**OE**	**FE**	**PE**	**TEI**	**XCRCI**	**CRCI**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CRCI	0	rw	**CRCI** is selected if bit is set.
XCRCI	1	rw	**XCRCI** is selected if bit is set.
TEI	2	rw	**TEI** is selected if bit is set.
PE	3	rw	**PE** is selected if bit is set.
FE	4	rw	**FE** is selected if bit is set.
OE	5	rw	**OE** is selected if bit is set.
RBI	6	rw	**RBI** is selected if bit is set.
HDI	7	rw	**HDI** is selected if bit is set.

Field	Bits	Type	Description
ETCx (x=0-7)	x+8	rw	**Enable Channel Trigger Counter CTVx.CTC** This bit enables CTVx.CTC. The bits ETC0 … x can be set with one write access to synchronously start all counters. This is required for proper sync pulse staggering. If set, CTCx counts on, starting from its current value. CTCx can be written only if ETCx is cleared (stopped).
CENx (x=0-7)	x+16	rw	**Enable Channel x** This bit enables PSI5-S Channel x. If cleared, all internal state machines of the receiver and the sender are forced to default idle state while all registers can be read and written. Used for configuration of a channel. Frames received for a disabled channel are copied to ChID 0, FID 1 with original IDs.
IDT	27:24	rw	**Idle Time (GLOBAL VALUE FOR ALL CHANNELS)** determines the number of stop bits in addition to the stop bit of the last UART Frame that is required for SOF detection. (IDT-1) idle bit times are allowed/tolerated within one Packet Frame. Default is IDT = 0, i.e. back to back transfer. 0_H 1 bit time idle … F_H 16 bit time idle
ASC	31	rw	**ASC only Mode** is selected if bit is set. The ASC registers are fully controllable by SW via SPB. If cleared, the ASC is controlled by the message reassembly unit and the message generation unit. RBUF and TBUF are no longer writable by SW and interrupts are handled by the message reassembly block automatically.
0	30:28	r	**Reserved** Read as 0; should be written with 0.

Number of Frames Control Register

This register contains the number of Packet Frames expected after a Sync Pulse.

It stores the compare values for the counters in register FCNT.

See **“FCNT” on Page 59**.

NFC

Number of Frames Control Register (0020$_H$) Application Reset Value: 0024 9249$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								NF7			NF6			NF5	
r								rw			rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NF5		NF4		NF3			NF2			NF1			NF0		
rw		rw		rw			rw			rw			rw		

Field	Bits	Type	Description
NFx (x=0-7)	3*x+2:3*x	rw	**Number of expected Frames on Channel x** 000_B reserved, do not use! Reads back 1. 001_B 1 … 110_B 6 111_B reserved, do not use! Reads back 6.
0	31:24	r	**Reserved** Read as 0; should be written with 0.

Frame Counter Register

This register contains the number of Packet Frames actually received after the last Sync Pulse. A Sync Pulse on channel x will clear FCx.

If FCNT.FCx equals NFC.NFx interrupt CHCI is issued and FCNT.FCx will roll over to '1' with the reception of the next recoverable message on channel x. The roll over happens in any case, even if no Sync Pulse clears FCx after CHCI.

If RCRAx.WDMS is set, NFC and FCNT affect the operation of the watch dog timer. See **"Watch Dog Timers" on Page 15**.

NFC and FCNT also affect the generation of interrupt CHCI. See **"Interrupts for DMA support" on Page 29**.

FCNT
Frame Counter Register **(0024_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NFCLR7	**NFCLR6**	**NFCLR5**	**NFCLR4**	**NFCLR3**	**NFCLR2**	**NFCLR1**	**NFCLR0**	**FC7**			**FC6**			**FC5**	
w	w	w	w	w	w	w	w	rh			rh			rh	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FC5	**FC4**			**FC3**			**FC2**			**FC1**			**FC0**		
rh	rh			rh			rh			rh			rh		

Field	Bits	Type	Description
FCx (x=0-7)	3*x+2:3*x	rh	**Frame Counter for Channel x** Contains the number of received frames on Channel x. Copied to RDR.FID if RCRAx.FIDS is set. 000_B 0 (after reset, Sync (if WDMS is set only) or setting NFCLRx) 001_B 1 … 110_B 6 111_B not valid

Field	Bits	Type	Description
NFCLRx (x=0-7)	x+24	w	**Clear Number of Frame Counter for Channel x** Clears the referring counter FCNT.FCx. Intended for use during recovery from TEI. If set while a frame is being received, this action results in FCx = '1'. Thus FCx will never be '0' when RDI/RSI signal a new receive frame. Use with care!

Input and Output Control Register

The Input and Output Control Register IOCR determines for the PSI5-S:

- the alternate input for the receiver

IOCR

Input and Output Control Register (0028$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														ALTI	
r														rw	

Field	Bits	Type	Description
ALTI	1:0	rw	**Alternate Input Select** Selects the alternate input for RX of the ASC: 00_B Alternate Input 0 selected … 11_B Alternate Input 3 selected
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Receiver Control Register Ax

The Receiver Control Registers RCRAx contain control bits/bit fields that are related to the PSI5-S receiver operation. It enables the channel and determines the use of CRC or parity. CENx must be clear for write access.

RCRAx (x=0-7)

Receiver Control Register Ax (0030$_H$+x*4) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				UFC5		UFC4		UFC3		UFC2		UFC1		UFC0	
r				rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					WDMS	FIDS	TSTS	TSP	TSEN	CRC5	CRC4	CRC3	CRC2	CRC1	CRC0
r					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CRCy (y=0-5)	y	rw	**CRC or Parity Selection** If set, a 3 bit CRC checksum is expected for the PSI5-S channel x in slot y. Else, 1 bit Parity is assumed. This bit field is looked up before potential modification of FID according to FIDS. 0_B 1 Parity Bit is configured (default) 1_B 3 CRC bits are configured
TSEN	6	rw	**Time Stamp Enable** Enables the time stamping for channel x 0_B off (default) TSCRx and thus TSM are forced to 0x0000 0000! 1_B on, see TSP and TSTS
TSP	7	rw	**Time Stamp Select** For non recoverable Packets (stored in ChID ‘0’, FID ’1’ with original IDs); TSCNTA is captured in TSM and not in TSCR0 (independent from TSP). 0_B TSCNTA.CTS is captured in TSCRx 1_B TSCNTB.CTS is captured in TSCRx
TSTS	8	rw	**Time Stamp Trigger Select** 0_B On Sync Pulse. TSCRx is updated on Sync Pulse. TSM is updated from TSCRx on Packet Frame reception after ChID extraction. For non recoverable Packets (stored in ChID ‘0’, FID ’1’ with original IDs); TSCR0 is not updated and TSM is updated with the current value of TSCNTA (independent from TSP). This happens at the time the FSM assumes the Packet Frame to be non recoverable. This allows following good packets to be time stamped correctly. 1_B On any Frame. TSCRx and TSM are updated simultaneously with the current value of TSCNTA/B depending from RCRAx.TSP on Packet Frame reception after ChID extraction. (RDI) This allows SW to read the time of reception on the referring channel from TSCRx. For non recoverable Packets (stored in ChID ‘0’, FID ’1’ with original IDs), TSM is updated with the current value of TSCNTA (independent from TSP and TSEN). This happens at the time the FSM assumes the Packet Frame to be non recoverable. TSCR0 is updated only if RCRA0.TSTS is set (‘1’)
FIDS	9	rw	**Frame ID Select** 0_B Frame ID is updated from Packet Frame Header (sync mode). Channel 0 should have FIDS cleared. This avoids that the module overwrites non recoverable messages with Transceiver Messages. Note that non recoverable messages are always stored in ChID ‘0’, FID ’1’ with original IDs. 1_B Frame ID is a rolling number 0 … 5 copied from FCNT.(FCx-1) (async mode) Non recoverable messages are still stored in ChID ‘0’, FID ’1’ with original IDs.
WDMS	10	rw	**Watch Dog Timer Mode Select** 0_B Watch Dog Timer is restarted on reception of each recoverable frame on Channel x (async mode). 1_B Watch Dog Timer is restarted on Sync Pulse and stopped at reception of the last frame configured in NFC.NFx.(sync mode)

Field	Bits	Type	Description
UFCy (y=0-5)	2*y+17:2*y+16	rw	**UART Frame Count per Packet Frame in Slot y** This bit field defines the number of UART Frames per Packet Frame that are expected for Slot y. This bit field is looked up before potential modification of FID according to FIDS. 00_B 3 UART Frames 01_B 4 UART Frames 10_B 5 UART Frames 11_B 6 UART Frames
0	15:11, 31:28	r	**Reserved** Read as 0; should be written with 0.

Receiver Control Register Bx

The Receiver Control Registers RCRBx configures the number of payload bits and implicitly the number of UART Frames (Bytes) each of the up to 6 slots in channel x. CENx must be clear for write access.

RCRBx (x=0-7)

Receiver Control Register Bx **(0050_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**PDL5**					**PDL4**					**PDL3**			
r		rw					rw					rw			

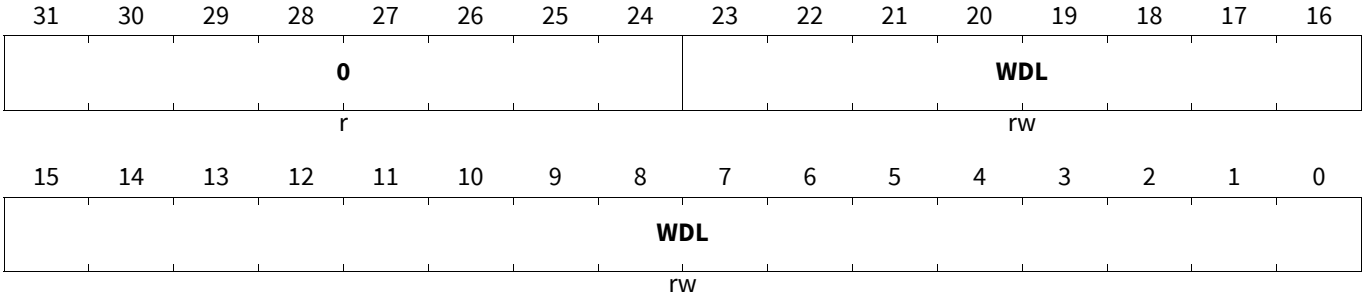
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDL3		**PDL2**				**PDL1**					**PDL0**				
rw		rw				rw					rw				

Field	Bits	Type	Description
PDLy (y=0-5)	5*y+4:5*y	rw	**Payload Data Length** PDL determines the number of bits in a Packet Frame frame for Slot y. It also determines the position of the CRC/Parity bit. E.g. 8 defines 8 data bits on position [7:0]. On bit position 8 the CRC/Parity is located. See **Figure 660**. If PDLy is cleared (‘0’) no frame is expected for this slot. Packet Frames received for a slot with PDL = ‘0’ are copied to ChID 0, FID 1 with original IDs without further processing. This bit field is looked up before potential modification of FID according to RCRAx.FIDS 00_H No Frame expected! 01_H 8 bits … 08_H 8 bits 09_H 9 bits … $1B_H$ 27 bits $1C_H$ 28 bits … $1F_H$ 28 bits

Field	Bits	Type	Description
0	31:30	r	**Reserved** Read as 0; should be written with 0.

Watch Dog Timer Register x

WDTx (x=0-7)
Watch Dog Timer Register x **(0070_H+x*4)** **Application Reset Value: 0000 0000_H**

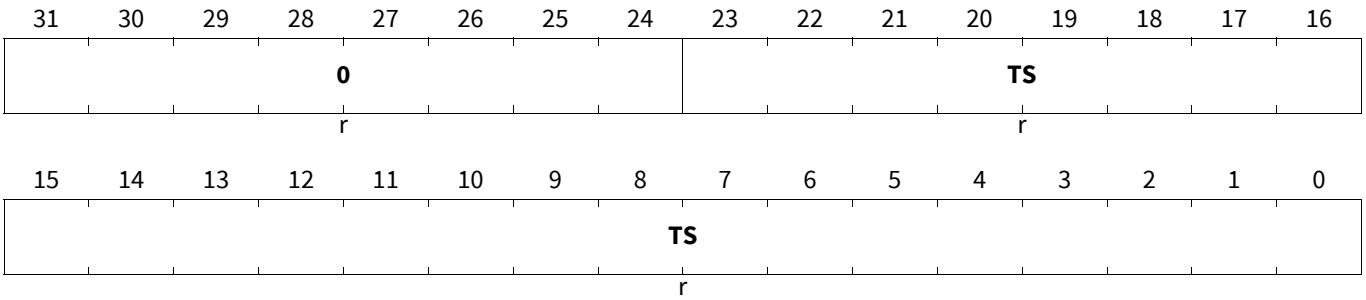


Field	Bits	Type	Description
WDL	23:0	rw	**Watch Dog Timer Limit** for channel x. If no watch dog is needed, WDL is cleared, the internal watch dog timer is stopped and no check is performed. CENx must be clear for write access.
0	31:24	r	**Reserved** Read as 0; should be written with 0.

Capture Register TSCRx

If RCRAx.TSEN is set, the time stamp is captured in this register each time a sync pulse is sent from the ASC FIFO to the ASC TX buffer for channel x. RCRAx.TSP selects, if TSCNTA or B is used.

TSCRx (x=0-7)
Capture Register TSCRx **(0090_H+x*4)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TS	23:0	r	**Time Stamp** of the last sync pulse sent for channel x.
0	31:24	r	**Reserved** Read as 0; should be written with 0.

Receive Status Register

The Receive Status Register RDS shows the status of a received data frame.

The internal receive buffer is always cleared (0x0000 0000$_H$) at each frame start. Thus unused bits are always read as zero.

The bits CRCI, XCRCI, HDI, PE, FE, OE, TEI and RBI reflect, wether the conditions are met to issue an interrupt signal and set the referring interrupt status bit in INTSTAT or CON (ASC sub module) for the latest frame. Thus it is independent from the old status of the referring sticky bit in INTSTAT or CON before latest frame reception.

RDS

Receive Status Register (00B0$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PFC				AFC			CID			FID			RBI	TEI	OE
rh				rh			rh			rh			rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FE	PE	HDI	ERR1	ERR0	CRCI	CRC2	CRC1	CRC0	XCRCI	XCRC5	XCRC4	XCRC3	XCRC2	XCRC1	XCRC0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
XCRCy (y=0-5)	y	rh	**XCRC** CRC of last Packet Frame. XCRC0 is on bit position 0.
XCRCI	6	rh	**XCRC Error Flag** This bit is set if the CRC check on the enveloping Packet Frame fails including the case where XCRC check can not be performed (non recoverable frames) see **Chapter 43.3.2.1.1**. 0_B correct XCRC 1_B wrong XCRC
CRCy (y=0-2)	y+7	rh	**CRC** of last frame. CRC0 / Parity is on bit position 7. If Parity is used, CRC1/2 are always 0.
CRCI	10	rh	**CRC Error Flag** This bit is set if the CRC or Parity check on the transported PSI5 frame fails. 0_B correct CRC/Parity 1_B wrong CRC/Parity
ERR0	11	rh	**Error signalling Flag 0** This bit represents the status of the error signalling flag Err0 in the enveloping Packet Frame.
ERR1	12	rh	**Error signalling Flag 1** This bit represents the status of the error signalling flag Err1 in the enveloping Packet Frame.

Field	Bits	Type	Description
HDI	13	rh	**Header Error Signalled Flag** This bit is set if at least one of the error signalling flags in the enveloping Packet Frame Err0 and Err1 is set. 0_B (Err0 OR Err1) = false (0) 1_B (Err0 OR Err1) = true (1)
PE	14	rh	**ASC Parity Error Flag** This bit is set if the error flag signalling a parity error was set during reception of one of the ASC Bytes transporting this PSI5 frame.
FE	15	rh	**ASC Framing Error Flag** This bit is set if the error flag signaling a framing error was set during reception of one of the ASC Bytes transporting this PSI5 frame.
OE	16	rh	**ASC Overrun Error Flag** This bit is set if the error flag signaling an overrun error was set during reception of one of the ASC Bytes transporting this PSI5 frame.
TEI	17	rh	**Time Error Flag** This bit is set if the watch dog timer expired. Depending from RCRAx.WDMS either the distance between two RDIs is longer than specified in WDL or it expired without reception of CHCI in time. I.e. the time from issuing the sync pulse to reception of the last expected frame configured in NFC.NFx was too long. If WDMS is in synchronous mode, all frames after TEI have TEI set in RDS until either CHCI is issued or INSTATx.TEI is cleared by SW. RDS.TEI flag is independently from INTENx.TEI. 0_B no error 1_B error
RBI	18	rh	**Receive Buffer Overflow Flag** This bit is set after a frame has been received while the old one was not read from RDR. I.e. the kernel wants to set interrupt RDI and finds RDI already set. The old data is overwritten by the new data. 0_B No overflow 1_B Overflow
FID	21:19	rh	**Frame ID (Frame Number)** See bit RCRAx.FIDS for actual content. 000_B Slot0 … 101_B Slot5 110_B not valid, frame is copied to ChID 0, FID 1 with original IDs 111_B not valid, frame is copied to ChID 0, FID 1 with original IDs
CID	24:22	rh	**Channel ID (Channel Number)** 000_B channel 0 001_B channel 1 … 111_B channel 7

Field	Bits	Type	Description
AFC	27:25	rh	**Actual UART Frame Count** This bit field shows the number of UART frames actually received. This is used to further analyze non recoverable frames by SW or during debugging.
PFC	31:28	rh	**Packet Frame Count** For data consistency, RDR, RDS and TSM are tagged with a Packet Frame count value. For each channel there is a separate internal packet counter. This allows to read a triplet of RDR, RDS and TSM from the system memory without looking at interrupts. If the PFC is identical all three values belong together. Otherwise they need to be read again until PFC is identical.

Receive Data Register

The Receive Data Register RDR shows the data content of a received data frame. If Messaging bits are used, they are located on bit position [1:0].

The internal receive buffer is always cleared (0x0000 0000_H) at each frame start. Thus unused bits are always read as zero.

RDR
Receive Data Register **($00B4_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PFC				**RD27**	**RD26**	**RD25**	**RD24**	**RD23**	**RD22**	**RD21**	**RD20**	**RD19**	**RD18**	**RD17**	**RD16**
rh				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

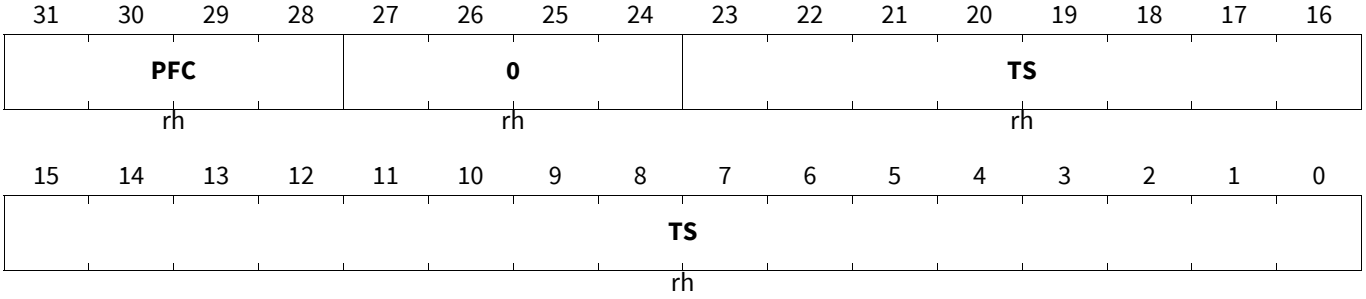
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RD15	**RD14**	**RD13**	**RD12**	**RD11**	**RD10**	**RD9**	**RD8**	**RD7**	**RD6**	**RD5**	**RD4**	**RD3**	**RD2**	**RD1**	**RD0**
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RDy (y=0-27)	y	rh	**PSI5 Receive Data** of last frame. D0 is on bit position 0.
PFC	31:28	rh	**Packet Frame Count** For data consistency, RDR, RDS and TSM are tagged with a Packet Frame count value. For each channel there is a separate internal packet counter. This allows to read a triplet of RDR, RDS and TSM from the system memory without looking at interrupts. If the PFC is identical all three values belong together. Otherwise they need to be read again until PFC is identical.

Time Stamp Mirror Register

Each time a new frame is received, the value from TSCRx for the referring channel is mirrored here. This register is updated, after XCRC was checked ok, as the referring channel number can not be securely determined before.

TSM
Time Stamp Mirror Register **($00B8_H$)** **Application Reset Value: $0000\ 0000_H$**

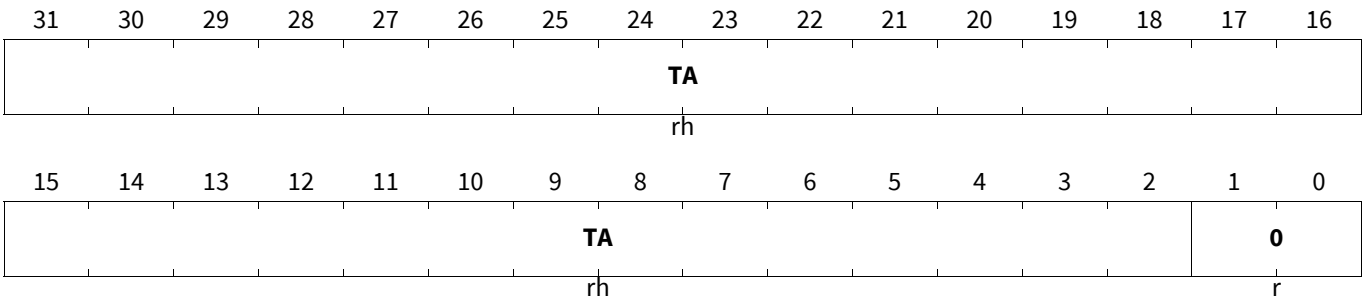


Field	Bits	Type	Description
TS	23:0	rh	**Time Stamp** of the last sync pulse sent on channel x.
PFC	31:28	rh	**Packet Frame Count** For data consistency, RDR, RDS and TSM are tagged with a Packet Frame count value. For each channel there is a separate internal packet counter. This allows to read a triplet of RDR, RDS and TSM from the system memory without looking at interrupts. If the PFC is identical all three values belong together. Otherwise they need to be read again until PFC is identical.
0	27:24	rh	**Reserved** Read as 0; should be written with 0.

Target Address Register

TAR contains the address that is to be copied by the first DMA into the target address register of the second DMA. This is required for the use case with 2 DMAs. See **"DMA Support" on Page 23**

TAR
Target Address Register **($00D0_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TA	31:2	rh	**Target Address** Contains the upper 30 bit of the target address for the next DMA transfer. The 32 bit target address must be word aligned. Thus the 2 LSBs are fixed to 0. It is updated each time a new Packet Frame is received completely.

Field	Bits	Type	Description
0	1:0	r	**Reserved** Read as 0; should be written with 0.

Base Address Register

BAR contains the address in the system memory where the PSI5-S buffer for all channels is built up.

BAR must be configured by the application for an address space that is big enough for all channels that are enabled. TAR will wrap around after the upper limit (0xFFFF FFFF) if adding the offsets (based on ChID and FID) exceeds this limit.

BAR
Base Address Register **($00D4_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BA															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BA														0	
rw														r	

Field	Bits	Type	Description
BA	31:2	rw	**Base Address** Contains the upper 30 bits of the base address for the DMA transfers. The 32 bit base address must be word aligned. Thus the 2 LSBs are fixed to 0.
0	1:0	r	**Reserved** Read as 0; should be written with 0.

Pulse Generation Control Register x

The Pulse Generation Control Register PGC contains control data for the sync pulse generation. It contains as well the trigger control bits required for sending data from the PSI5-S module to the sensor / external PHY.

PGCx (x=0-7)
Pulse Generation Control Register x **($00F0_H$+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								ETE	ETS			PTE	ETB		
r								rw	rw			rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBS	0		ATXCMD					0			TXCMD				
rw	r		rw					r			rw				

Field	Bits	Type	Description
TXCMD	4:0	rw	**TX Command** Defines the value that is copied to the ASC FIFO for coding a '0'.
ATXCMD	12:8	rw	**Alternate TX Command** Defines the value that is copied to the ASC FIFO for the alternate pulse width i.e. for coding a '1'.
TBS	15	rw	**Time Base Select** This bit selects the clock source for CTVx 0_B Internal, CTV counts in clock cycles of f_{TS} 1_B External, CTV counts in clock cycles of f_{TRIGx} according to the setting of bit ETB.
ETB	18:16	rw	**External Time Base Select** Selects the external clock line for counter CTVx. 000_B TRIG0 001_B TRIG1 … 111_B TRIG7
PTE	19	rw	**Periodic Trigger Enable** Periodic trigger is defined by CTVx. Should be 0 if ETE is set. 0_B disabled 1_B enabled
ETS	22:20	rw	**External Trigger Select** Selects the external trigger line for pulse generation (e.g. angle synchronous). 000_B TRIG0 001_B TRIG1 … 111_B TRIG7
ETE	23	rw	**External Trigger Enable** "Angle sync. trigger", external line is selected by ETS. Should be 0 if PTE is set. 0_B disabled 1_B enabled
0	7:5, 14:13, 31:24	r	**Reserved** Read as 0; should be written with 0.

Channel Trigger Value Register x

CTV contains the value that determines the period of periodic triggers for each channel. It contains as well the counter that can be initialized with an offset so that the phase of harmonic trigger frequencies on different channels can be staggered.

CTVx (x=0-7)

Channel Trigger Value Register x **(0110_H+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTC															
rwh															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTV															
rw															

Field	Bits	Type	Description
CTV	15:0	rw	**Channel Trigger Value CTV** Contains the compare value (exact match) of Channel Trigger CTC at which a sync pulse is triggered for channel x and the counter CTC is cleared. If cleared, CTC is stopped and no pulse triggered.
CTC	31:16	rwh	**Channel Trigger Counter** This bit field allows to read the current counter value of the reset timer cell CTVx. If GCR.ETCx is cleared, CTC can be written.

Send Control Register x

The Send Control Register SCRx contains control data bits required for sending data from the PSI5-S module to the sensor / external PHY.

Data is collected in SDRx and the control bits are collected in SCRx.

The send control unit is build to transmit complete ECU to sensor frames.

SCRx (x=0-7)

Send Control Register x **(0130_H+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					TPF	0		STA	CRC	0					
r					r	r		rw	rw	r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	FLUS	0					BSC	EPS		0	PLL				
r	w	r					rw	rw		r	rw				

Field	Bits	Type	Description
PLL	4:0	rw	**Pay Load Length of Registers SDRx** Defines the length that is taken into account. PLL needs to written before SDRx is used for proper operation. 00_H length is 6 … 06_H length is 6 07_H length is 7 … 17_H length is 23 18_H length is 24 … $1F_H$ length is 24
EPS	7:6	rw	**Enhanced Protocol Selection** EPS[0] controls the Bit Format and MSB Fill bit. (0: Tooth Gap, MSB Fill 1) (1: PWM, MSB Fill 0) EPS[1] controls the Frame Format (0: Frame Format 1 - 3) (1: Frame Format 4) 00_B Tooth Gap Method, '1' for filling the shift register from MSB; Frame format 1 -3 (3 bit start sequence, 3 bit stuffing distance, '1' for stuffing, 3 bit CRC). 01_B Pulse Width Method, '0' for filling the shift register from MSB; Frame format 1 -3 (3 bit start sequence, 3 bit stuffing distance, '1' for stuffing, 3 bit CRC). 10_B reserved, do not use! 11_B Pulse Width Method, '0' for filling the shift register from MSB; Frame Format 4 (9 bit start sequence, 6 bit stuffing distance, '0' for stuffing, 6 bit CRC).
BSC	8	rw	**Bit Stuffing Control** Depending from bit EPS[1] after 3 bits a '1' is inserted (Frame Format 1 - 3) or after 6 bits a '0' is inserted (Frame Format 4) 0_B No automatic bit stuffing 1_B Automatic bit stuffing is enabled.
FLUS	14	w	**Flush SDRx** Setting this bit stops shifting out the data in SDRx, the start sequence or CRC if any and clears the referring FSMs and counters, if EPS[0]=0 SDRx is flushed by setting all bits if EPS[0]=1 SDRx is flushed by clearing all bits clears TPF TPIx is issued at the end of successful flushing. Reads always as zero.
CRC	22	rw	**CRC Generation Control** 0_B CRC is not generated automatically, it still can be written by SW together with the data (e.g. to test the remote CRC) 1_B CRC is automatically generated by HW (according to EPS[1])
STA	23	rw	**Start Sequence Generation Control** 0_B no start sequence generated, shifting out payload starts at bit 0! 1_B automatically generated by HW (according to EPS[1]) shifting out payload starts after 3/9 bits (EPS[1] = 0/1)

Field	Bits	Type	Description
TPF	26	r	**Transmit in Progress Flag** If set, data preparation and transmission is in progress: start sequence or CRC or stuffing bits or data from SDRx are being transferred. It is cleared automatically after preparation and transmitting is finished. If set, write access to SDRx will not change any data and issue TPOI.
0	5, 13:9, 21:15, 25:24, 31:27	r	**Reserved** Read as 0; should be written with 0.

Send Data Register x

The Send Data Register SDRx shows the data content of a data frame to be sent.

CENx must be set for write access. For initialization SCRx.FLUS can be used.

SDRx (x=0-7)
Send Data Register x **(0150_H+x*4)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								SD23	SD22	SD21	SD20	SD19	SD18	SD17	SD16
r								rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
SDy (y=0-23)	y	rwh	**SDy** Send data of next ECU to Sensor frame. Each time a bit is shifted out, the whole content is shifted right by 1 position and the MSB is filled with ‘1’ or ‘0’ depending from EPS[0]. This allows to read back SDR and determine the status of the shift process by SW. The unused MSBs (bit position SCRx.PLL and higher) must be written with ‘0’ if EPS[0] is set (PWM Method) and with ‘1’ if EPS[0] is cleared (Tooth Gap Method). This is required, as the respective standard value is shifted into the register from MSB during shift out operation. SDRx will be filled with this value after shift out process.
0	31:24	r	**Reserved** Read as 0.

CPU Direct Write Register

The CPU Direct Write Register CDW allows the CPU to insert a command into the ASC FIFO. Note that Bits SD[7:5] select the channel for which the time stamp is captured. These bits are copied to the UART Frame Bit positions [2:0] (ChID) while SD[4:0] are copied to UART Frame Bit positions [7:3] (Command). See **Figure 662**.

CDW
CPU Direct Write Register **(0170_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							TSI	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SDy (y=0-7)	y	rw	**SDy** Send data of next ECU to Sensor frame.
TSI	8	rw	**Trigger Pulse Indicator** If this bit is set, a sync pulse is assumed and thus a time stamp is captured when the command leaves the FIFO and is written to ASC TX Register. Bits SD[7:5] select the channel for which the time stamp is captured. RCRAx.TSTS must be cleared. If set, the time stamp is captured on Packet Frame reception only.
0	31:9	r	**Reserved** Read as 0.

Control Register

The serial operating modes of the ASC module are controlled by its Control Register CON. This register contains control bits for mode and error check selection, and status flags for error identification.

CON
Control Register **(0210_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			ODDTX	0									MTX		
r			rw	r									rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LB	BRS	ODD	FDE	OE	FE	PE	OEN	FEN	PEN	REN	STP	M		
rw	rw	rw	rw	rw	rh	rh	rh	rw	rw	rw	rh	rw	rw		

Field	Bits	Type	Description
M	2:0	rw	**Mode Selection** 000_B 8-bit dataSynchronous Mode. MTX needs to be cleared as well for proper operation. 001_B 8-bit data Asynchronous Mode 010_B Reserved. Do not use this combination. 011_B 7-bit data + parity Asynchronous Mode 100_B 9-bit dataAsynchronous Mode 101_B 8-bit data + wake up bit Asynchronous Mode 110_B Reserved. Do not use this combination. 111_B 8-bit data + parity Asynchronous Mode
STP	3	rw	**Number of Stop Bit Selection** 0_B One stop bit 1_B Two stop bits
REN	4	rh	**Receiver Enable Control** Bit is reset by hardware after reception of a Byte in Synchronous Mode. 0_B Receiver disabled 1_B Receiver enabled
PEN	5	rw	**Parity Check Enable (asynchronous mode only)** 0_B Ignore parity 1_B Check parity
FEN	6	rw	**Framing Check Enable (asynchronous mode only)** 0_B Ignore framing errors 1_B Check framing errors
OEN	7	rw	**Overrun Check Enable** 0_B Ignore overrun errors 1_B Check overrun errors
PE	8	rh	**ASC Parity Error Flag** Set by hardware on a parity error (PEN = 1). Must be reset by software.
FE	9	rh	**ASC Framing Error Flag** Set by hardware on a framing error (FEN = 1). Must be reset by software.
OE	10	rh	**ASC Overrun Error Flag** Set by hardware on an overrun error (OEN = 1). Must be reset by software.
FDE	11	rw	**Fractional Divider Enable** FDE is don't care and assumed '0' in Synchr. Mode. 0_B Fractional divider disabled 1_B Fractional divider is enabled and used as prescaler for baud rate timer (bit BRS is don't care)
ODD	12	rw	**Parity Selection** 0_B Even parity selected (parity bit = 1 on odd number of 1s in data, parity bit = 0 on even number of 1s in data) 1_B Odd parity selected (parity bit = 1 on even number of 1s in data, parity bit = 0 on odd number of 1s in data)

Field	Bits	Type	Description
BRS	13	rw	**Baud Rate Selection** BRS is don't care if FDE = 1 (fractional divider enabled) FDE is don't care and assumed '0' in Synchr. Mode. 0_B Baud rate timer prescaler divide-by-2 selected 1_B Baud rate timer prescaler divide-by-3 selected
LB	14	rw	**Loop-back Mode Enable** 0_B Loop-Back mode disabled 1_B Loop-Back mode enabled
R	15	rw	**Baud Rate Generator Run Control** Register BG should only be written if R = 0. 0_B Baud rate generator disabled (ASC inactive) 1_B Baud rate generator enabled
MTX	18:16	rw	**Mode Selection TX direction** While bit field M controls the RX path, MTX controls the mode for the TX path. 000_B 8-bit data Synchronous Mode. M needs to be cleared as well for proper operation. 001_B 8-bit data Asynchronous Mode 010_B Reserved. Do not use this combination. 011_B 7-bit data + parity Asynchronous Mode 100_B 9-bit dataAsynchronous Mode 101_B 8-bit data + wake up bit Asynchronous Mode 110_B Reserved. Do not use this combination. 111_B 8-bit data + parity Asynchronous Mode
ODDTX	28	rw	**Parity Selection TX direction** While bit field ODD controls the RX path, ODDTX controls the mode for the TX path. 0_B Even parity selected (parity bit = 1 on odd number of 1s in data, parity bit = 0 on even number of 1s in data) 1_B Odd parity selected (parity bit = 1 on even number of 1s in data, parity bit = 0 on odd number of 1s in data)
0	27:19, 31:29	r	**Reserved** Read as 0; should be written with 0.

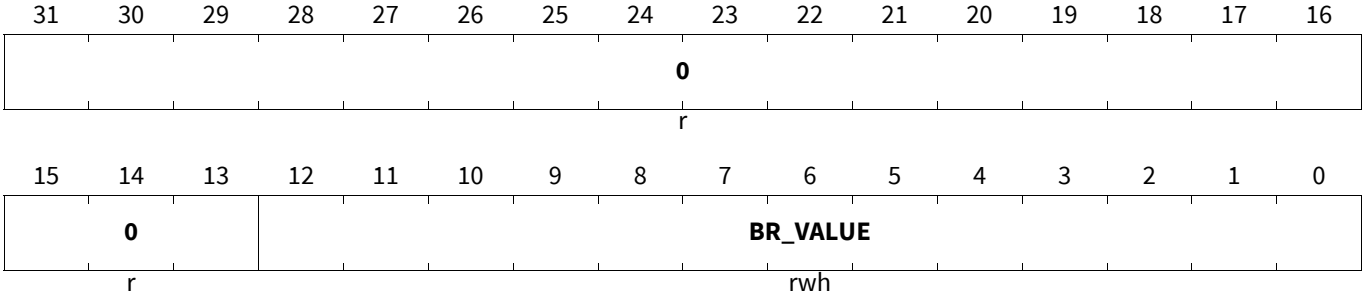
Serial data transmission or reception is possible only when the run bit CON.R is set to 1. Otherwise, the serial interface is idle. To avoid unpredictable behavior of the serial interface, do not program the mode control field CON.M to one of the reserved combinations.

Baud Rate Timer/Reload Register

The Baud Rate Timer Reload Register BG of the ASC module contains the 13-bit reload value for the baud rate timer in Asynchronous and Synchronous Modes.

BG

Baud Rate Timer/Reload Register (0214$_H$) Application Reset Value: 0000 0000$_H$



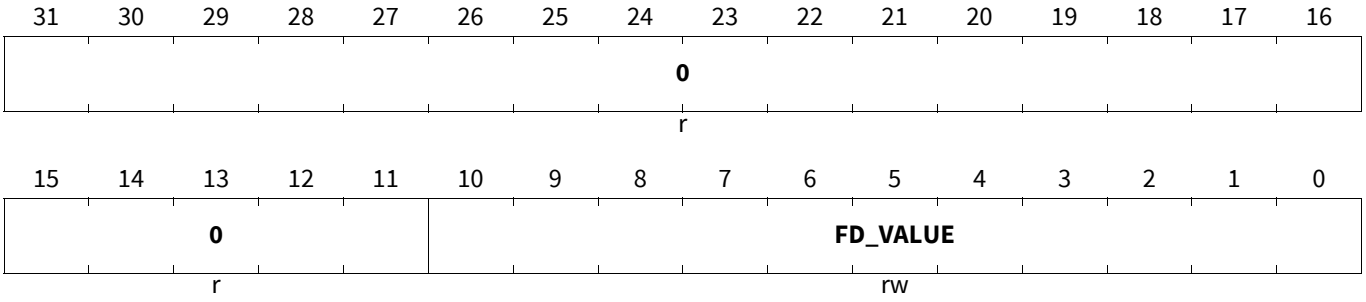
Field	Bits	Type	Description
BR_VALUE	12:0	rwh	**Baud Rate Timer/Reload Register Value** Reading BR_VALUE returns the 13-bit content of the baud rate timer. Writing BR_VALUE loads the baud rate timer reload register. BG should only be written if CON.R = 0.
0	31:13	r	**Reserved** Read as 0; should be written with 0.

Fractional Divider Register

The Fractional Divider Register FDV of the ASC module contains the 11-bit divider value for the fractional divider (asynchronous mode only).

FDV

Fractional Divider Register (0218$_H$) Application Reset Value: 0000 0000$_H$



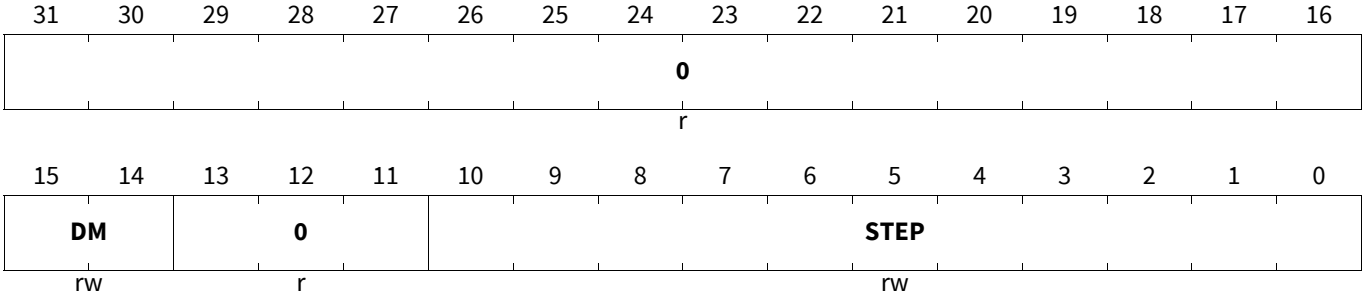
Field	Bits	Type	Description
FD_VALUE	10:0	rw	**Fractional Divider Register Value** FD_VALUE contains the 11-bit value n of the fractional divider which determines the fractional divider ratio n/2048 (n = 0-2047). With n = 0, the fractional divider is switched off (divider ratio = 1).
0	31:11	r	**Reserved** Read as 0; should be written with 0.

Fractional Divider for Output CLK Register

The Fractional Divider Register FDO contains the 11-bit divider value for the fractional divider that is generating $f_{PSISCLK}$ from f_{ASC}. PSISCLK is a clock that can be used on a pin to drive the external PHY.

FDO

Fractional Divider for Output CLK Register (021C$_H$) Application Reset Value: 0000 0000$_H$



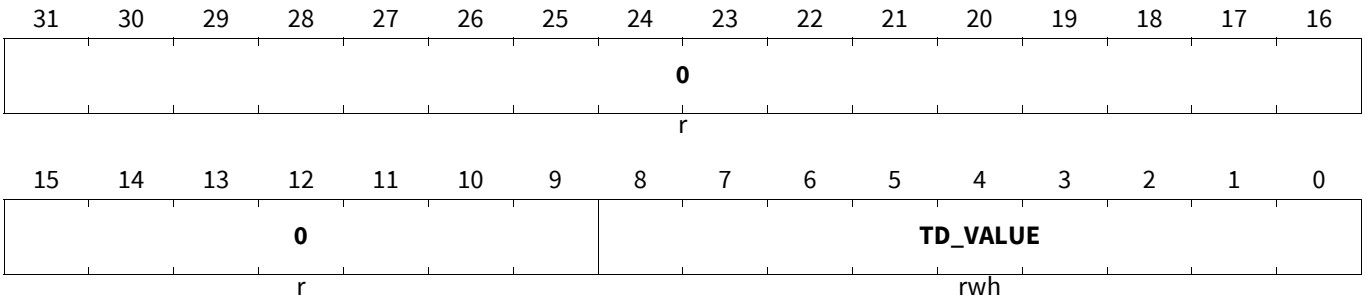
Field	Bits	Type	Description
STEP	10:0	rw	**Step Value** Reload or addition value for internal accumulator.
DM	15:14	rw	**Divider Mode** DM selects normal or fractional divider mode. 00_B Fractional divider is switched off; no output clock is generated. The Reset External Divider signal is 1. (default after System Reset). 01_B Normal Divider Mode selected. 10_B Fractional Divider Mode selected. 11_B Fractional divider is switched off; no output clock is generated.
0	13:11, 31:16	r	**Reserved** Read as 0; should be written with 0.

Transmit Buffer Register

The Transmit Buffer Register TBUF of the ASC module contains the transmit data value in Asynchronous And Synchronous Modes. If GCR.ASC is cleared, TBUF is no longer writable by SW and interrupts are handled by the message reassembly block automatically.

TBUF

Transmit Buffer Register (0220$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TD_VALUE	8:0	rwh	**Transmit Data Register Value** TBUF contains the data to be transmitted in the asynchronous and synchronous operating modes of the ASC. Data transmission is double-buffered; therefore, a new value can be written to TBUF before the transmission of the previous value is complete.

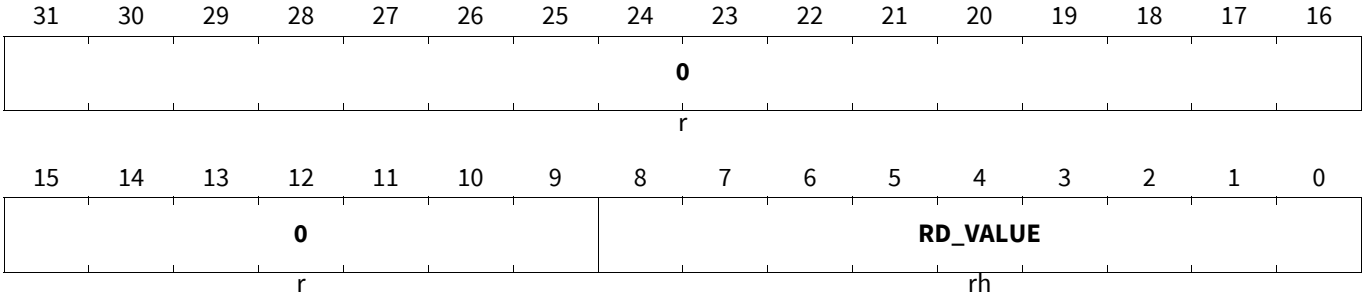
Field	Bits	Type	Description
0	31:9	r	**Reserved** Read as 0; should be written with 0.

Receive Buffer Register

The receive buffer register RBUF of the ASC module contains the receive data value in Asynchronous and Synchronous Modes.

RBUF

Receive Buffer Register **(0224_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RD_VALUE	8:0	rh	**Receive Data Register Value** RBUF contains the received data bits and, depending on the selected mode, the parity bit in the asynchronous and synchronous operating modes of the ASC. In Asynchronous Mode, with CON.M = 011_B (7-bit data + parity), the received parity bit is written into RBUF.7. In Asynchronous Mode, with CON.M = 111_B (8-bit data + parity), the received parity bit is written into RBUF.8.
0	31:9	r	**Reserved** Read as 0.

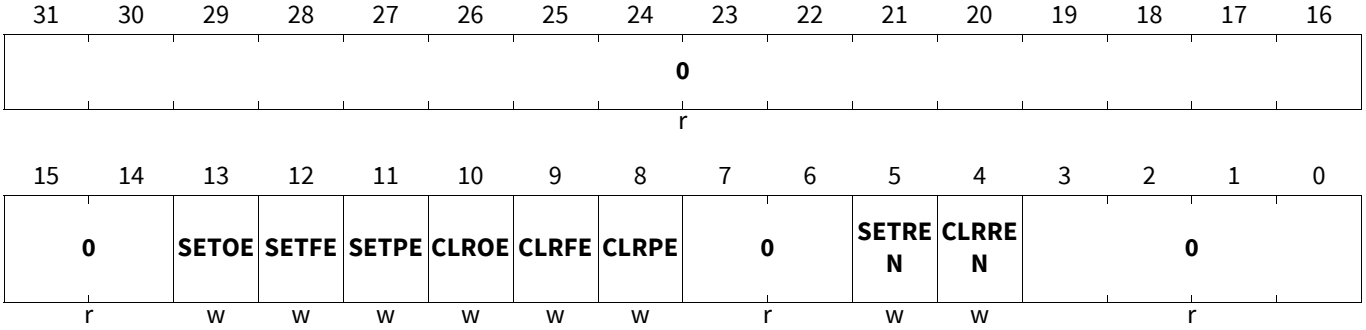
Write Hardware Bits Control Register

The three error flags in register CON and the REN bit can be set or cleared by software via register WHBCON. WHBCON is a write-only register. Reading WHBCON always returns $0000\ 0000_H$.

Note: When the set and clear bits for an error flag are set at the same time during a WHBCON write operation (e.g SETPE = CLRPE = 1), the error flag in CON is not affected.

WHBCON

Write Hardware Bits Control Register (0250$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
CLRREN	4	w	**Clear Receiver Enable Bit** Bit is always read as 0. 0_B No effect 1_B Bit CON.REN is cleared.
SETREN	5	w	**Set Receiver Enable Bit** Bit is always read as 0. 0_B No effect 1_B Bit CON.REN is set.
CLRPE	8	w	**Clear Parity Error Flag** Bit is always read as 0. 0_B No effect 1_B Bit CON.PE is cleared.
CLRFE	9	w	**Clear Framing Error Flag** Bit is always read as 0. 0_B No effect 1_B Bit CON.FE is cleared.
CLROE	10	w	**Clear Overrun Error Flag** Bit is always read as 0. 0_B No effect 1_B Bit CON.OE is cleared.
SETPE	11	w	**Set Parity Error Flag** Bit is always read as 0. 0_B No effect 1_B Bit CON.PE is set.
SETFE	12	w	**Set Framing Error Flag** Bit is always read as 0. 0_B No effect 1_B Bit CON.FE is set.
SETOE	13	w	**Set Overrun Error Flag** Bit is always read as 0. 0_B No effect 1_B Bit CON.OE is set.

Field	Bits	Type	Description
0	3:0, 7:6, 31:14	r	**Reserved** Read as 0; should be written with 0.

Interrupt Status Register x

The Interrupt Status Registers INTSTATx contains status bits that show the status of any interrupt of PSI5-S channel x.

The bits are set independently from the referring Interrupt Enable in Register INTENx. Thus they can be used as status bits as well e.g. by a SW based on polling.

INTSTATx (x=0-7)

Interrupt Status Register x **(0260_H+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**HDI**	**TPOI**	**TPI**	**CRCI**	**CHCI**	**TEI**	**RBI**	**RDI**	**RSI**
r							rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RSI	0	rh	**Receive Success Interrupt Request Flag** This bit is set at the successfully received end of a frame. It indicates that this frame is free of the errors CRCI, XCRCI, TEI, PE, FE, OE, RBI, HDI if selected to be taken into account in register GCR. This bit can be cleared by bit INTCLRx.RSI. This bit can be set by bit INTSETx.RSI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
RDI	1	rh	**Receive Data Interrupt Request Flag** RDI is activated when a received frame is moved to a Receive Data Register RDR. Both RDI and RSI will be issued together at correct reception. This bit can be cleared by bit INTCLRx.RDI. This bit can be set by bit INTSETx.RDI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

Field	Bits	Type	Description
RBI	2	rh	**Receive Buffer Overflow Interrupt Request Flag** This bit is set after a frame has been received while the old one was not read from RDR. I.e. the kernel wants to set interrupt RDI and finds this interrupt already set. The old data is overwritten by the new data. This bit is NOT cleared by reading RDR. This bit can be cleared by bit INTCLRx.RBI. This bit can be set by bit INTSETx.RBI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
TEI	3	rh	**Timing Error Interrupt Request Flag** This bit is set if the watch dog timer expired. Depending from RCRAx.WDMS either the distance between two RDIs is longer than specified in WDL or it expired without reception of CHCI in time. I.e. the time from issuing the sync pulse to reception of the last expected frame configured in NFC.NFx was too long. Note that the root cause might be a non recoverable frame! This bit can be cleared by bit INTCLRx.TEI. This bit can be set by bit INTSETx.TEI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
CHCI	4	rh	**Channel Completed Interrupt Request Flag** This bit is set if FCNT.FCx equals NFC.NFx. This bit can be cleared by bit INTCLRx.CHCI. This bit can be set by bit INTSETx.CHCI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
CRCI	5	rh	**CRC Error Request Flag** This bit is set if the CRC fails. This bit can be cleared by bit INTCLRx.CRCI. This bit can be set by bit INTSETx.CRCI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
TPI	6	rh	**Transfer Preparation Interrupt Request Flag** This bit is set after data to be transferred has been moved completely. Thus a new value can be written to SDRx. This bit can be cleared by bit INTCLRx.TPI. This bit can be set by bit INTSETx.TPI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

Field	Bits	Type	Description
TPOI	7	rh	**Transmit Preparation Overflow Interrupt Request Flag** This bit is set if SDR is written while TPF is set. The old data is NOT overwritten. This bit can be cleared by bit INTCLRx.TPOI. This bit can be set by bit INTSETx.TPOI. This bit is set independently from INTENx. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
HDI	8	rh	**Header Error Signalled Flag** This bit is set if at least one of the error signalling flags in the enveloping Packet Frame Err0 and Err1 is set. Up 0_B (Err0 OR Err1) = false (0) 1_B (Err0 OR Err1) = true (1)
0	31:9	r	**Reserved** Read as 0.

Interrupt Set Register x

The Interrupt Set Registers INTSETx contain control bits that trigger an interrupt pulse for any interrupt of PSI5-S channel x.

INTSETx (x=0-7)

Interrupt Set Register x **(0280_H+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							HDI	TPOI	TPI	CRCI	CHCI	TEI	RBI	RDI	RSI
r							w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RSI	0	w	**Set Interrupt Request Flag RSI** Setting this bit set bit INTSTATx.RSI. Clearing this bit has no effect. Reading this bit returns always zero.
RDI	1	w	**Set Interrupt Request Flag RDI** Setting this bit set bit INTSTATx.RDI. Clearing this bit has no effect. Reading this bit returns always zero.
RBI	2	w	**Set Interrupt Request Flag RBI** Setting this bit set bit INTSTATx.RBI. Clearing this bit has no effect. Reading this bit returns always zero.

Field	Bits	Type	Description
TEI	3	w	**Set Interrupt Request Flag TEI** Setting this bit set bit INTSTATx.TEI. Clearing this bit has no effect. Reading this bit returns always zero.
CHCI	4	w	**Set Interrupt Request Flag CHCI** Setting this bit set bit INTSTATx.CHCI. Clearing this bit has no effect. Reading this bit returns always zero.
CRCI	5	w	**Set Interrupt Request Flag CRCI** Setting this bit set bit INTSTATx.CRCI. Clearing this bit has no effect. Reading this bit returns always zero.
TPI	6	w	**Set Interrupt Request Flag TPI** Setting this bit set bit INTSTATx.TPI. Clearing this bit has no effect. Reading this bit returns always zero.
TPOI	7	w	**Set Interrupt Request Flag TPOI** Setting this bit set bit INTSTATx.TPOI. Clearing this bit has no effect. Reading this bit returns always zero.
HDI	8	w	**Set Interrupt Request Flag HDI** Setting this bit set bit INTSTATx.HDI. Clearing this bit has no effect. Reading this bit returns always zero.
0	31:9	r	**Reserved** Read as 0; should be written with 0.

Interrupt Clear Register x

The Interrupt Clear Register INTCLRx contain control bits that clear the status of any interrupt of PSI5-S channel x.

INTCLRx (x=0-7)

Interrupt Clear Register x **(02A0$_H$+x*4)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**HDI**	**TPOI**	**TPI**	**CRCI**	**CHCI**	**TEI**	**RBI**	**RDI**	**RSI**
r							w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
RSI	0	w	**Clear Interrupt Request Flag RSI** Setting this bit clears bit INTSTATx.RSI. Clearing this bit has no effect. Reading this bit returns always zero.
RDI	1	w	**Clear Interrupt Request Flag RDI** Setting this bit clears bit INTSTATx.RDI. Clearing this bit has no effect. Reading this bit returns always zero.
RBI	2	w	**Clear Interrupt Request Flag RBI** Setting this bit clears bit INTSTATx.RBI. Clearing this bit has no effect. Reading this bit returns always zero.
TEI	3	w	**Clear Interrupt Request Flag TEI** Setting this bit clears bit INTSTATx.TEI. Clearing this bit has no effect. Reading this bit returns always zero.
CHCI	4	w	**Clear Interrupt Request Flag CHCI** Setting this bit clears bit INTSTATx.CHCI. Clearing this bit has no effect. Reading this bit returns always zero.
CRCI	5	w	**Clear Interrupt Request Flag CRCI** Setting this bit clears bit INTSTATx.CRCI. Clearing this bit has no effect. Reading this bit returns always zero.
TPI	6	w	**Clear Interrupt Request Flag TPI** Setting this bit clears bit INTSTATx.TPI. Clearing this bit has no effect. Reading this bit returns always zero.
TPOI	7	w	**Clear Interrupt Request Flag TPOI** Setting this bit clears bit INTSTATx.TPOI. Clearing this bit has no effect. Reading this bit returns always zero.
HDI	8	w	**Clear Interrupt Request Flag HDI** Setting this bit clears bit INTSTATx.HDI. Clearing this bit has no effect. Reading this bit returns always zero.
0	31:9	r	**Reserved** Read as 0; should be written with 0.

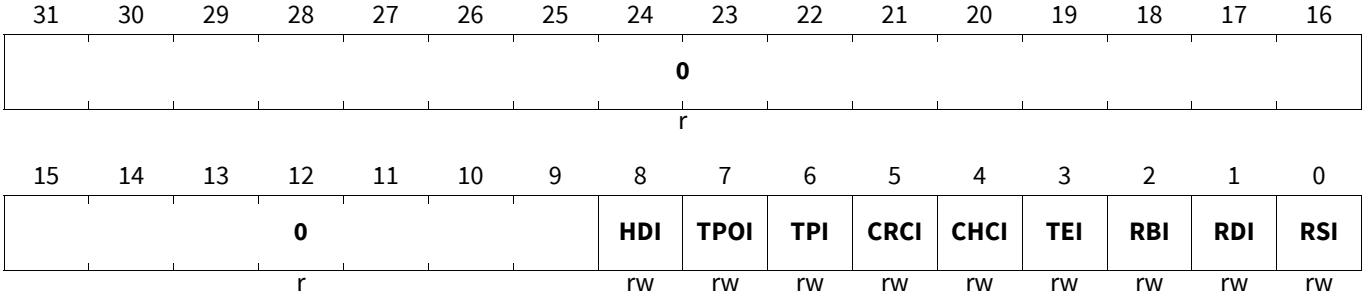
Interrupt Enable Register x

The Interrupt Enable Register INTEN x contain control bits that enable the interrupt source of any interrupt of PSI5-S channel x.

The Interrupt Status bits in register INTSTATx are set independently from the Interrupt Enable in Register INTENx.

INTENx (x=0-7)

Interrupt Enable Register x **($02C0_H$+x*4)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RSI	0	rw	**Enable Interrupt Request RSI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RDI	1	rw	**Enable Interrupt Request RDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RBI	2	rw	**Enable Interrupt Request RBI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TEI	3	rw	**Enable Interrupt Request TEI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
CHCI	4	rw	**Enable Interrupt Request CHCI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
CRCI	5	rw	**Enable Interrupt Request CRCI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TPI	6	rw	**Enable Interrupt Request TPI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TPOI	7	rw	**Enable Interrupt Request TPOI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
HDI	8	rw	**Enable Interrupt Request HDI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
0	31:9	r	**Reserved** Read as 0; should be written with 0.

Interrupt Node Pointer Register x

The Interrupt Node Pointer Register INPx contains the node pointers of PSI5-S channel x.

INPx (x=0-7)

Interrupt Node Pointer Register x **($02E0_H$+x*4)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					HDI			TPOI			TPI			CRCI	
r					rw			rw			rw			rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCI		CHCI			TEI			RBI			RDI			RSI	
rw		rw			rw			rw			rw			rw	

Field	Bits	Type	Description
RSI	2:0	rw	**Interrupt Node Pointer for Interrupt RSI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.RSI (if enabled by bit INTENx.RSI). 000_B Trigger Output TRIGO 0 is selected … 111_B Trigger Output TRIGO 7 is selected
RDI	5:3	rw	**Interrupt Node Pointer for Interrupt RDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.RDI (if enabled by bit INTENx.RDI). For bit field definition, see RSI.
RBI	8:6	rw	**Interrupt Node Pointer for Interrupt RBI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.RBI (if enabled by bit INTENx.RBI). For bit field definition, see RSI.
TEI	11:9	rw	**Interrupt Node Pointer for Interrupt TEI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.TEI (if enabled by bit INTENx.TEI). For bit field definition, see RSI.
CHCI	14:12	rw	**Interrupt Node Pointer for Interrupt CHCI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.CHCI (if enabled by bit INTENx.CHCI). For bit field definition, see RSI.
CRCI	17:15	rw	**Interrupt Node Pointer for Interrupt CRCI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.CRCI. For bit field definition, see RSI.
TPI	20:18	rw	**Interrupt Node Pointer for Interrupt TOI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.TPI (if enabled by bit INTENx.TPI). For bit field definition, see RSI.
TPOI	23:21	rw	**Interrupt Node Pointer for TPOI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.TPOI. For bit field definition, see RSI.

Field	Bits	Type	Description
HDI	26:24	rw	**Interrupt Node Pointer for HDI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATx.HDI. For bit field definition, see RSI.
0	31:27	r	**Reserved** Read as 0; should be written with 0.

Interrupt Overview Register

INTOV

Interrupt Overview Register **(0300$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**FOI**	**XCRCI**	**TBIR**	**EIR**	**RIR**	**TIR**	**HDI**	**TPOI**	**TPI**	**CRCI**	**CHCI**	**TEI**	**RBI**	**RDI**	**RSI**
r	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
RSI	0	rh	**Interrupt Pending on Node Pointer RSI** If any interrupt requested flag is set for this Node Pointer in register (INTSTATx or INTSTATG) AND the referring interrupt is enabled in (INTENx or INTENG) then this bit is set. It is automatically reset if all flags in INTSTATx/G are cleared for which the referring interrupt is enabled in INTENx/G.
RDI	1	rh	**Interrupt Pending on Node Pointer RDI** See details of INTOV.RSI.
RBI	2	rh	**Interrupt Pending on Node Pointer RBI** See details of INTOV.RSI.
TEI	3	rh	**Interrupt Pending on Node Pointer TEI** See details of INTOV.RSI.
CHCI	4	rh	**Interrupt Pending on Node Pointer CHCI** See details of INTOV.RSI.
CRCI	5	rh	**Interrupt Pending on Node Pointer CRCI** See details of INTOV.RSI.
TPI	6	rh	**Interrupt Pending on Node Pointer TPI** See details of INTOV.RSI.
TPOI	7	rh	**Interrupt Pending on Node Pointer TPOI** See details of INTOV.RSI.
HDI	8	rh	**Interrupt Pending on Node Pointer HDI** See details of INTOV.HDI.

Field	Bits	Type	Description
TIR	9	rh	**Interrupt Pending on Node Pointer TIR** If any interrupt requested flag is set for this Node Pointer in register INTSTATG AND the referring interrupt is enabled in INTENG then this bit is set. It is automatically reset if all flags in INTSTATG are cleared for which the referring interrupt is enabled in INTENG.
RIR	10	rh	**Interrupt Pending on Node Pointer RIR** See details of INTOV.TIR.
EIR	11	rh	**Interrupt Pending on Node Pointer EIR** See details of INTOV.TIR.
TBIR	12	rh	**Interrupt Pending on Node Pointer TBIR** See details of INTOV.TIR.
XCRCI	13	rh	**Interrupt Pending on Node Pointer XCRCI** See details of INTOV.TIR.
FOI	14	rh	**Interrupt Pending on Node Pointer FOI** See details of INTOV.TIR.
0	31:15	r	**Reserved** Read as 0.

Interrupt Status Register Global

The Interrupt Status Registers INTSTATG contains status bits that show the status of any global interrupt, i.e. of the ASC inside PSI5-S and XCRCI indicating a non recoverable message is received. On XCRCI, the non recoverable message is stored in ChID ‘0’, FID ’1’ with original IDs. INTSTATG contains as well FOI indicating a FIFO overrun condition. In a correct setup, this will never be set as the bandwidth of the ASC is assumed to be by far higher than the write bandwidth in order to have short delays.

The bits are set independently from the referring Interrupt Enable in Register INTENG. Thus they can be used as status bits as well e.g. by a SW based on polling.

INTSTATG

Interrupt Status Register Global **(0304_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										FOI	XCRCI	TBIR	EIR	RIR	TIR
r										rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
TIR	0	rh	**Transmit Interrupt Request Flag** This bit can be cleared by bit INTCLRG.TIR. This bit can be set by bit INTSETG.TIR. This bit is set independently from INTENG. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time

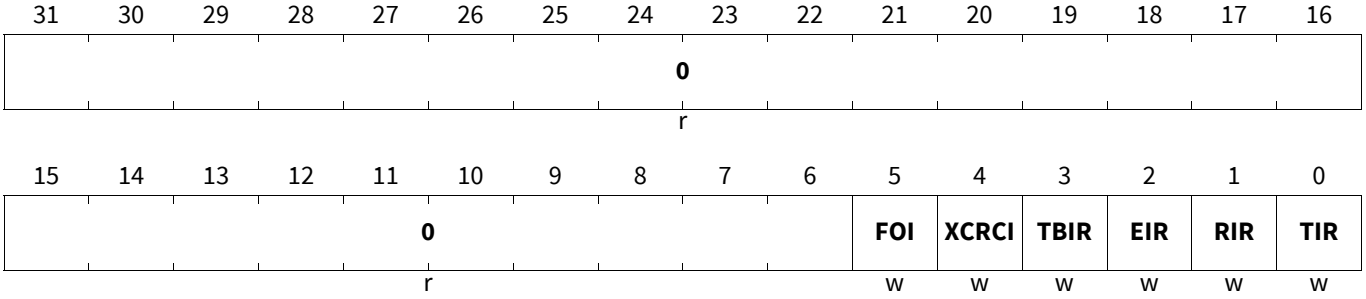
Field	Bits	Type	Description
RIR	1	rh	**Receive Interrupt Request Flag** This bit can be cleared by bit INTCLRG.RIR. This bit can be set by bit INTSETG.RIR. This bit is set independently from INTENG. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
EIR	2	rh	**Error Interrupt Request Flag** The cause of an error interrupt request EIR (framing, parity, overrun error) can be identified by the error status flags CON.FE, CON.PE, and CON.OE, This bit can be cleared by bit INTCLRG.EIR. This bit can be set by bit INTSETG.EIR. This bit is set independently from INTENG. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
TBIR	3	rh	**Transmit Buffer Interrupt Request Flag** This bit can be cleared by bit INTCLRG.TBIR. This bit can be set by bit INTSETG.TBIR. This bit is set independently from INTENG. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
XCRCI	4	rh	**XCRC Error Request Flag** This bit is set if the CRC check on the enveloping Packet Frame fails including the case where XCRC check can not be performed (non recoverable frames) see **Chapter 43.3.2.1.1**.The received data is not reliable and stored in ChID ‘0’, FID ’1’ with original IDs. This bit can be cleared by bit INTCLRG.XCRCI. This bit can be set by bit INTSETG.XCRCI. This bit is set independently from INTENG. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
FOI	5	rh	**FIFO Error Request Flag** This bit is set if the Transmit FIFO of the message generation is overrun i.e. a transfer to the FIFO was generated by the message generation unit or by CDW (CPU direct write register) while the FIFO was already full. This bit can be cleared by bit INTCLRG.FOI. This bit can be set by bit INTSETG.FOI. This bit is set independently from INTENG. 0_B No interrupt was requested since this bit was cleared the last time 1_B An interrupt was requested since this bit was cleared the last time
0	31:6	r	**Reserved** Read as 0.

Interrupt Set Register Global

The Interrupt Set Register INTSETG contains control bits that trigger an interrupt pulse for any interrupt of the ASC integrated in PSI5-S and for XCRCI and FOI.

INTSETG

Interrupt Set Register Global **(0308_H)** **Application Reset Value: $0000\ 0000_H$**



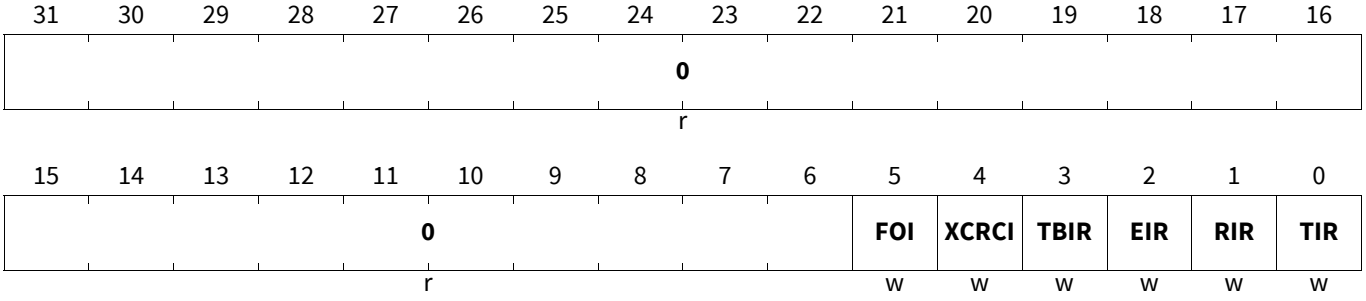
Field	Bits	Type	Description
TIR	0	w	**Set Interrupt Request Flag TIR** Setting this bit set bit INTSTATG.TIR. Clearing this bit has no effect. Reading this bit returns always zero.
RIR	1	w	**Set Interrupt Request Flag RIR** Setting this bit set bit INTSTATG.RIR. Clearing this bit has no effect. Reading this bit returns always zero.
EIR	2	w	**Set Interrupt Request Flag EIR** Setting this bit set bit INTSTATG.EIR. Clearing this bit has no effect. Reading this bit returns always zero.
TBIR	3	w	**Set Interrupt Request Flag TBIR** Setting this bit set bit INTSTATG.TBIR. Clearing this bit has no effect. Reading this bit returns always zero.
XCRCI	4	w	**Set Interrupt Request Flag XCRCI** Setting this bit set bit INTSTATG.XCRCI. Clearing this bit has no effect. Reading this bit returns always zero.
FOI	5	w	**Set Interrupt Request Flag FOI** Setting this bit set bit INTSTATG.FOI. Clearing this bit has no effect. Reading this bit returns always zero.
0	31:6	r	**Reserved** Read as 0; should be written with 0.

Interrupt Clear Register Global

The Interrupt Clear Register INTCLRG contain control bits that clear the status of any interrupt of the ASC integrated in PSI5-S and for XCRCI and FOI.

INTCLRG

Interrupt Clear Register Global **($030C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TIR	0	w	**Clear Interrupt Request Flag TIR** Setting this bit clears bit INTSTATG.TIR. Clearing this bit has no effect. Reading this bit returns always zero.
RIR	1	w	**Clear Interrupt Request Flag RIR** Setting this bit clears bit INTSTATG.RIR. Clearing this bit has no effect. Reading this bit returns always zero.
EIR	2	w	**Clear Interrupt Request Flag EIR** Setting this bit clears bit INTSTATG.EIR. Clearing this bit has no effect. Reading this bit returns always zero.
TBIR	3	w	**Clear Interrupt Request Flag TBIR** Setting this bit clears bit INTSTATG.TBIR. Clearing this bit has no effect. Reading this bit returns always zero.
XCRCI	4	w	**Clear Interrupt Request Flag XCRCI** Setting this bit clears bit INTSTATG.XCRCI. Clearing this bit has no effect. Reading this bit returns always zero.
FOI	5	w	**Clear Interrupt Request Flag FOI** Setting this bit clears bit INTSTATG.FOI. Clearing this bit has no effect. Reading this bit returns always zero.
0	31:6	r	**Reserved** Read as 0; should be written with 0.

Interrupt Enable Register Global

The Interrupt Enable Register INTENG contain control bits that enable the interrupt source of any interrupt of the ASC integrated in PSI5-S and for XCRCI and FOI.

The Interrupt Status bits in register INTSTATG are set independently from the Interrupt Enable in Register INTENG.

INTENG

Interrupt Enable Register Global **(0310_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										FOI	XCRCI	TBIR	EIR	RIR	TIR
r										rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TIR	0	rw	**Enable Interrupt Request TIR** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
RIR	1	rw	**Enable Interrupt Request RIR** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
EIR	2	rw	**Enable Interrupt Request EIR** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
TBIR	3	rw	**Enable Interrupt Request TBIR** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
XCRCI	4	rw	**Enable Interrupt Request XCRCI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
FOI	5	rw	**Enable Interrupt Request FOI** 0_B No interrupt request can be generated for this source 1_B An interrupt request can be generated for this source
0	31:6	r	**Reserved** Read as 0; should be written with 0.

Interrupt Node Pointer Register Global

The Interrupt Node Pointer Register INPG contains the node pointers of PSI5-S ASC and for XCRCI and FOI.

INPG

Interrupt Node Pointer Register Global **(0314_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0														FOI	
r														rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FOI	XCRCI			TBIR			EIR			RIR			TIR		
rw	rw			rw			rw			rw			rw		

Field	Bits	Type	Description
TIR	2:0	rw	**Interrupt Node Pointer for Interrupt TIR** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATG.TIR (if enabled by bit INTENG.TIR). 000_B Trigger Output TRIGO 0 is selected … 111_B Trigger Output TRIGO 7 is selected
RIR	5:3	rw	**Interrupt Node Pointer for Interrupt RIR** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATG.RIR (if enabled by bit INTENG.RIR). For bit field definition, see TIR.
EIR	8:6	rw	**Interrupt Node Pointer for Interrupt EIR** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATG.EIR (if enabled by bit INTENG.EIR). For bit field definition, see TIR.
TBIR	11:9	rw	**Interrupt Node Pointer for Interrupt TBIR** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATG.TBIR (if enabled by bit INTENG.TBIR). For bit field definition, see TIR.
XCRCI	14:12	rw	**Interrupt Node Pointer for Interrupt XCRCI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATG.XCRCI (if enabled by bit INTENG.XCRCI). For bit field definition, see TIR.
FOI	17:15	rw	**Interrupt Node Pointer for Interrupt FOI** This bit field defines the interrupt node, that is requested due to the set condition for bit INTSTATG.FOI (if enabled by bit INTENG.FOI). For bit field definition, see TIR.
0	31:18	r	**Reserved** Read as 0; should be written with 0.

OCDS Control and Status

The OCDS Control and Status (OCS) register is cleared by Debug Reset. The OCS register can only be written when the OCDS is enabled. If OCDS is being disabled, the OCS register value will not change. When OCDS is disabled the OCS suspend control is ineffective. Write access is 32 bit wide only and requires Supervisor Mode.

OCS

OCDS Control and Status (03CC$_H$) Debug Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		SUSSTA	SUS_P	SUS				0							
r		rh	w	rw				r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
SUS	27:24	rw	**OCDS Suspend Control** Controls the sensitivity to the suspend signal coming from the OCDS Trigger Switch (OTGS) *Note: Suspend disables the kernel clocks f_{PSI5-S} and the ASC clock f_{ASC}.* 0_H Will not suspend 1_H Hard suspend. Clock is switched off immediately. 2_H Soft suspend option A (Suspends after end of current send or receive transfers, if any are ongoing) **others**, reserved (will not suspend)
SUS_P	28	w	**SUS Write Protection** SUS is only written when SUS_P is 1, otherwise unchanged. Read as 0.
SUSSTA	29	rh	**Suspend State** 0_B Module is not (yet) suspended 1_B Module is suspended
0	23:0, 31:30	r	**Reserved** Read as 0; must be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, ... , EN31 -> TAG ID 011111B.

ACCEN0

Access Enable Register 0 ($03D0_H$) Application Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

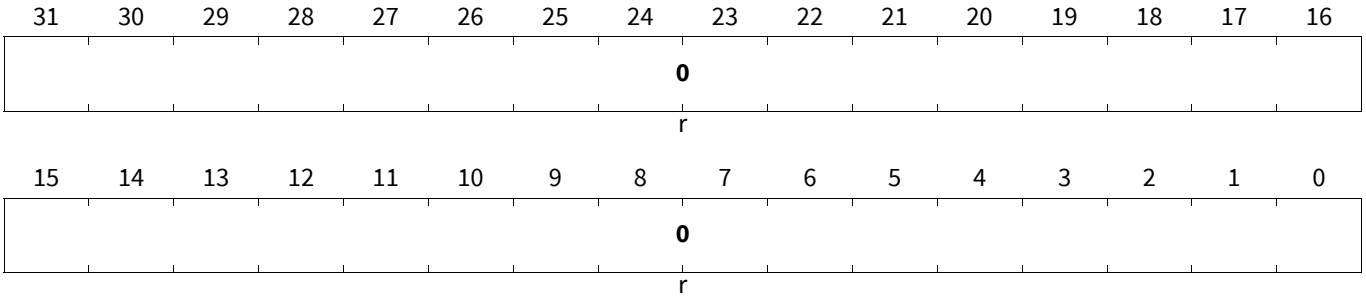
Access Enable Register 1

The Access Enable Register 1 controls write access[1)] for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the AURIX devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, ... , EN31 -> TAG ID 111111B.

ACCEN1

Access Enable Register 1 **($03D4_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

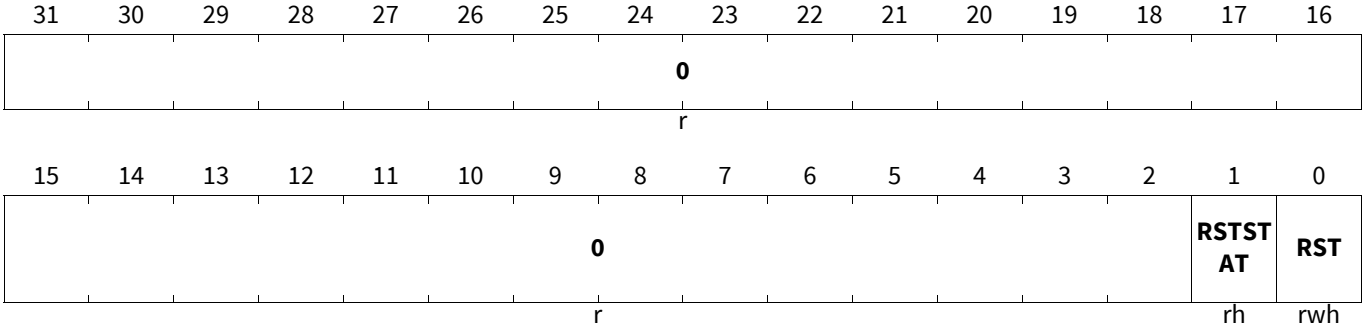
Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCL.CLR register bit.

During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **(03D8$_H$)** **Application Reset Value: 0000 0000$_H$**

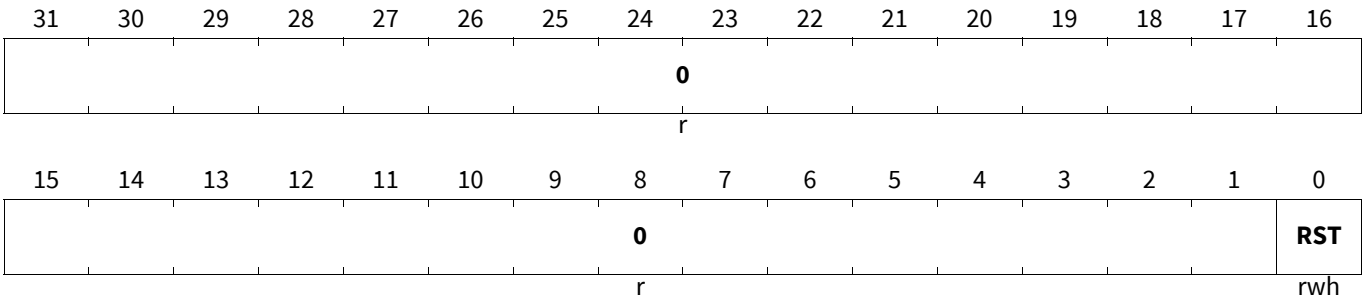


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to the related module kernel. Kernel registers related to the Debug Reset are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **(03DC$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register (03E0$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															CLR
r															w

Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

43.5 IO Interfaces

Table 435 List of PSI5S Interface Signals

Interface Signals	I/O	Description
sx_fpi		**FPI slave interface**
sx_irq_psi5s		
sx_gtm2psi5s		
TRIGO(7:0)	out	**PSI5-S Service Request**
TRIG(7:0)	in	**GTM timer output vector**

Table 435 List of PSI5S Interface Signals (cont'd)

Interface Signals	I/O	Description
CLK	out	**PSI5S CLK is a clock that can be used on a pin to drive the external PHY.** PSISCLK is a clock that can be used on a pin to drive the external PHY. It can be programmed in register FDO.
RXA	in	**RX data input** This is the receive (RX) input signal. PSI5S_IOCR.ALTI controls which pad is used as input to this signal.
RXB		
RXC		
RXD		
TX	out	**TX data output** This is the transmit (TX) output signal. The registers in the port group control at which actual port pin this signal used.

43.6 Revision History

Table 436 Revision History

Reference	Change to Previous Version	Comment
V1.12.10		
Page 98	Clean up revision history.	

44 Gigabit Ethernet MAC (GETH)

This chapter describes the Ethernet MAC [1)2)].

1) Excerpted portions are Synopsys Proprietary. Used with permission.

2) Module is not available in some variants. In these variants registers are still accessible but functionality cannot be guaranteed.

44.1 Feature List

The detailes feature list can be found in **Chapter 44.2.4**.

44.1.1 Delta to AURIX™ TC2xx

GETH was not implemented in AURIX™ TC2xx.

44.2 Overview

44.2.1 Basic Structure Diagram

Figure 691 shows the basic structure of the module:

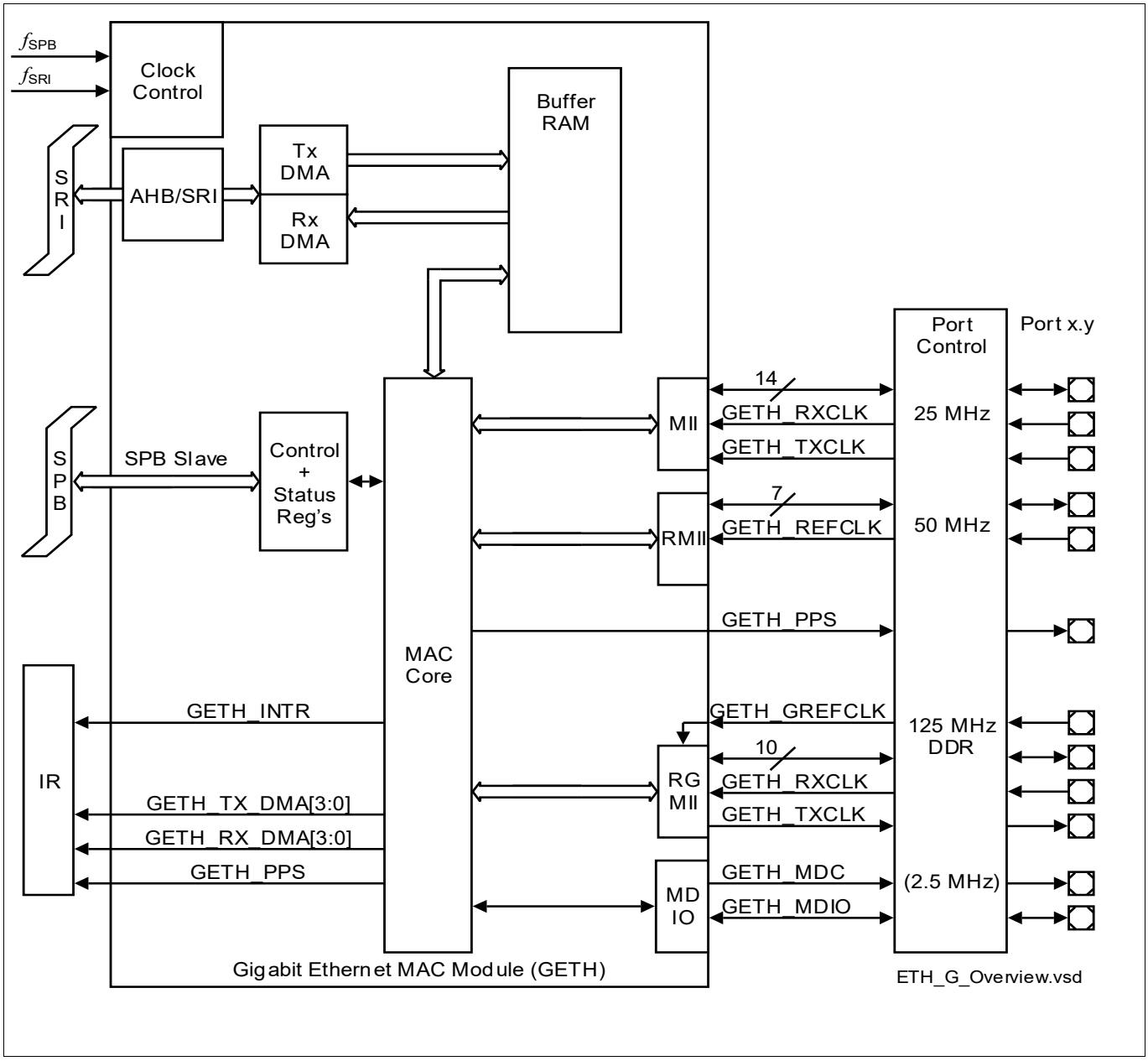


Figure 691 Ethernet MAC Module Overview

44.2.2 DWC_ether_qos General Product Description

The DWC_ether_qos enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008.

As shown in **Figure 692**, one AHB or AXI Master interface is connected to all DMA channels. The DMA arbiter helps in arbitration of all the paths (Transmit and Receive) in all channels. Each channel has a separate set of Control

and Status registers (CSR) for managing the Transmit and Receive functions, descriptor handling, and interrupt handling.

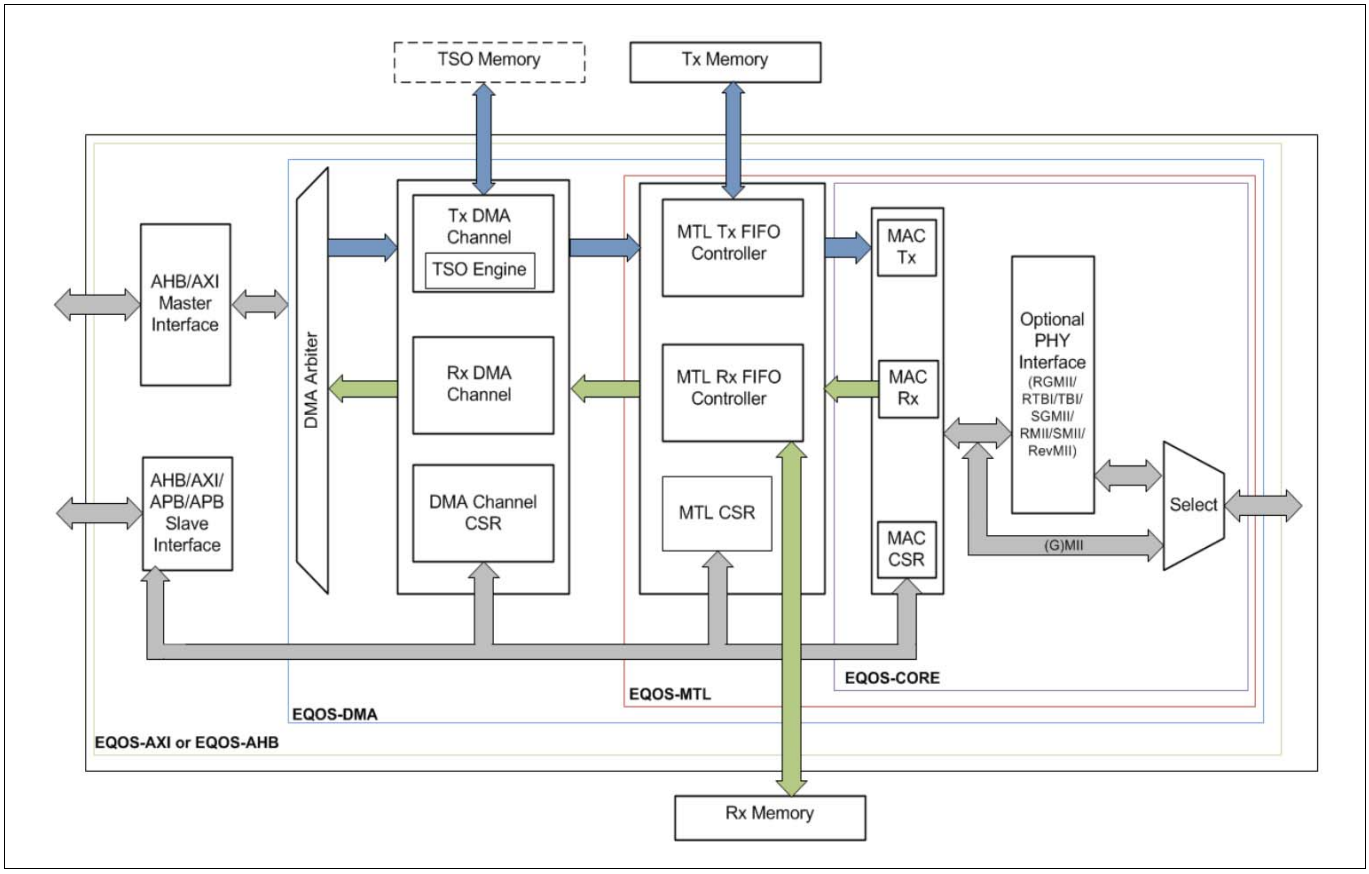


Figure 692 System-Level Block Diagram

Note: When you configure the MAC for a single PHY interface, the MUX logic is removed and the corresponding PHY interface is directly connected to the ports.

44.2.3 DWC_ether_qos Interfaces

The DWC_ether_qos supports several options to interface to the application: to the MAC through native CORE, MTL, or DMA interfaces; or through AHB or AXI interfaces. The DWC_ether_qos also supports a variety of PHY interfaces, as well.

The DWC_ether_qos controller supports the following interfaces:

- **AHB Interface**
  The AHB interface is designed to integrate with AMBA High-Performance Bus (AHB) on the application side. The AHB interface transfers the data to and from system memory through the AHB master interface. When the AHB interface is selected, the host CPU uses the default 32-bit AHB slave interface to access the Control and Status registers (CSRs) of the DWC_ether_qos subsystem.
- **PHY Interfaces**
  The DWC_ether_qos supports any one or a combination of the following PHY interfaces:
  - Media Independent Interface (MII) [Default]
  - Reduced GMII (RGMII)
  - Reduced MII (RMII)

44.2.4 DWC_ether_qos Features

44.2.4.1 Standard Compliance

The DWC_ether_qos supports defaults interfaces defined in the IEEE 802.3 specifications and several industry standard to the PHY.

In addition to the default interfaces defined in the IEEE 802.3 specifications, the DWC_ether_qos supports several industry standard interfaces to the PHY. The DWC_ether_qos is compliant with the following standards:

- IEEE 802.3-2008 for Ethernet MAC, Media Independent Interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization
- IEEE 802.1AS-2011 and 802.1-Qav-2009 for Audio Video (AV) traffic
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- AMBA 2.0 for AHB master, AHB slave, and APB slave ports
- RGMII/RTBI specification version 2.6 from HP/Marvell
- RMII specification version 1.2 from RMII consortium

44.2.4.2 MAC Features

44.2.4.2.1 MAC Tx and Rx Common Features

The DWC_ether_qos controller supports a number common features of the MAC Tx and Rx.

The DWC_ether_qos controller supports the following features common to both MAC Tx and Rx:

- Separate transmission, reception, and control interfaces to the application
- Configurable big-endian and little-endian mode for Transmit and Receive paths
- 10, 100, and 1000 Mbps data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant interface to communicate with an external Gigabit or Fast Ethernet PHY
  - RGMII interface (optional) to communicate with an external gigabit PHY
  - RMII interface (optional) to communicate with an external Fast Ethernet PHY
- Half-duplex operation:
  - CSMA/CD Protocol support
  - Flow control using backpressure support (based on implementation-specific white papers and UNH Ethernet Clause 4 MAC Test Suite - Annex D)
  - Packet bursting and packet extension in 1000 Mbps half-duplex operation
- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in (G)MII and Reduced Gigabit Media Independent Interface (RGMII) PHYs.
- 64-bit data transfer interface on the application side
- Data center bridging (DCB) is not supported
- Full-duplex flow control operations (IEEE 802.3x Pause packets, Priority flow control (PFC) is not supported)
- Network statistics with RMON or MIB Counters (RFC2819/RFC2665)
- Module to support Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping is supported in TX direction.

- Flexibility to control the Pulse-Per-Second (PPS) output signal (ptp_pps_o)
- Optional MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management
- Programmable control to clear interrupt status bits on writing 1 to it
- Programmable control to provide Slave error response for accesses to the reserved registers within the CSR address space

44.2.4.2.2 MAC Tx Features

The DWC_ether_qos controller supports a number of MAC Tx features.

The DWC_ether_qos supports the following MAC Tx features

- Preamble and start of packet data (SFD) insertion
- Separate 32-bit status for each packet transmitted from the application
- Automatic CRC and pad generation controllable on a per-packet basis
- Programmable packet length to support Standard or Jumbo Ethernet packets with up to 9 KB of size
- Programmable Inter Packet Gap (40–96 bit times in steps of 8)
- IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in full-duplex mode)
- Option to transmit packets with reduced preamble size in full-duplex mode
- Insert, replace, or delete queue/channel-based VLAN tags

44.2.4.2.3 MAC Rx Features

The DWC_ether_qos controller supports a number of MAC Rx features.

The DWC_ether_qos controller supports the following MAC Rx features:

- Automatic Pad and CRC Stripping options
- Option to disable Automatic CRC checking
- Preamble and SFD deletion
- Separate 112-bit or 128-bit status
- Programmable watchdog timeout limit
- Flexible address filtering modes:
  - Up to 31 additional 48-bit perfect (DA) address filters with masks for each byte
  - Up to 31 48-bit SA address comparison check with masks for each byte
  - Option to pass all multi-cast addressed packets
  - Promiscuous mode to pass all packets without any filtering for network monitoring
  - Pass all incoming packets (as per filter) with a status report
- Additional packet filtering:
  - VLAN tag-based: Perfect match and Hash-based (optional) filtering. Filtering based on either outer or inner VLAN tag is possible.
  - Extended VLAN tag based filtering 8 filter selection
- IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets
- Optional module to detect remote wake-up packets and AMD magic packets
- Optional forwarding of received Pause packets to the application (in full-duplex mode)
- Optional Receive module for Layer 3/Layer 4 checksum offload for received packets
- Optional stripping of up to two VLAN Tags and providing the tags in the status.

44.2.4.3 Transaction Layer (MTL) Features

The DWC_ether_qos supports a number of Tx and Rx Transaction Layer (MTL) features.

The MTL block consists of the following FIFOs: Tx FIFO and Rx FIFO. The FIFO size is 4kB for Tx and 8 kB for Rx . The FIFO space is shared by multiple queues (up to 4 Tx and up to 4 Rx queues). You can configure the buffer size for each queue in multiples of 256 bytes. The MTL block supports the following features:

The MTL includes the following feature groups:

- MTL Tx and Rx Common Features
- MTL Tx Features
- MTL Rx Features

44.2.4.3.1 MTL Tx and Rx Common Features

The DWC_ether_qos supports a number of common features of the Tx and Rx Transaction Layer (MTL).

The DWC_ether_qos controller supports the following common features of the MTL Tx and Rx:

- 32-bitTransaction Layer block (bridges the application and the MAC)
- Data transfers executed using simple FIFO protocol
- Synchronization for all clocks in the design (Transmit, Receive, and Application clocks)
- Optimization for packet-oriented transfers with packets delimiters
- Two dual-port RAM-based synchronous FIFO controllers
- RAM memory instantiation outside the top-level module to facilitate memory testing or instantiation
- Programmable burst length, up to half the size of the MTL Rx queue or Tx queue size, to support burst data transfer in the EQOS-MTL configuration
- Programmable threshold capability for each queue (default of 64 bytes)
- Optional Debug and slave mode operation on Queue 0 (default queue)

44.2.4.3.2 MTL Tx Features

The DWC_ether_qos supports a number of Tx Transaction Layer (MTL) features.

The DWC_ether_qos controller supports the following MTL Tx features:

- Following FIFO sizes on transmission: 256 byte, 512 byte, 1 KB, 2 KB, 4 KB
- Multiple queues (up to 4) on the Transmit path with a common memory for all Tx queues
- Store-and-Forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Programmable queue size in configurations with multiple queues. Each queue size can be programmed in terms of 256 bytes
- Automatic retransmission of collision packets in half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Disabling of Data Memory RAM chip-select when inactive to reduce power consumption
- Optional module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum
- Programmable interrupt options for different operational conditions
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Optional statistics related to bandwidth consumption by each queue of up to 16 blocks over a 125 µ's period
- Optional packet-level control for
  - VLAN tag insertion or replacement

  - Ethernet source address insertion
  - Layer3/Layer4 Checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control
- Following scheduling algorithms in configurations with multiple queues:
  - Weighted Round Robin (WRR)
  - Strict Priority (SP)
- Option to support dropping of Tx Status to improve the Transmit throughput

44.2.4.3.3 MTL Rx Features

The DWC_ether_qos supports a number of Rx Transaction Layer (MTL) features.

The DWC_ether_qos controller supports the following Rx MTL features:

- Following Rx queue sizes in the Receive path: 256 byte, 512 byte, 1 KB, 2 KB, 4 KB, 8 KB
- Multiple queues (up to 4) on the Receive path with a common memory for all Rx queues
- Programmable Rx queue threshold (default fixed at 64 bytes) in Threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of Pause packet control or backpressure signal to the MAC based on the Rx Queue fill level
- Arbitration among queues when multiple queues are present. The following arbitration schemes are supported:
  - Weighted Round Robin (WRR)
  - Weighted Strict priority (WSP)
  - Strict Priority (SP)

44.2.4.4 DMA Block Features

The DWC_ether_qos controller support a variety of DMA block features.

The DMA block exchanges data between the MTL block and system memory. The well defined descriptors structure acts as a software and hardware interface. The application can use a set of registers (DMA CSR) to control the DMA operations. The DMA block supports the following features:

- 32-bit data transfers
- Multi-channel Transmit and Receive engines (up to 4 Transmit channels; up to 4 Receive channels)
- Separate DMA channel in the Transmit path for each queue in MTL
- Single or multiple DMA channels for any number of queues in MTL Receive path
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture to allow large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 KB of data)

- Comprehensive status reporting for normal operation and transfers with errors
- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-packet Transmit or Receive Complete Interrupt control
- Round-robin or fixed-priority arbitration between the Receive and Transmit engines
- Start and Stop modes
- Separate ports for host CSR access and host data interface
- Routing of received packets to the DMA channels based on the DA or VLAN Priority in multi-channel DMA configurations
- Option to split the packet header (Layer 3 and Layer 4) and payload in a different buffers
- Time-sensitive conditional packet transmission by comparing the Slot Time information provided in the descriptor (useful for AV applications)
- Programmable control for Transmit Descriptor posted writes to improve the throughput

44.2.4.5 AHB Interface Features

44.2.4.5.1 AHB Master Interface Features

The DWC_ether_qos controller supports the AHB Master interface as the application interface.

The AHB master interface supports the following features:

- Interfaces with the application through AHB
- Little-endian and big-endian modes
- 32-bitdata on the AHB master port
- Option to select address-aligned bursts from AHB master port
- Split, Retry, and Error AHB responses
- AHB 1K boundary burst splitting
- Software-selected type of AHB burst (fixed burst, indefinite burst, or mix of both)

The AHB master interface does not generate the following:

- Wrap burst
- Locked or Protected transfers

44.2.4.5.2 AHB Slave Interface Features

The DWC_ether_qos controller supports the AHB Slave interface as the CSR interface.

The AHB slave interface supports the following features:

- Interfaces with the application through AHB
- Little-endian and big-endian modes
- AHB slave interface (32-bit) for CSR access in which only 32-bit or less (byte, half-word) accesses are possible
- All AHB burst types

The AHB slave interface does not generate the following responses:

- Split
- Retry
- Error

44.2.4.6 Audio and Video Features

DWC_ether_qos can be used in Audio Video (AV) mode, and the supported features are compliant to the industry standards for AV traffic.

Note: Data center bridging (DCB) and Priority Flow Control (PFC) are not supported.

DWC_ether_qos supports the following Audio Video (AV) features:

- Separate channels or queues for AV data transfer in 100 Mbps and 1000 Mbps modes
- IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm for Transmit channels
- Single Tx FIFO and Rx FIFO (MTL) for all selected queues (system-side interface [AHB] remains the same)
- Programmable Slot Interval with range from 1µ to 4096µs and granularity of 1µs.

44.2.4.7 Generic Queuing Support

- Programmable control for routing Receive packets with Multicast/Broadcast destination address to a programmable Receive Queue. Support routing of Untagged Receive packets to a programmable Receive Queue. Programmable control for routing VLAN tagged and untagged IEEE 1588 PTP over Ethernet Receive packets to a same or separate programmable Receive Queue. Programmable control for routing Unicast/Multicast Receive packets that fail the destination address filter to a separate programmable Receive Queues.

44.2.4.8 Monitor, Testing, and Debugging Features

The monitoring, testing, and debugging mechanism in DWC_ether_qos help in effective analysis of the configured features.

DWC_ether_qos supports the following features for monitoring, testing, and debugging:

- Internal loopback from Tx to Rx on the GMII or MII for debugging
- DMA states (Tx and Rx) as status bits
- Debug status register that gives status of FSMs in Transmit and Receive data paths and FIFO fill-levels
- Application Abort status bits
- MMC (RMON) module
- Current Tx or Rx Buffer pointer as status registers
- Current Tx or Rx Descriptor pointer as status registers
- Statistical counters to calculate the bandwidth served by each Transmit channel when AV or DCB support is enabled
- Tx or Rx Queues memory accessible through Slave port for debug

44.3 Functional Description

44.3.1 Architecture

This chapter describes the DWC_ether_qos interfaces, protocols, functionality, and implementation.

44.3.1.1 CSR Slave Interface

The CSR slave interface provides access to the CSR space.

By default, all the slave interface will provide OKAY response to all CSR accesses. If SEEN bit ofMAC_CSR_SW_Ctrl register is programmed to 1, DWC_ether_qos will provide error response whenreserved registers within the CSR space are accessed.

44.3.1.1.1 AHB Slave Interface

The AHB slave interface communicates with an application using the AHB interface. When the AHB interface is selected, the host CPU uses the default 32-bit AHB slave interface to access the Control and Status Registers (CSRs) of the DWC_ether_qos subsystem.

The 32-bit AHB slave interface provides access to the DMA, MTL, and MAC CSR space. The AHB slave interface supports the following features:

- Fully AMBA 2.0 Compliant AHB slave—no restrictions
- Single and all burst transfers
- Busy and early terminations
- 32-bit, 16-bit, and 8-bit write or read transfers to the CSRThe 32-bit CSR access is recommended to avoid any software synchronization problems.
- OKAY response:
  The SPLIT, RETRY, or ERROR responses are not supported. When a reserved address is read in MAC register space, it gives all zeros data and generates an OKAY response (When SEEN bit is set, MAC generates ERROR response). Similarly, when a reserved address is written, the ignores the write request and generates an OKAY response (When SEEN bit is set, MAC generates ERROR response).

44.3.1.2 Application Master Interface

44.3.1.2.1 AHB Master Interface

The AHB interface is designed to integrate with AMBA High-Performance Bus (AHB) on the application side. The AHB interface transfers the data to and from system memory through AHB master interface.

In the EQOS-AHB configuration, the DMA controller interfaces with the application through the AMBA AHB interface. The AHB master interface controls the data transfers and the AHB slave interface accesses the CSR space.

AHB Master Interface Features

The AHB master interface supports several features.

The AHB master interface converts the internal DMA request cycles into AHB cycles. The AHB master interface has the following features:

- The AHB master interface is AMBA 2.0-compliant AHB master with no restrictions
- The AHB master interface supports the following burst length modes: fixed, unspecified, and mixed. These modes are explained in the Burst Length Modes section.
- The AHB slaves interfacing with the AHB master support the SINGLE and INCR transfers.

- The AHB master interface can handle the AHB SPLIT, RETRY, and ERROR conditions. Any ERROR response halts all further transactions for that DMA and indicates the error as fatal through CSR and interrupt. The application can choose to give a hard or soft reset to the module to restart the operation.
- The AHB master interface can handle the AHB 1K boundary breaking.
- The AHB master interface can align all AHB burst transfers to an address value. For more information, see Aligning AHB Burst Transfers to An Address Value.
- The DMA requests an AHB Burst Read transfer only when it can completely accept the received burst data. The data read from the AHB is always pushed into the DMA without any delay or Busy cycles. However, when multiple channels are selected, there may be some delay and Busy cycles may be inserted because of the shared-memory architecture for the channels.
- The DMA requests an AHB Burst Write transfer only when it has the sufficient data to completely transfer the burst. The AHB interface always assumes that it has the data available to push into the AHB bus. However, the DMA can prematurely indicate end-of-valid data (because of the transfer of EOP of an Ethernet packet) during the burst.
  In Fixed Burst Length mode, the AHB master interface continues the burst with dummy data until the specified length is completed. In INCR mode, it ends the burst transfer prematurely.

Burst Length Modes

The AHB master interface supports the following burst length modes:

- Fixed Burst Length
  In fixed burst length mode, the AHB master always initiates a burst with SINGLE, INCR4, INCR8, or INCR16 type. However, when such a burst is responded with SPLIT, RETRY, or Early Burst Termination (EBT), by default, the AHB master again initiates the pending transfers of the burst with INCR or SINGLE burst length type. When only one beat is remaining, the AHB master performs a SINGLE transfer. If more than one beat is remaining, the AHB master performs an INCR transfer. The AHB master terminates such INCR bursts when remaining beats of the original requested fixed burst are transferred.
  To ensure that the AHB master always performs the INCRx and SINGLE transfers even when a part of the burst is initiated again after a SPLIT, RETRY, or EBT response, use the RIB bit of the DMA_Mode register. However, when you enable this mode, the pending burst may further get split into multiple transfers as per the following rules before any new transfer is initiated:
  - If the DMA requests a burst transfer that is not equal to INCR4, INCR8, or INCR16, the AHB application interface splits the transfer into multiple burst transactions. For example, if the DMA requests a 15-beat burst transfer, the AHB interface splits it into multiple transfers of INCR8 and INCR4, and three SINGLE transactions.
  - If the burst length is more than 16, the AHB master interface splits it into multiple burst transfers. For example, if the burst length is 32 and the DMA starts a 32-beat transfer, the AHB master splits the burst into two INCR16.
- Unspecified Burst Length
  In unspecified burst mode, the AHB master always initiates a transfer with INCR and completes the DMA requested burst in one transfer.
- Mixed Burst Mode
  In mixed burst mode, the AHB master always initiates the bursts with fixed-size (INCRx) when the DMA requests transfers of size less than or equal to 16 beats. When the DMA requests bursts of length more than 16, the AHB master initiates such transfers with INCR and completes the request in one transfer.
  To ensure that the pending burst of a transfer is rebuilt only with INCRx transfers when an INCRx transfer is interrupted by RETRY, SPLIT, or EBT response, use the RB bit of the DMA_SysBus_Mode register. However, if an INCR burst initiated in this mode is interrupted, the AHB master rebuilds the burst only with INCR transfers.
  You can choose the burst modes by programming the FB bit in the DMA_SysBus_Mode register.

Note: The AHB master does not generate the WRAP bursts, and locked or protected transfers. Therefore, the DWC_ether_qos does not have the HLOCK or HPROT output signals.

Aligning AHB Burst Transfers to An Address Value

The AHB master interface can align all AHB burst transfers to an address value.

You can enable aligning all AHB burst transfers to an address value, by using the AAL bit in the DMA_SysBus_Mode register. If Bit 0 and Bit 12 of DMA_SysBus_Mode register are set to1, the AHB interface and the DMA ensure that all initiated beats are aligned to the address, completing the packet transfer in the minimum number of required beats.

44.3.1.3 DMA Controller

The DMA controller transfers the data packets received by the MAC to the Rx Buffer in system memory and Tx data packets from the Tx Buffer in the system memory. The DWC_ether_controller can be programmed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal and error conditions.

The DMA has independent Transmit (Tx) and Receive (Rx) engines, and a CSR space. The Tx engine transfers data from the system memory to the device port (MTL), whereas the Rx engine transfers data from the device port to the system memory. The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programmed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal or error conditions.

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx Buffer in system memory and Tx data packets from the Tx Buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The DMA supports up to 4 Tx and 4 Rx Descriptor lists (or DMA channels). The base address of each list is written to the respective Tx Descriptor List Address register and Rx Descriptor List Address register. The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. You can control the offset is controlled by the DSL field of DMA_Ch[n]_Control register. The number of descriptors in the list is programmed in the respective Tx (or Rx) Descriptor Ring Length register. Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List Address register to create a descriptor ring.

The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet but cannot exceed a single packet. Buffers contain only data. Buffer status is maintained in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

The DWC_ether_qos supports the ring structure for the DMA descriptor. For more information on the descriptor, see “Descriptors” that describes the descriptor structure and how the DMA accesses the descriptors.

44.3.1.3.1 DMA Arbiter (EQOS-DMA and EQOS-AHB Configurations)

The arbiter inside the DMA module performs the arbitration between the Tx and Rx channel accesses to the AHB master interface.

The arbiter inside the DMA module performs the arbitration between the Tx and Rx channel accesses to the AHB master interface. The following two types of arbitrations are supported:

- Round-Robin Arbitration
  When Bit 1 of the DMA_Mode register is reset and both Tx and Rx DMAs are simultaneously requesting for access, the arbiter allocates the data bus in ratio set by Bits[14:12] of DMA_Mode register.
- Fixed-Priority Arbitration
  When Bit 1 of the DMA_Mode register is set, the Rx DMA always gets priority over the Tx DMA for data access by default. When Bit 11 of DMA_Mode Register is also set, the Tx DMA gets priority over the Rx DMA as explained in the following table.

Note: The Rx DMA places the next request only after the earlier request is complete. There is delay to find data availability in the MTL queue (ari_rdy and ari_pbl is enabled, if data equal to PBL is available; MTL asserts ari_rxwatermark which indicates required data availability) and place the request. So, there is no overlap of Rx DMA request with Tx DMA request when operating in single DMA mode. However, this is possible with multiple Rx DMA enabled.

Table 437 Priority Scheme for Tx DMA and Rx DMA

Bit 14	Bit 13	Bit 12	Bit 11	Bit 1	Priority Scheme
x	x	x	0	1	Rx always has priority over Tx
0	0	0	0	0	Tx and Rx have equal priority. Rx gets the access first on simultaneous requests.
0	0	1	0	0	Rx has priority over Tx in ratio 2:1.
0	1	0	0	0	Rx has priority over Tx in ratio 3:1.
0	1	1	0	0	Rx has priority over Tx in ratio 4:1.
1	0	0	0	0	Rx has priority over Tx in ratio 5:1.
1	0	1	0	0	Rx has priority over Tx in ratio 6:1.
1	1	0	0	0	Rx has priority over Tx in ratio 7:1.
1	1	1	0	0	Rx has priority over Tx in ratio 8:1.
x	x	x	1	1	Tx always has priority over Rx.
0	0	0	1	0	Tx and Rx have equal priority. Tx gets the access first on simultaneous requests.
0	0	1	1	0	Tx has priority over Rx in ratio 2:1.
0	1	0	1	0	Tx has priority over Rx in ratio 3:1.
0	1	1	1	0	Tx has priority over Rx in ratio 4:1.
1	0	0	1	0	Tx has priority over Rx in ratio 5:1.
1	0	1	1	0	Tx has priority over Rx in ratio 6:1.
1	1	0	1	0	Tx has priority over Rx in ratio 7:1.
1	1	1	1	0	Tx has priority over Rx in ratio 8:1.

44.3.1.3.2 DMA Transmit Operation

DMA Transmit Operation: Default (Non-OSP) Mode

When the Transmit (Tx) DMA engine transfers data from the system memory to the device port (MTL), it can operate in the default (non-OSP) mode.

The Tx DMA engine in default mode proceeds as follows:

1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application advances the Descriptor Tail pointer offset value of the Transmit Channel.
3. While in the Run state, the DMA runs an Arbitration cycle to select the next Tx DMA channel from which the packets requiring transmission should be processed.
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended and Bit 2 and Bit 16 of Status Register of corresponding DMA channel are set and the Tx Engine proceeds to step 11:
   a) The descriptor is flagged as owned by the application (TDES3 [31] = 1'b0)
   b) The Descriptor Tail pointer is equal to the Current Descriptor pointer in Ring Descriptor list Mode
   c) An error condition occurs
6. If the acquired descriptor is flagged as owned by the DMA (TDES3[31] = 1’b1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
7. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission.
8. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end of- Ethernet-packet data is transferred to the MTL.
9. When packet transmission is complete, if IEEE 1588 Timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step.If timestamp feature is not enabled for this packet, the DMA does not alter the contents of TDES0 and TDES1.
10. Bit 0 of Status Register of corresponding channel is set after completing transmission of a packet that has Interrupt on Completion (TDES2[31]) set in its Last Descriptor. The DMA engine returns to step 3.
11. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the DMA_CH0_TxDesc_Tail_Pointer when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the application stopped the DMA by clearing Bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.

The Tx DMA transmission flow in default mode is shown in **Figure 693**.

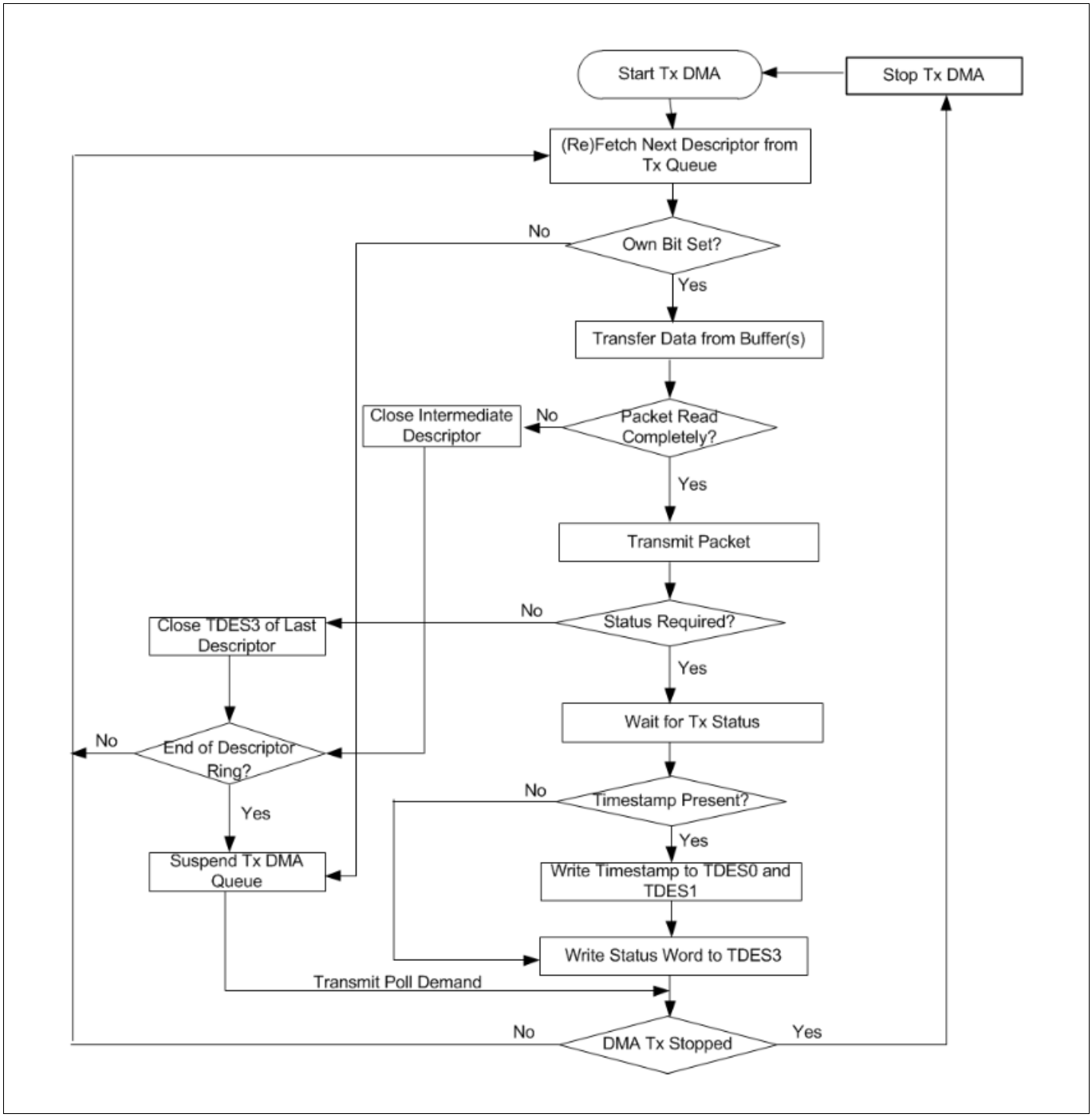


Figure 693 Tx DMA Operation in Default Mode

DMA Transmit Operation: OSP Mode

When the Transmit (Tx) DMA engine transfers data from the system memory to the device port (MTL), it can operate in the OSP mode.

In the Run state, if Bit 4 is set in the Transmit Control Register of corresponding DMA channel, the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. As the Transmit process finishes transferring the first packet, it immediately polls the Transmit Descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.

In OSP mode, the Run state Tx DMA operates in the following sequence:

1. The DMA operates as described in step 1 - step 7 of the Tx DMA (default mode).
2. The DMA fetches the next descriptor without closing the last descriptor of previous packet.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and skips to step 7.
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES1, thus closing the descriptor.
   If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES2 and TDES3.
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA does the following:If no status is pending and the application stopped the DMA by clearing Bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.
   a) Writes the timestamp (if enabled for the current packet) to TDES0 and TDES1
   b) Writes the status to the corresponding TDES1
   c) Sets relevant interrupts and returns to Suspend mode

   Note: If no status is pending and the application stopped the DMA by clearing Bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.

8. The DMA can exit Suspend mode and enter the Run state (goes to step 1 or step 2 depending on pending status) only after receiving a Transmit Poll demand in Transmit Descriptor Tail Pointer register of corresponding channel.

   Note: The DMA fetches the next descriptor before closing the current descriptor. Therefore, the descriptor ring length must be more than two. Synopsys recommends having a minimum descriptor length of four.

The basic flow is described in **Figure 694**.

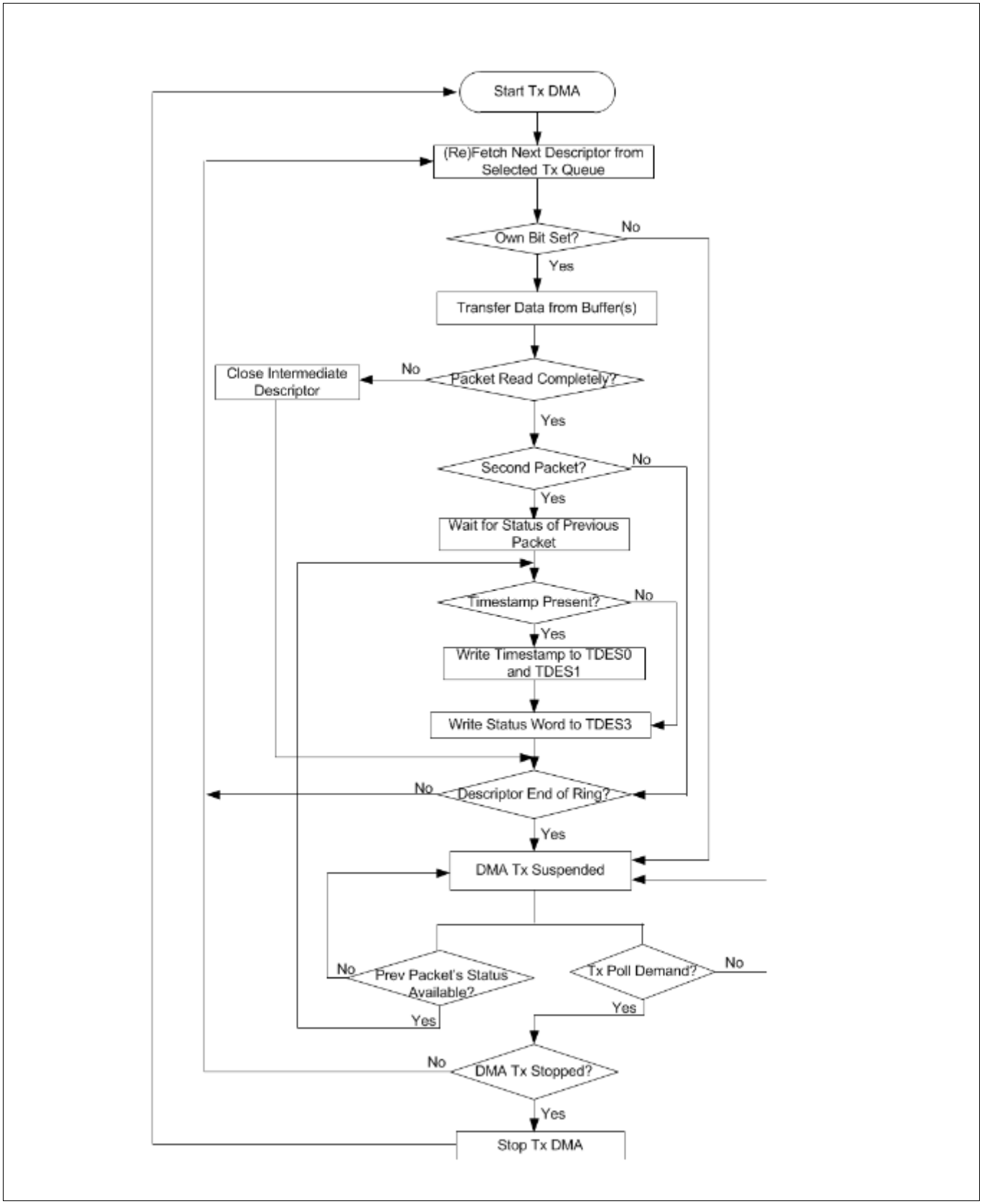


Figure 694 Tx DMA Operation in OSP Mode

Transmit Packet Processing

It is important to understand how the Tx DMA processes packets.

The Tx DMA expects that the data buffers contain complete Ethernet packets, excluding preamble, pad bytes, and FCS fields. The DA, SA, and Type/Length fields contain valid data. If the Tx Descriptor indicates that the MAC must disable CRC or PAD insertion, the buffer must have complete Ethernet packets (excluding preamble), including the CRC bytes.

Packets can be data-chained and can span several buffers. Packets must be delimited by the First Descriptor (TDES3[29]) and the Last Descriptor (TDES3[28]). As transmission starts, the First Descriptor must have TDES3[29] set. When this occurs, the packet data is transferred from the application buffer to the MTL Tx Queue. Concurrently, if the current packet has the Last Descriptor (TDES3[28]) clear, the Tx Process attempts to acquire the Next Descriptor. The Tx Process expects this descriptor to have TDES3[29] clear. If TDES3[28] is clear, it indicates an intermediary buffer. If TDES3[28] is set, it indicates the last buffer of the packet.

After the last buffer of the packet has been transmitted, the DMA writes back the final status information to the Transmit Descriptor 3 (TDES3) word of the descriptor that has the Last Descriptor Bit set in Transmit Descriptor 3 (TDES3[28]). At this time, if Interrupt on Completion (TDES2[31]) is set, Bit 0 of Status Register of corresponding DMA channel is set, the Next Descriptor is fetched, and the process repeats. The actual packet transmission begins after either of the following:

- The MTL Tx Queue has reached a programmable Transmit threshold (Bits[6:4] of Transmit Operation Mode register of corresponding MTL Transmit Queue)
- A full packet is contained in the FIFO

You can also use the store-and-forward mode (Bit 1 of MTL Transmit Operation Mode Register of a queue). In this mode, descriptors are released (Own bit TDES0[31] clears) when the DMA finishes transferring the packet.

Note: To ensure proper transmission of a packet and the next packet, you must specify a non-zero buffer size for the Transmit descriptor that has the Last Descriptor (TDES3[28]) set.

Transmit Polling Suspended

It is important to understand how the Tx DMA suspends polling during the DMA Transmit operation.

Transmit polling can be suspended by any of the following conditions:

- The DMA detects a descriptor owned by the application (TDES3[31]=0).
  To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command by writing the Tail Pointer register. If the DMA goes into SUSPEND state because of this condition, Bit 15 and Bit 2 of Status Register of corresponding DMA channel are set.
- A packet transmission is aborted when a Transmit error is detected because of underflow.The appropriate Transmit Descriptor 3 (TDES3) bit is set. When this condition occurs, the following bits are set and the information is written to Transmit Descriptor 0, causing the suspension:
  - Bit 14 of Status Register of corresponding DMA channel
  - Transmit Underflow bit of corresponding queue in MTL_Interrupt_Status
- The DMA detects that the Tail Pointer is equal to the Current descriptor closed by the it.To resume, the software driver must modify the Tail Pointer register.

In all conditions, the position in the Transmit List is retained. The retained position is that of the descriptor following the Last Descriptor closed by the DMA. The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension cause.

DMA Transmit Channel Arbitration

It is important to understand how the Tx DMA handles channel arbitration during the DMA Transmit operation.

An arbiter provides access to multiple DMAs trying to access the Bus Interface Unit (BIU). **Figure 695** shows the arbitration process.

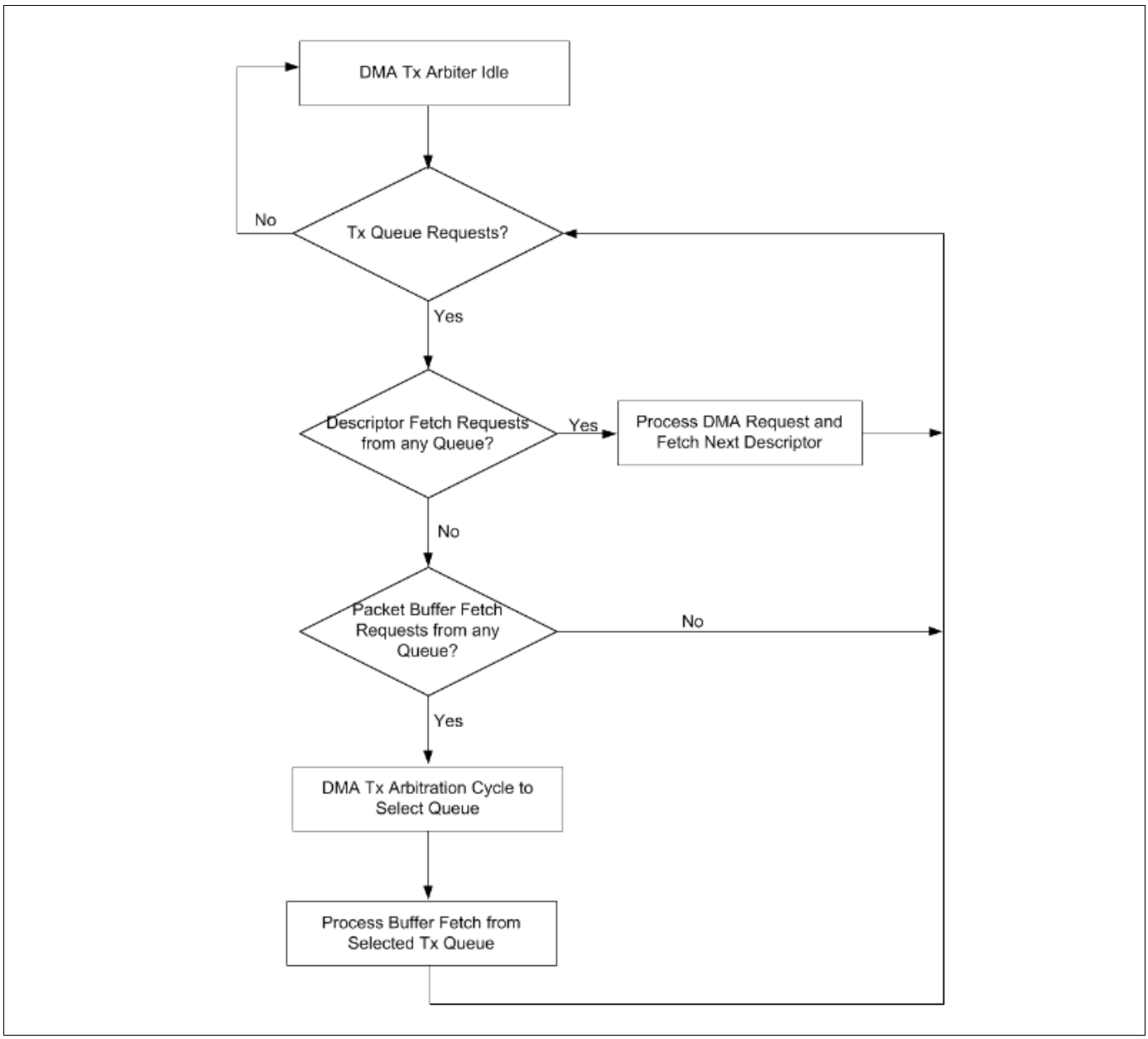


Figure 695 DMA Tx Channel Arbitration Process

When there is any request in the Tx queue, the DMA arbiter checks the type of the request: packet buffer fetch or descriptor fetch request. The descriptor fetch requests have higher priority than the buffer requests.Therefore, when there is a descriptor fetch request, the DMA arbiter acknowledges the DMA channel that is requesting for a descriptor fetch. If there is no descriptor fetch request, the arbiter looks for packet buffer fetch requests.

The DMA arbiter acknowledges the descriptor fetch request of one DMA channel at a time. Descriptor fetch requests are granted using a fixed priority with the higher channel having higher priority (Channel 1 having priority over Channel 0, Channel 2 having priority over Channel 1 and so on). For packet buffer fetches, the DMA arbiter uses the programmed channel weight and priority to decide which channel to acknowledge. The DMA arbiter performs a burst-by-burst arbitration based on one of the following algorithms:

- Weighted Strict Priority (WSP): In WSP arbitration mode, the arbiter first processes Channel 7 (or the last enabled channel) and then Channel 6, Channel 5, and so on. If any channel does not have a frame to transmit, the weight of that channel gets reassigned to Channel7 (or last enabled channel). If Channel 7 has no frames to transmit, the remaining weight is assigned to Channel 6 and so on.
- Weighted Round Robin (WRR): In WRR arbitration mode, the arbiter first selects the channel with the highest weight programmed, and then the channel with next highest weight, and so on. If any channel does not have a frame to transmit, the weight of that channel gets equally distributed to all channels that have frames to transmit.
- Fixed priority (FP): In Fixed priority mode, Channel 0 has the lowest priority and the last selected channel has the highest priority. The weight programmed in the Transmit Control register of a channel is ignored.

In WSP or WRR arbitration, the channel weight corresponds to the number of DMA burst transfers for which the DMA arbiter grants the bus to a channel. When a channel completes all the DMA burst transfers, the arbiter grants the bus to the next channel.

44.3.1.3.3 DMA Receive Operation

The Receive (Rx) DMA engine transfers data from the device port to the system memory.

In the Receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel. The ARI data at the start of the frame indicates the channel number to which the current frame must be written. If only one DMA channel is selected, this information is not provided.

Figure 696 shows the reception sequence for Rx DMA engine. The following list describes this sequence:

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[31]).
   The application must set the correct value in the Receive Descriptor Tail Pointer register of corresponding DMA channel.
2. When Bit 0 of Receive Control register of corresponding DMA channel is set, the DMA enters the Run state. The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor Tail Pointer register values. If there are no free descriptors, the DMA Channel enters the suspend state and goes to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the CTXT field (RDES3[30]).
5. The DMA processes the incoming packets and places these in the data buffers of acquired descriptor.
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
7. The DMA takes the status of the Receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last Descriptor bit set.
8. The DMA writes the Frame Length to RDES3 and VLAN Tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. If IEEE 1588 Timestamp feature is enabled, the DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).
10. If more descriptors are available in the Rx DMA Descriptor Ring, go to step 3; otherwise, go to the Suspend state (step 11).
11. The Receive DMA exits the Suspend state when a Receive Poll demand is given and the application advances the Receive Tail Pointer register of a channel.
    The engine proceeds to step 2 and refetches the next descriptor.

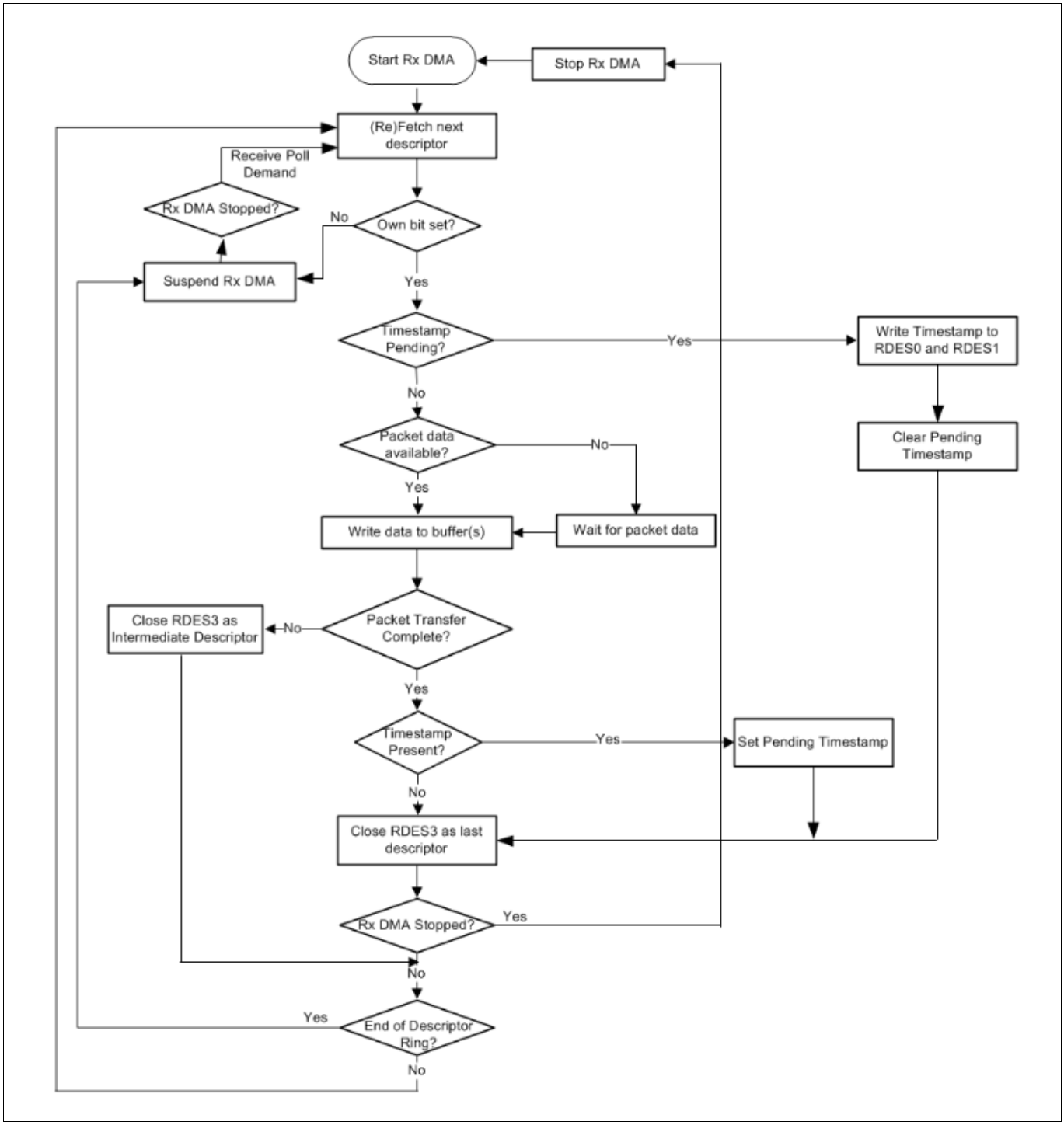


Figure 696 Receive DMA Operation

Receive Descriptor Acquisition

Under certain conditions, DWC_ether_qos attempts to acquire an extra descriptor in the anticipation of an incoming packet.

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming packet. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- Bit 0 of Receive Control Register of corresponding DMA channel is set immediately after being placed in the Run state.
- The Descriptor Tail pointer register value is ahead of the Current Descriptor acquired by the Rx DMA.
- The controller has completed packet reception, but the current Receive Descriptor is not yet closed.
- A Receive poll demand is issued (update of the Tail Pointer register).

Receive Packet Processing

During operation, the Rx engine tries to acquire an extra descriptor before it receives a packet and processes.

The sequence for processing a Receive packet is as follows:

1. The MAC transfers the received packets to the MTL memory only if the packet passes the address filter. If the packet fails the address filtering, it is dropped in the MAC block (unless Bit 31 of MAC_Packet_Filter register is set).
2. If packet size is greater than or equal to configurable threshold bytes set for Rx Queue of MTL, or when the complete packet is written to the queue in the store-and-forward mode, the MTL block requests the DMA block to begin transferring the packet data to the Receive Buffer pointed by the current descriptor.
   Packets smaller than 64 bytes, because of collision or premature termination, are removed from the MTL Receive Queue.
3. When the DMA application Interface (AHB/AXI or MDC) becomes ready, it transfers the data and sets the following:
   a) If the packet fits in a single descriptor, the DMA sets both Last Descriptor (RDES3[28]) and First Descriptor (RDES3[29]).
   b) If the packets fits into more than one descriptor, the DMA sets the First Descriptor (RDES3[29]) to delimit the packet.
4. The DMA releases the descriptors by resetting the Own (RDES3[31]) bit to 1’b0, either because the Receive buffer filled up or the last segment of the packet is transferred to the Receive buffer. The received packets status is updated in the last descriptor.
5. If Interrupt Enabled on Completion (RDES3[30]) bit is set in any of the Descriptors between the First and Last Descriptor of the Packet and Bit 6 of Interrupt Enable Register of corresponding DMA channel is set, the DMA sets Bit 6 of Status register of corresponding DMA channel.
   The same process repeats unless the DMA encounters a descriptor flagged as being owned by the application or when there are no more descriptors in the ring. When the DMA finds a descriptor owned by the application and if Bit 7 of Interrupt Enable Register of corresponding DMA channel is set, the Receive Process sets Bit 7 of Status register of corresponding DMA channel and then enters the Suspend state. The position in the receive list is retained.

44.3.1.3.4 Error Response to DMA

For any data transfer initiated by a DMA channel, if the slave replies with an error response, the DMA stops all operations and updates the error bits and the Fatal Bus Error bit in the Status Register of corresponding DMA channel. The application can either perform a reset to DWC_ether_qos or re-initialize the DMA descriptor list and start again. The rest of the DMA channels are not affected by such errors.

44.3.1.3.5 DMA Interrupts

There are two groups of interrupts that occur in the DMA Rx and Tx channel pair: Normal and Abnormal interrupts. Normal interrupts happen during packet transfers and abnormal interrupts indicate error events.

Interrupts can be generated as a result of various events in the DWC_ether_qos controller. These events are captured in status registers, and interrupt enables are provided for each source of an interrupt such that the interrupt signal (sbd_intr_o) is asserted for an event only when the corresponding interrupt enable is set.

In EQOS-AXI configuration, the interrupt status and corresponding enable registers are organized in an hierarchical manner so that it is easier for software to traverse and identify the source of an interrupt event quickly. When sbd_intr_o is asserted, the DMA_Interrupt_Status register is the first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX and RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- DMA_CH(#i)_Status (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register
- MTL_Interrupt_Status register
- MAC_Interrupt_Status register

The sbd_intr_o interrupt is a level signal and will get de-asserted only when all the enabled interrupt events are cleared in their respective status registers and correspondingly all the bits in the DMA_Interrupt_Status register are cleared.

The DMA_CH[n]_Status register captures all the interrupt events of that Tx DMA and Rx DMA channel pair. The DMA_CH[n]_Interrupt_Enable register contains the corresponding enable bits for each of the interrupt event. There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts. They are indicated by Bits[15:14] of DMA_CH[n]_Status register respectively. The normal group is for events that happen during the normal transfer of packets (TI, RI, TBU) while the abnormal interrupt events are for error events. Interrupt events are cleared by writing 1’b1 to the corresponding bit position. When all the enabled interrupt events are cleared (including the NIS and AIS), the interrupt source from the DMA Channel is cleared and the corresponding bit in DMA_Interrupt_Status register is also cleared.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. For example, Receive Interrupt Bit[6] of DMA_CH[n]_Status register indicates that one or more packets were transferred to the application buffer. The driver must scan all descriptors, from the last recorded position to the first one, owned by the DMA to determine how many packets are received.

An interrupt is generated only once for multiple events. The driver must scan the DMA_Interrupt_Status register for the cause of the interrupt and clear the source in the respective Status register. The sbd_intr_o is cleared only when all the bits of DMA_Interrupt_Status register are cleared.

Periodic Scheduling of Transmit and Receive Interrupt

To improve the throughput and performance, DWC_ether_qos supports interrupt timer and transmit descriptor to generate interrupts periodically, instead of every DMA transfer.

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The DWC_ether_qos gives the flexibility to schedule the interrupt at regular intervals using two methods:

- Set Interrupt on Completion bit in Transmit descriptor (TDES2[31]) once for every “required” number of packets to be transmitted.
- Similarly, set the IOC (RDES3[30]) bit only at some specific intervals of Receive descriptors. This way, whenever a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated.

In addition to above, an interrupt timer (DMA_CH[n]_Rx_Interrupt_Watchdog_Timer) is given for flexible control and periodic scheduling of Receive Interrupt. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive Interrupt because the corresponding interrupt of completion IOC bit (RDES3[30]) is not set. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in DMA_CH[n]_Interrupt_Enable register.The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Per Channel Transfer Complete Interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in DMA_CH[n]_Status register.

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in DMA_CH[n]_Status register. The TI bit is set whenever the Tx DMA channel closes the a descriptor in which the IOC (Interrupt On Completion - TDES2[31]) bit is set. Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with LD bit set and in any of the descriptors used for transferring that packet, IOC (Interrupt Enable on completion - RDES3[30]) bit is set.

The common sbd_intr_o output signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in DMA_CH[n]_Interrupt_Enable register.

The behavior of the RI/TI/ sbd_perch_tx_intr_o[]/sbd_perch_rx_intr_o[] changes depending on the settings of INTM field bit[17:16] in DMA_Mode. **Table 438** explains the Transfer Complete Interrupt behavior.

Table 438 DWC_ether_qos Transfer Complete Interrupt Behavior

Sl No	Interrupt Mode	Behavior of the sbd_perch_tx_intr_o[] and sbd_perch_rx_intr_o[]	Behavior of TI/RI and sbd_intr_o
1.	INTRM=0	A pulse is asserted on these output signals when corresponding TX/RX transfercomplete event (for which IOC descriptor bits are enabled) is detected.	The TI/RI status signals are set whenever the "Transfer complete" event is detected. The bits get cleared whenever the software driver writes'1' to these bits. The sbd_intr_o is asserted when ever the corresponding interrupts are also enabled in DMA_CH[n]_Interrupt_Enable register.

Table 438 DWC_ether_qos Transfer Complete Interrupt Behavior (cont'd)

Sl No	Interrupt Mode	Behavior of the sbd_perch_tx_intr_o[] and sbd_perch_rx_intr_o[]	Behavior of TI/RI and sbd_intr_o
2.	INTM=1	These signals reflect the value ofcorresponding TI/RI bits inDMA_CH[n]_Status register when thecorresponding interrupt enable is set. Hence they are level signals and are cleared by the application by writing 1'b1 to the RI/TI status bits. This signal will not be asserted when the corresponding interrupt enable bit is not set.	The TI/RI is set as explained above.However, the sbd_intr_o signal is not asserted for any RI/TI events.
3.	INTM=2	In this mode, RI/TI interrupts are queued. These signals reflect the value of corresponding TI/RI bits inDMA_CH[n]_Status register when thecorresponding interrupt enable is set. They are level signals and cleared by software by writing 1'b1 to the RI/TI status bits. However, it is set again if another TI/RI event(s) is detected before the TI/RI bits are cleared forthe previous event.	The RI/TI status bits are set whenever the Transfer Complete event is detected and gets reset whenever software driver clears those bits by writing 1. However, if another Transfer Complete event is detected before it is cleared (serviced) by the software, then DWC_ether_qos automatically set these status bits again.However, the sbd_intr_o signal is not generated based on TI/RI

44.3.1.4 MAC Transaction Layer

44.3.1.4.1 SPRAM Interface

The MAC supports the Single Port RAM Interface. In the DPRAM, each transmit and Receive FIFO memory is split into two SPRAM blocks.

Figure 697 shows the high level architectural approach for supporting single port RAM interface. It describes the data flow, clock domains and memory interfaces in EQOS-MTL configuration with GMII/MII PHY interface.

As compared to a single DPRAM instance (with application clock on 1 port and GMII/MII clock on the other port), each Transmit and Receive FIFO memory is split into two SPRAM blocks (Odd and Even) and operates with the application clock. Each SPRAM block will have the same width as earlier (DPRAM) but will have half the depth of the total FIFO memory selected in the configuration. Thus the total memory space will be retained as configured.

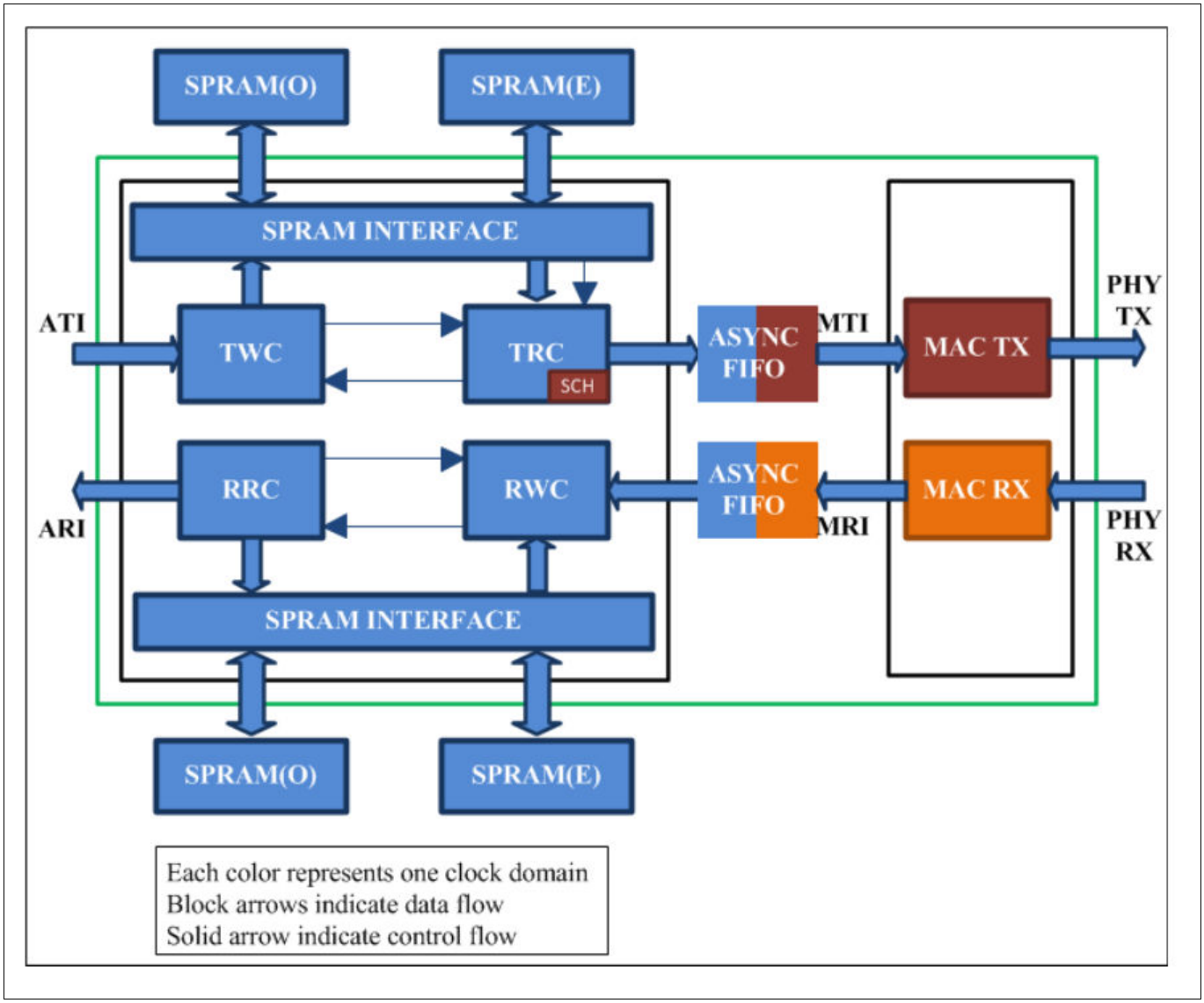


Figure 697 SPRAM Architecture Block Diagram

Application Clock Frequency Requirements for Memory Transfers

The application clock frequency must be such that the data transfer bandwidth of memory must be at least twice the bandwidth as that of the line.

Due to the requirement of simultaneous read and write transfers from the FIFO memories, the application clock frequency should be such that the data transfer bandwidth of memory must be at least twice the bandwidth as that of the line.

In addition to the previous requirement, additional bandwidth is required to allow delays due to contention between read and writes to a single memory block (Odd or Even). The probability of contention increases with multiple Queues (in Tx Side) especially in EQOS-AXI and EQOS-MTL configurations. These are mainly due to

- Possibility of receiving and writing interleaved data of different queues, all of which are to the same (odd or even) memory clocks in consecutive clock cycles.
- Non-sequential reads & writes of control words for each packet by scheduler in TX side
- Non-sequential writes of status words for each packet in RX side.

Considering the above, the minimum application clock frequency estimate in the various modes/speeds/configurations to avoid underflow in the "Async FIFO" in TX path is given in **Table 439**.

Table 439 Application Clock Frequency Estimate in Various Modes/Speeds

Configuration	No of Tx Queues	DW = 32
EQOS-AHB, EQOS-DMA	Any	62.5 MHz

When the speed is 100Mbps (MII mode), the minimum application clock frequency is reduced by factor of 10.

In the RX side, contention can happen if the slave memory does not accept data continuously and in the worst case there can be a contention for every memory access if the slave memory introduces one busy cycle after every transfer. However, status word writes steal cycles during the IPG (Inter-packet Gap) of received packets. The number of status words to be written depends on the configuration (for example, IEEE 1588 requires 64-bit timestamps, VLAN stripping requires 32-bit VLAN tags, etc). The depth (or area) of the register Asynchronous FIFO has to be larger to avoid overflows.

However, the minimum application clock frequency will be the same in all configurations in 1 Gbps speed as shown in **Table 440**.

Table 440 Minimum Application Clock Frequency in Various Configurations

Configuration	No of Rx Queues	DW = 32
Any	Any	69 MHz

For 100Mbps operating mode, the application clock frequency can be reduced by a factor of 10.

Notes

1. *The previous tables are pessimistic and estimated with a worst case scenario assumptions. These will be validated in simulation and updated later.*
2. *To handle valid frames of size < 64 bytes received in the line, increase the application clock frequency further, as the overhead due to status is considerably large on Rx side.*
3. *The previous computation assumes 12 bytes of IPG and 8 bytes of preamble+SFD. To handle reduced IPG and preamble, increase the application clock frequency further, to transfer the status in the available gaps between packets on the application interface (MRI).*

44.3.1.5 MAC

The MAC communicates with the MTI, MRI, and MCI interfaces of the DWC_ether_controller.

The MAC supports many interfaces towards the PHY chip. The PHY interface can be selected only once after reset. The MAC communicates with the application side with the MAC Transmit Interface (MTI), MAC Receive Interface (MRI), and MAC Control Interface (MCI).

44.3.1.5.1 MAC Transmission

The following six modules constitute the transmission function of the MAC:

- Transmit Bus Interface Module
- Transmit Packet Controller Module
- Transmit Protocol Engine Module
- Transmit Scheduler Module
- Transmit CRC Generator Module
- Transmit Flow Control Module

MAC Transmit Bus Interface Module

The MAC Transmit Bus Interface (TBU) accepts data in 32, 64, or 128-bit wide bus and runs on the clk_tx_i clock of GMII interface.

The Transmit Bus Interface (TBU) module connects the Transmit path of the MAC with an external packet through a FIFO interface.

MAC Transmit Packet Controller Module

The MAC Transmit Packet Controller (TPC) module consists of eight registers to hold the data and the last data control received from the TBU.

The Transmit Packet Controller (TPC) module consists of eight registers to hold the data and the last data control received from the TBU.

MAC Transmit Protocol Engine Module

The MAC Transmit Protocol Engine (TPE) module consists of a Transmit State Machine that controls the operation of Ethernet packet transmission.

The Transmit Protocol Engine (TPE) module consists of a Transmit State Machine that controls the operation of Ethernet packet transmission.

The Transmit State Machine of this module contains the following features to meet the IEEE 802.3/802.3z specification:

- Generates preamble and SFD
- Generates jam pattern in the half-duplex mode after normal collision
- Generates carrier extension in the half-duplex (only in the GMII) mode when packet is smaller than 512 bytes
- Supports packet bursting in the half-duplex (only in the GMII) mode
- Supports jabber timeout
- Supports flow control for the half-duplex mode (backpressure)
- Generates Transmit packet status
- Contains timestamp snapshot logic for IEEE 1588 support

When the TPC module requests the TPE module for a new packet transmission, the Transmit State Machine sends out the preamble and SFD, followed by the data received. The preamble is defined as 7 bytes of 8’b10101010 pattern and the SFD is defined as 1 byte of 8b’10101011 pattern.

The collision window is defined as 1 slot time (512-bit times for 10/100 Mbps Ethernet and 4,096 bit times for 1,000 Mbps Ethernet). The jam pattern generation is applicable only to half-duplex mode, not to full duplex mode. In full-duplex mode, the Transmit State Machine ignores the phy_col_i signal from the PHY.

In MII mode, if a collision occurs any time from the beginning of the packet to the end of the CRC field, the Transmit State Machine sends a 32-bit jam pattern of 32’h55555555 on MII to inform all other stations that a collision has occurred. If the collision is seen during the preamble transmission phase, the Transmit state machine completes the transmission of preamble and SFD, and then sends the jam pattern.

In GMII mode, if a collision occurs any time between the beginning of the packet and the end of the extension field, the Transmit State Machine sends a 32-bit jam pattern of 32’h55555555 on GMII to inform all other stations of the collision. If the collision is seen during the preamble transmission phase, the Transmit State Machine completes the transmission of preamble and SFD, and then sends the jam pattern. If a collision occurs during the extension field, the Transmit State Machine sends a 32-bit jam pattern of 32'h1F1F1F1F.

If the collision occurs after the collision window and before the end of the FCS field (or the end of Burst if the Packet Burst mode is enabled), the Transmit State Machine sends a 32-bit jam pattern and sets the late collision bit in the Transmit packet status.

Note: At the GMII or MII interface, the collision signal (phy_col_i) being asynchronous is checked by the transmitter after it is double-synchronized to clk_tx domain. This additional latency delays the recognition of collision or late-collision event. When the output of phy_col_i synchronizer is asserted after the complete packet is transmitted, it is not recognized as collision even if the COL signal is high at the GMII interface before the end of transmission. Similarly, an assertion of COL signal at the GMII input at the last byte of normal collision window might be identified as late collision because of the synchronizer delay.

In GMII half-duplex mode (1000 Mbps), the Transmit State Machine ensures that all valid carrier events exceed a slot time of 4,096 bit times. To accomplish this, any Transmit packet shorter than 512 bytes from the TFC module is extended using a carrier extension. On GMII, this is signaled to the PHY by asserting phy_txer_o, de-asserting phy_txen_o, and setting phy_txd_o[7:0] to 8h'0F.

When the Packet Burst mode is enabled, only the first packet of the burst is carrier extended if it is shorter than 512 bytes. The carrier extension is not applicable for MII half-duplex and GMII or MII full-duplex modes. When the Packet Burst mode is enabled, the Transmit State Machine transmits a burst of packets (as long as packets are available from the TFC module) without releasing the carrier of the PHY. To accomplish this, the state machine inserts the carrier extension for a minimum IPG period (96 bit times) between the packets. The Transmit State Machine continues to burst packets as long as additional packets are available from the TPC module and a burst limit of 8,192 byte times has not been exceeded. If an additional packet is not available at the end of the IPG period in the middle of the burst, the Transmit State Machine releases the carrier by de-asserting the phy_txer_o and phy_txen_o signals on GMII.

The TPE module maintains a jabber timer to stop the transmission of Ethernet packets if the TFC module transfers more than 2,048 (default) bytes. The timeout is changed to 10,240 bytes when the Jumbo packet is enabled.

The Transmit State Machine uses the deferral mechanism for flow control (backpressure) in the half-duplex mode. When the application requests to stop receiving packets, the Transmit State Machine sends a JAM pattern of (8'h55) 32 bytes whenever it senses a reception of a packet, provided the Transmit flow control is enabled. This results in a collision and the remote station backs off. The application requests the flow control through a sideband signal mti_flowctrl_i (or by setting BPA bit of the Flow Control Register of corresponding MTL queue). If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort their transmissions because of excessive collisions.

If IEEE 1588 timestamp is enabled for the Transmit packet, this block takes a snapshot of the system time when the SFD is put onto the Transmit GMII or MII bus. The system time source is either an external input or it is internally generated according to the configuration selected.

Transmit Module

The MAC Transmit Scheduler (STX) module is responsible for scheduling the packet transmission on GMII or MII.

The Transmit Scheduler (STX) module is responsible for scheduling the packet transmission on GMII or MII. It provides an enable signal to the TPE module after satisfying the IPG and back-off delays.

The STX module performs the following functions:

- Maintains the inter-packet gap between two transmitted packetsThe STX module maintains an idle period of the configured inter-packet gap (IPG bits of MAC_Configuration register) between any two transmitted packets. If packets from the TPC arrive at the TPE module sooner than the configured IPG time, the TPE module waits for the enable signal from the STX module before starting the transmission on GMII or MII. The STX module starts its IPG counter as soon as the carrier signal of GMII or MII goes inactive. At the end of programmed IPG value, the module issues an enable signal to the TPE module in the full-duplex mode.

- Implements the Truncated Binary Exponential Back-off algorithm
  The STX module implements the Truncated Binary Exponential Back-off algorithm when it operates in the half-duplex mode.

MAC Transmit CRC Generator Module

The MAC Transmit CRC Generator (CTX) module interfaces with the TFC module to generate CRC for the FCS field of the Ethernet packet.

The Transmit CRC Generator (CTX) module interfaces with the TFC module to generate CRC for the FCS field of the Ethernet packet.

The TPC module sends the packet data and any necessary padding to the CTX module through an 8-bit interface.

The CTX module calculates the 32-bit CRC for the FCS field of the Ethernet packet. The encoding is defined by the following generating polynomial:

G (x) = x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

The CTX module gets the byte data of Ethernet packet from the TPC module (DA + SA + LT + DATA + PAD) qualified with a Data Valid signal. The TPC also indicates to the CTX when to reset the previously calculated CRC and to start the new CRC calculation for the coming packet. The TPC module issues the start command before sending the new packet data for calculation. The calculated CRC is valid on the next clock after the data is received.

In GMII mode, the Data Valid signal is valid for every clock, from the first data byte through the last data byte. In MII mode, this signal is valid every alternate clock.

MAC Transmit Flow Control Module

The MAC Transmit Flow Control (FTX) module generates and transmits the Pause packets to the TFC module based on the flow control triggers in full-duplex mode.

The Transmit Flow Control (FTX) module generates and transmits the Pause packets to the TFC module based on the flow control triggers in full-duplex mode.

The TFC module receives the Pause packet from the FTX module, appends the calculated CRC, and sends the packet to the TPE module.

44.3.1.5.2 MAC Reception

A receive operation is initiated when the MAC detects an SFD on GMII or MII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

The following are the functional blocks in the Receive path of the MAC.

- Receive Protocol Engine Module
- Receive CRC Module
- Receive Packet Controller Module
- Receive Flow Control Module
- Receive Bus Interface Unit Module
- Address Filtering Module

MAC Receive Protocol Engine Module

The MAC Receive Protocol Engine (RPE) consists of the Receive State Machine which strips the preamble, SFD, and carrier extension of the received Ethernet packet (in half-duplex 1000-Mbps mode).

The Receive Protocol Engine (RPE) consists of the Receive State Machine which strips the preamble, SFD, and carrier extension of the received Ethernet packet (in half-duplex 1000-Mbps mode).

MAC Receive CRC Module

The MAC Receive CRC (CRX) interfaces with the RPE module to check any CRC error in the packet being received.

The Receive CRC (CRX) interfaces with the RPE module to check any CRC error in the packet being received. This module calculates the 32-bit CRC for received packet that includes the Destination address field through the FCS field.

MAC Receive Packet Controller Module

The MAC Receive Packet Controller (RPC) receives the Ethernet packet data and status from the RPE module.

The Receive Packet Controller (RPC) receives the Ethernet packet data and status from the RPE module.

MAC Receive Bus Interface Unit Module

The MAC Receive Bus Interface Unit (RBU) converts the 32-bit data received from the RPC module into a 32-bit, 64-bit, or 128-bit FIFO protocol on the Application side.

The Receive Bus Interface Unit (RBU) converts the 32-bit data received from the RPC module into a 32-bit, 64-bit, or 128-bit FIFO protocol on the Application side.

Address Filtering Module

The MAC Address Filtering (AFM) module performs the destination and source address checking function on all received packets and reports the address filtering status to the RPC module.

The Address Filtering (AFM) module performs the destination and source address checking function on all received packets and reports the address filtering status to the RPC module.

44.3.2 Processing Double VLAN

44.3.2.1 Introduction to Double VLAN Processing

The DWC_ether_qos supports the double VLAN tagging feature, in which the MAC can process up to two VLAN tags.

DWC_ether_qos supports two VLAN tags, namely inner and outer, for processing double VLANs. This feature is referred as the double VLAN tagging feature in which the MAC can process two VLAN tags. With this feature, the DWC_ether_qos supports the following:

- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path.
- Packet filtering and stripping based on any one of the two VLAN Tags in the Receive path. Stripping and providing up to two VLAN Tags in the Receive path as a part of the Receive status.

44.3.2.2 Description to Double VLAN Processing

DWC_ether_qos supports the double VLAN tagging feature to enable the MAC layer to support up to two VLAN tags. The MAC layer supports various features on the transmit and receive side to support the double VLAN tagging feature.

Table 441 describes the features supported by the MAC on the Transmit side.

Table 442 describes the features supported by the MAC on the Receive side and the corresponding bits in the MAC_VLAN_Tag register.

Table 441 Double VLAN Processing Features in Transmit Path

Feature	Description
Support for C-VLAN and S-VLAN Tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of MAC_VLAN_Incl and MAC_Inner_VLAN_Incl registers. The DWC_ether_qos supports processing of any sequence of outer and inner VLAN tags. Note: The DWC_ether_qos does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: • The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. • The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. • The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. • The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.
VLAN Tag deletion	You can enable the VLAN tag deletion for outer or inner tag through VLC field in the MAC_VLAN_Incl or MAC_Inner_VLAN_Incl register, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN Tag Insertion or Replacement	You can enable the VLAN tag insertion or replacement for outer or inner tag through VLC field in the MAC_VLAN_Incl or MAC_Inner_VLAN_Incl register, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the MAC_VLAN_Incl or MAC_Inner_VLAN_Incl register is used to determine whether the VLAN tag should be taken from the register or the Control Word.

Table 442 Double VLAN Processing in Receive Path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.
Outer and Inner VLAN Tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.
16-bit outer and inner VLAN Tag and Type in Rx status	The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.
Disabling or skipping checking of outer VLAN Tag type	The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.

44.3.3 Source Address and VLAN Insertion, Replacement, or Deletion

44.3.3.1 Introduction to SA and VLAN Insertion, Replacement, or Deletion

DWC_ether_qos supports the feature to insert, replace, or delete the source address (SA) and VLAN fields that are part of transmit packets.

The source address (SA) and VLAN fields are Tx packet-related control information that are provided as part of the control word through ATI interfaces. DWC_ether_qos supports the feature to insert or replace the source address based on the information in the MAC Address Registers, and also the feature to insert, replace or delete the VLAN fields (VLAN Type and VLAN Tag) based on the setting of the VLTI bit in the MAC_VLAN_Incl register. You can enable the SA insertion or replacement feature for all Transmit packets or selective packets. Similarly, you can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets.

44.3.3.2 Programming Source Address Insertion or Replacement

DWC_ether_qos supports the feature to insert or replace the source address (SA) with information in the SA field of the MAC Address register.

The software can use the SA insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address Registers in the SA field
- Replace the content of the SA field with the content of the MAC Address Registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

You can enable the SA insertion or replacement feature for all Transmit packets or selective packets:

- Enabling SA insertion or replacement for all packets
  To enable this feature for all packets, program the SARC field of the MAC_Configuration register.
- Enabling SA insertion or replacement for selective packets
  To enable this feature for selective packets, use the following method depending on the configuration:
  - EQOS-AHB, , or :
    Program the SA Insertion Control field (Bits[25:23] of TDES3) in the first Transmit descriptor of the packet. When Bit 25 of TDES3 is set, the SA Insertion Control field indicates insertion or replacement by MAC Address1 registers. When Bit 25 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers.

If MAC Address1 Registers are not enabled, the MAC Address0 registers are used for insertion or replacement irrespective of the value of the most-significant bit of the SA Insertion Control field.

44.3.3.3 Programming VLAN Insertion, Replacement, or Deletion

DWC_ether_qos supports the feature to insert, replace or delete the VLAN fields (VLAN type and VLAN tag) based on the setting of the VLTI bit in the MAC_VLAN_Incl register.

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields

- Insert or replace the VLAN Type and VLAN Tag fields
  Insertion or replacement is done based on the setting of VLTI bit in the MAC_VLAN_Incl register as described **Table 443**

Table 443 VLAN Insertion or Replacement Based on VLTI Bit

Condition	Description
VLTI bit is set	The MAC inserts or replaces the following: • VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of MAC_VLAN_Incl register) • VLAN Tag field with one of the following depending upon the configuration: – EQOS-AHB, , or : Content of the VT field of Transmit context descriptor of the packet
VLTI bit is reset	The MAC inserts or replaces the following: • VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of MAC_VLAN_Incl register) • VLAN Tag field with the VLT field of MAC_VLAN_Incl register

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88a8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- Enabling VLAN insertion, replacement, or deletion for all packets
  To enable this feature for all packets, program the VLC and VLP fields of MAC_VLAN_Incl register.
- Enabling VLAN insertion, replacement, or deletion for selective packets
  To enable this feature for selective packets, use the following method depending upon the configuration:In addition, the VLP (VLAN Priority control) bit must be reset in Register 24 (for outer VLAN) and Register 25 (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.
  – EQOS-AHB, , or : Program the VTIR field of TDES2 Normal Descriptor).

In addition, the VLP (VLAN Priority control) bit must be reset in Register 24 (for outer VLAN) and Register 25 (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.

44.3.4 Queue Channel Based VLAN Tag Insertion

44.3.4.1 Introduction to Queue/Channel Based VLAN Tag Insertion on Tx

DW_ether_qos supports insertion of VLAN tags in Transmit packets and these VLAN tags can be based on a queue or channel.

DWC_ether_qos supports channel/queue based VLAN tag insertion on all transmitted packets.

44.3.4.2 Accessing Queue/Channel Specific VLAN Tag Registers

You can access VLAN tags specific to a queue/channel using the MAC_VLAN_Incl register.

Queue/Channel specific VLAN tag registers are accessed using indirect addressing via the MAC_VLAN_Incl register. VLAN type and tag value can be independently programmed for each queue/channel.

44.3.5 Managing Buffers and Memories

44.3.5.1 Introduction to Transmit and Receive FIFOs

DWC_ether_qos provides transmit and receive FIFOs to buffer data while transmitting and receiving information from an application.

The Transmit FIFO (Tx FIFO) buffers the data transferred from the application . Similarly, the Receive FIFO (Rx FIFO) stores the Ethernet frames received from the line until they can be transferred to the application. These are asynchronous FIFOs because they also transfer the data between the application clock and the MAC line clocks. Tx memory and Rx memory are required to be two-ported RAM of 35-bit or 68-bit or 133-bit width for 32, 64 or 128 data bus widths, respectively. The extra bits are used for storing the byte-enable information.

When multiple queues are selected, all Tx queue share the Tx FIFO memory and all Rx queue share the Rx FIFO memory. The application can program the size of the FIFO memory allocated to each Tx or Rx queue.

44.3.5.2 Transmit and Receive FIFO-Related Registers

The Transmit (Tx) and Receive (Rx) registers are used as buffers while sending and receiving information from an application.

Following are the Transmit and Receive FIFO-related registers:

- MTL_TxQ0_Operation_Mode
- MTL_RxQ0_Operation_Mode

44.3.6 Using PHY Interface

The DWC_ether_qos supports a number of interfaces that can be enabled as the Line interface to the PHY.

The following topics are discussed:

- Station Management Agent
- Reduced Gigabit Media Indepent Interface
- Reduced Media Independent Interface

44.3.6.1 Station Management Agent

44.3.6.1.1 Introduction to Station Management Agent

The application can access the PHY registers through the Station Management Agent (SMA) module. SMAis a two-wire Station Management interface (MIM).

44.3.6.1.2 Functional Description of Station Management Agent

For MIM accesses, the maximum operating frequency of the MDC (gmii_mdc_o) is 2.5 MHz, as specified inthe IEEE 802.3. In the DWC_ether_qos core, the gmii_mdc_o clock is derived from the application clock orclk_csr_i , using a divider-counter. The divide factor depends on the clock range setting (CR field) in theMAC_MDIO_Address register.Select the clock divide factor as mentioned in the description of CR field of MAC_MDIO_Address register,to meet IEEE specifications. However, if your system supports higher clock frequencies on the MIMinterface, there is a provision to select a different divider.

The MDIO frame structure is as follows:

Table 444 MDIO Clause 45 Frame Structure

Field	Description
IDLE	The mdio line is three-state; there is no clock on gmii_mdc_o.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 2’00
OPCODE	2’b09, 2’b01, 2’b10, 2’b11
PHY ADDR	5-bit address select for one of 32 PHYs
DEV ADDR	5-bit address select for one of 32 devices
TA	Turnaround is 2’bZ0 for Read and post-read increment address and 2’b10 for Write and address MDIO accesses Where Z is the tri-state level
DATA/ADDRESS	16-bit value. For an address cycle (OPCODE = 2'b00), this frame contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, this field contains the data to be written to the register. For read or post-read increment address frames, this field contains the contents of the register read from the PHY. In address and data write cycles, the DWC_ether_qos drives the MDIO line during the transfer of these 16 bits.In read and post-read increment address cycles, the PHY drives the MDIO line during the transfer of these 16 bits.

The frame structure for Clause 22 frames is also supported. The C45E bit in the MAC_MDIO_Addressregister can be programmed to enable Clause 22 or Clause 45 mode of operation. The following table showsthe Clause 22 frame format.

Table 445 MDIO Clause 22 Frame Structure

Field	Description
IDLE	The mdio line is three-state; there is no clock on gmii_mdc_o.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 2’01
OPCODE	2’b10 for Read and 2’b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	5-bit address to select the register within each MMD
TA	Turnaround is 2’bZ0 for Read and 2’b10 for Write Where Z is the tri-state level
DATA/ADDRESS	Any 16-bit value. In a Write operation, the MAC drives mdio. In a Read operation, the PHY drives it.

In addition to normal read and write operations, the SMA also supports post-read increment address whileoperating in Clause 45 mode.

GMII/MII Management Write Operation

After the Station Management Agent receives the PHY address and the write data from the MAC CSR module, the SMA starts a Write operation to the PHY registers.

Figure 698 illustrates the flow for a write operation from the SMA module to the PHY registers.

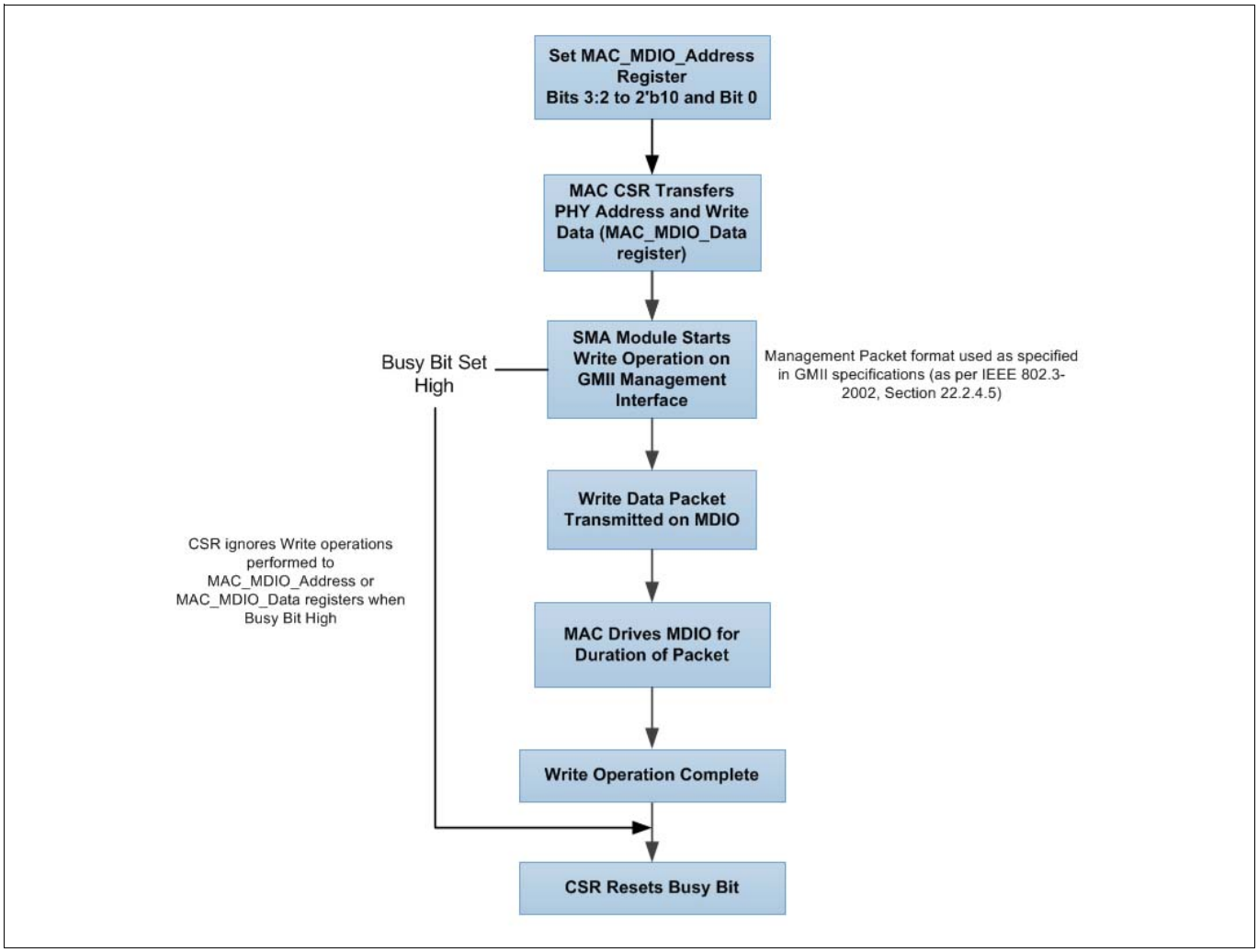


Figure 698 SMA Write Operation Flow

GMII/MII Management Read Operation

After the Station Management Agent receives the PHY address and the register address in the PHY from the MAC CSR module, the SMA initiates a Read operation to the PHY registers.

The flow of a Management read operation is as follows:

1. Set Bit[3:2] to 2'b11 and Bit 0 in the MAC_MDIO_Address register
2. The MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers
3. The SMA module starts a Read operation on the GMII Management Interface using the Management Packet Format specified in the GMII specifications (as per IEEE 802.3-2002, Section 22.2.4.5).
4. When the SMA module starts a Read operation on the MDIO, the CSR ignores the Write operations to the MAC_MDIO_Address or MAC_MDIO_Data register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface.
5. When the Read operation is complete, the SMA indicates this to the CSR.
6. The CSR resets the Busy bit and updates the MAC_MDIO_Data register with the data read from the PHY.

44.3.6.1.3 Preamble Suppression

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The DWC_ether_qos providescontrols to support preamble suppression, it will transmit MDIO frames with only 1 preamble bit. Thepreamble suppression can be enabled by setting PSE bit of MAC_MDIO_Address register.

44.3.6.1.4 Trailing Clocks and Back to Back transactions

The DWC_ether_qos drives MDC clock on the gmii_mdc_o port for duration of the MDIO frame. There isno clock driven during the idle period. The trailing clocks feature can be used if the PHY needs the MDCclock to be active for some cycles after the MDIO frame. The NTC field in MAC_MDIO_Address registerallows programming of trailing clocks from 0 to 7.

The DWC_ether_qos supports Back to Back transactions which allows start of next MDIO frame even beforethe trailing clocks are completed for previous MDIO frame. This feature can be enabled by setting BTB bit inMAC_MDIO_Address register when trailing clocks feature is also enabled. When Back to Back transactionsis enabled, the GMII Busy will be cleared immediately after MDIO frame completion allowing SW to issuenext command, which will be executed by DWC_ether_qos while trailing clocks are still on for previousMDIO frame. When Back to Back transactions is not enabled, the GMII Busy will be cleared after the trailingclocks are completed for MDIO frame.

44.3.6.1.5 Interrupt for MDIO transaction completion

The MDIO interface generates an interrupt on completion of MDIO read or write transactions. The application can avoid polling the GMII Busy bit of MAC_MDIO_Address register and know the completion of MDIO commands based on the interrupt generated. The interrupt is generated only if the MDIO Interrupt Enable bit of MAC Interrupt Enable bit is set.

44.3.6.2 Reduced Gigabit Media Independent Interface

44.3.6.2.1 Introduction to Reduced Gigabit Media Independent Interface (RGMII)

The Reduced Gigabit Media Independent Interface (RGMII) module is implemented in DWC_ether_qos to reduce the pin count as mentioned in the RGMII specification.

The Reduced Gigabit Media Independent Interface (RGMII) specification reduces the pin count of the interconnection between the MAC and the PHY for GMII and MII interfaces. To achieve this, the data path and control signals are reduced and multiplexed together with both edges of the Transmit and Receive clocks. For Gigabit operation, the clocks operate at 125 MHz; for 10/100 operation, the clock rates are 2.5 MHz/25 MHz.

The RGMII module is instantiated between the GMII of the MAC and the PHY to translate the control and data signals between the GMII and RGMII protocols.

The RGMII block has the following characteristics:

- Supports 10 Mbps, 100 Mbps, and 1000 Mbps operation rates
- Requires no extra clock because both edges of the incoming clocks are used. To simplify back-end implementation of the MAC, there are separate clock inputs (180 degrees out-of-phase with the associated transmit/receive clocks) for logic that uses the falling edges. For guidelines for clock connections when the MAC operates with an RGMII, see DesignWare Cores Ethernet Quality-of-Service User Guide.
- Extracts the in-band (link speed, link status, and duplex mode) status signals from the PHY and provides them to the MAC for link detection

44.3.6.3 Reduced Media Independent Interface

44.3.6.3.1 Introduction to Reduced Media Independent Interface

The Reduced Media Independent Interface is instantiated between the MAC and the PHY. The RMII helps to translate the MII signals (to/from the MAC) to RMII signals (to/from the PHY).

The Reduced Media Independent Interface (RMII) specification reduces the pin count between Ethernet PHYs and Switch ASICs (only in 10/100 mode). According to the IEEE 802.3u, an MII contains 16 pins for data and control. In devices incorporating multiple MAC or PHY interfaces (such as switches), the number of pins adds significant cost with increase in port count. The RMII specification addresses this problem by reducing the pin count to 7 for each port — a 62.5% decrease in pin count

The RMII module is instantiated between the MAC and the PHY. This helps in translating the MII of the MAC into the RMII. The RMII block has the following characteristics:

- Supports 10 Mbps and 100 Mbps operating rates. It does not support the 1000 Mbps operation
- Provides independent 2-bits wide Transmit and Receive paths by sourcing two clock references externally.

44.3.7 Packet Filtering

44.3.7.1 Packet Filtering Sequence

The receive packet passes through various filters such as source/destination address filter, VLAN filter, before being delivered to the host or being discard.

Figure 699 shows the filtering sequence for Rx packets.

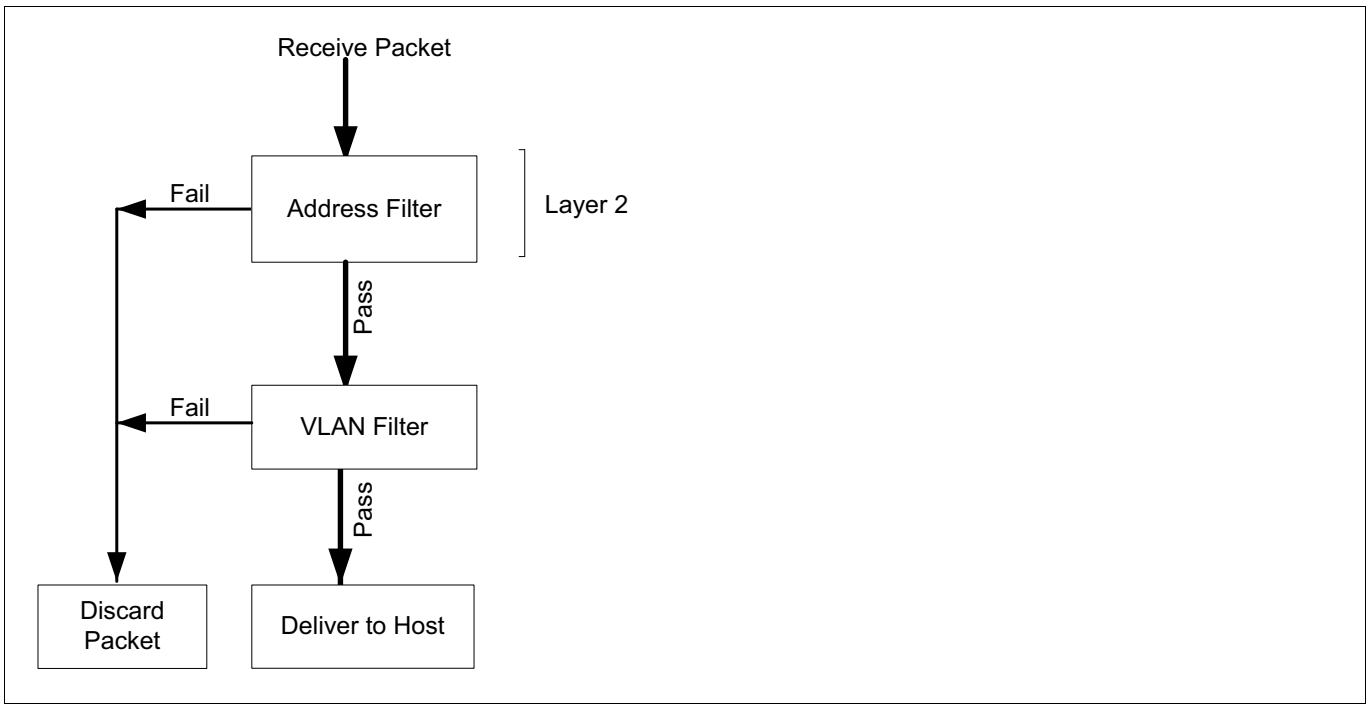


Figure 699 Packet Filtering Sequence

44.3.7.2 Source Address or Destination Address Filtering

44.3.7.2.1 Introduction to Source or Destination Address Filtering

DWC_ether_qos supports the address filtering module that checks the source and destination address of incoming packets.

The Address Filtering Module of the MAC checks the source address (SA) and destination address (DA) fields of each incoming packet.

44.3.7.2.2 Programming Different Types of Address Filtering

The different types of address filtering that you can program are unicast destination, multicast destination, hash and perfect address, broadcast address, unicast source address and inverse filtering.

Unicast Destination Address Filtering

The MAC supports up to 128 MAC addresses for unicast perfect filtering. MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr127 addresses are selected with an individual enable bit. For MacAddr1 to MacAddr31 addresses, you can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in the register. This enables group address filtering for the DA. The MacAddr32 to MacAddr127 addresses do not have mask control and all 6-bytes of the MAC address are compared with the received 6-bytes of DA.

Multicast Destination Address Filtering

To program the MAC to pass all multicast packets, set the PM bit in MAC_Packet_Filter register.

The multicast address is compared with the programmed MAC Destination Address registers (1–31). Group address filtering is also supported.

Broadcast Address Filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in MAC_Packet_Filter register.

Unicast Source Address Filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. You can configure the MAC Address registers[1–31] to use SA instead of DA for comparison by setting Bit 30 of corresponding register.

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in MAC_Packet_Filter register. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word. When the SAF bit is set, the SA filter and DA filter result is AND'ed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

Inverse Filtering

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of MAC_Packet_Filter register. The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of

the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 446 summarizes the DA and SA filtering based on the type of packets received.

Note: When the RA bit of MAC_Packet_Filter register is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx Status.

Table 446 Destination Address Filtering

Packet Type	PR	DAIF	PM	DBF	DA Filter Operation
Broadcast	1	X	X	X	Pass
	0	X	X	0	Pass
	0	X	X	1	Fail
Unicast	1	X	X	X	Pass all packets
	0	0	X	X	Pass on Perfect/Group filter match
	0	1	X	X	Fail on Perfect/Group filter match
Multicast	1	X	X	X	Pass all packets
	X	X	1	X	Pass all packets
	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	1	0	X	Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x

Packet Type	PR	SAIF	SAF	SA Filter Operation
Unicast	1	X	X	Pass all packets.
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

44.3.7.3 VLAN Filtering

44.3.7.3.1 VLAN Tag Perfect Filtering

The VLAN tag perfect filtering in DWC_ether_qos enables the MAC to forward VLAN-tagged packets along with VLAN tag match status and to drop the VLAN packets that do not match.

In VLAN tag perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of MAC_VLAN_Tag register. In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of MAC_VLAN_Tag register. The VLAN packet status bit (Bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results. By default, the VLAN tag-based perfect filter is available in all configurations.

44.3.7.3.2 Introduction to VLAN Tag Hash Filtering

The 16-bit VLAN Hash Table is used for group address filtering based on the VLAN tag.

The MAC provides VLAN tag hash filtering with a 16-bit Hash table.

44.3.7.3.3 VLAN Tag Hash Filtering

The 16-bit VLAN Hash Table is used for group address filtering based on the VLAN tag. The VLAN Tag Hash Filtering feature can be enabled using the VTHM (VLAN Tag Hash Table Match Enable) bit of the MAC_VLAN_Tag register.

The MAC provides VLAN tag hash filtering with a 16-bit Hash table.

The MAC performs the VLAN hash matching based on the VTHM of the MAC_VLAN_Tag register. If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

Notes

1. *The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in MAC_VLAN_TAG register.*
2. *When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of MAC_VLAN_Hash_Table register.*
3. *When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of MAC_VLAN_Hash_Table register.*

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the perfect or hash filter, the packet should be dropped. If the VLAN perfect and VLAN hash match are enabled, a packet is considered as matched if either the VLAN hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and hash filters indicate mismatch.

Table 447 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of MAC_Packet_Filter register is set, all packets are received and the VLAN match status is indicated in the VF bit of RDES2 Normal Descriptor (Write-Back Format). When the RA bit is not set and the VTFE bit is set in

MAC_Packet_Filter register, the packet is dropped if the final VLAN match status is Fail. In the following table, value X means that this column can have any value.

When VLAN VID is programmed to 0 in the VL field of MAC_VLAN_Tag register, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN hash match depends on the VTHM and VTIM bits in MAC_VLAN_Tag register.

Table 447 VLAN Match Status

VID	VLAN Perfect Filter Match Result	VTHM Bit	VLAN Hash Filter Match Result	VTIM Bit	Final VLAN Match Status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID!= 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

In **Table 447**, X represents any value.

44.3.7.4 Extended Rx VLAN Filtering and Routing

44.3.7.4.1 Introduction to Extended Receive VLAN Filtering

The Extended Receive VLAN Filtering feature enables both Perfect and Hash filtering in an application. The filter result is based on the perfect filter result and the Hash Filter result, and the result is passed to the application as part of the status bits.

Perfect filtering is done for the VLAN Tag Filters. For each VLAN Tag Filter, the VLAN Tag ID is compared. The combined result of all the enabled VLAN Tag Filters determines the overall VLAN Tag Filter result. The filter result is passed to the application as part of the status bits.

When the VLAN Tag Filter gives a pass result, the frame is considered to have passed the VLAN Tag Filters. If the frame fails all the filters, the frame is considered to have failed the VLAN filter.

In addition to Perfect Filtering, Hash Filtering can also be enabled. In Hash Filtering, the overall VLAN Filter result is determined by the Hash Filter result and the perfect filter result. The overall filter result determines if the frame will be dropped or forwarded to the application based on the programming on the VTFE and RA bits of the MAC Filter Register.

44.3.7.4.2 Comparision Modes

An application has several comparison options for each VLAN Tag Filter, and a pass or fail result can be obtained for the comparison.

For each VLAN Tag Filter, the application has the following comparison options:

- It can program the MAC to compare an outer VLAN Tag or an inner VLAN Tag with the programmed VID.
- It can choose if 12 or 16 bits of the VID field need to be compared.
- Type check can be disabled or enabled for each filter; if enabled, the application can choose if the VID comparison is for SVLAN or CVLAN type frames only.For example, if a filter is enabled for 16 bit comparison, SVLAN Type, and Outer VLAN Tag, any single or double VLAN Tagged frames with Outer SVLAN Tags are compared with this filter, and a pass or fail result is obtained.

Note: The inner VLAN Tag comparison is applicable only if Double VLAN Tag processing is enabled through the parameter and the MAC VLAN Control Register bit.

44.3.7.4.3 VLAN Filter Fail Packets Queue

When VLAN filtering is enabled (by setting the VTFE bit), the VLAN Filter Fail Packets can be routed to a programmable Queue (VFFQ) when Receive All (RA) is enabled and the enable bit (VFFQE) for the queue is set.

RA and VTFE bits are implemented in the MAC_Packet_Filter register.

VFFQ and VFFQE fields are implemented in the MAC_RxQ_Ctrl4 register.

The packets that pass the VLAN filtering are routed based on the VLAN TAG priority field. The VLAN Tag priorities can be assigned to Rx Queues by programming the PSRQ field in the corresponding MAC_RxQ_Ctrl2 register. The packets that fail the VLAN filtering are expected to be dropped, however when RA bit is set the VLAN filter fails are forwarded to the application. If the VLAN Filter Fails and Queue Enable (VFFQE) bit is selected then the VLAN filter fail packets are forwarded to the queue number programmed in VFFQ field, and if VFFQE is not enabled, the queue number is determined by the VLAN tag priority.

Rx Queue routing table for Unicast tagged packets:

Table 448 OTS and ITS Bit Values with At Least 1 Perfect Filter Enabled

RA	VTFE	SA/DA Filter Result	VLANFilter Result	VFFQE	Queue Routing
X	X	PASS	PASS	X	PSRQ
0	0	PASS	FAIL	0	PSRQ
0	0	PASS	FAIL	1	VFFQ
0	X	FAIL	X	X	DROPPED
0	1	PASS	FAIL	X	DROPPED
1	X	FAIL	X	0	UFFQ*/PSRQ
1	X	FAIL	X	1	UFFQ*/VFFQ
1	X	PASS	FAIL	0	PSRQ
1	X	PASS	FAIL	1	VFFQ

X : Don’t care condition

UFFQ* : When UFFQE is enabled

44.3.7.4.4 Filter Status

The Extended Receive VLAN Filtering & Routing feature provides two status bits to indicate the comparison result of the VLAN tags.

Currently the MAC indicates the VLAN Filter Status through one bit in the status – VF in RDES2. When Extended RX VLAN Filtering & Routing is enabled, two status bits are used to indicate the comparison result of VLAN tags. The Outer VLAN Tag Filter Pass and Inner VLAN Tag Filter Pass bits are defined in the following positions in various configurations. The status indicated through these bits is highly dependent on the programming as explained below.

In RDES2:

- Bit 15 – Outer VLAN Tag Filter Status
  - Bit 14 – Inner VLAN Tag FilterStatus

In ARI status: MAC Filter Status:

- Bit 15 – Outer VLAN Tag Filter Status
- Bit 14 – Inner VLAN Tag Filter Status

In MRI Status:

- Bit 47 – Outer VLAN Tag Filter Status
- Bit 46 – Inner VLAN Tag Filter Status

Outer VLAN Tag Filter Status (OTS)

- In perfect Filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame's Outer VLAN Tag has matched one of the VLAN Tag Filters.
- If this bit is reset, it indicates that the frame's Outer VLAN Tag has either failed the relevant Outer VLAN Tag Filters or bypassed them.
- If none of the filters are enabled for Outer VLAN Tag Comparison, then this bit is reset.
- If Inverse Filtering is enabled and this bit is set, then the frame's VLAN Tag has passed all the relevant VLAN Tag Filters. If it is reset, then it has failed at least one of or bypassed all the Filters programmed for Outer VLAN Tag Comparison.
- This bit is valid for both Single and Double VLAN Tagged frames.

Inner VLAN Tag Filter Status (ITS)

- In perfect Filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame's Inner VLAN Tag has matched one of the VLAN Tag Filters.
- If this bit is reset, it indicates that the frame's Inner VLAN Tag has either failed the relevant Inner VLAN Tag Filters or bypassed them.
- If none of the filters are enabled for Inner VLAN Tag Comparison, then this bit is reset.
- If Inverse Filtering is enabled and this bit is set, then the frame's VLAN Tag has passed all the relevant VLAN Tag Filters. If it is reset, then it has failed at least one of or bypassed all the Filtersprogrammed for Inner VLAN Tag Comparison.
- This bit is valid for only Double VLAN Tagged frames, when Double VLAN Processing is enabled.

The application must look at the status bits and the programming to determine if the Frame has passed or failed the VLAN Filter.

Table 449 and **Table 450** show the possible Filter combinations and the corresponding filter results. These tables explain the scenarios when Double VLAN Processing and Hash VLAN filter are enabled in the design.

Legend for the **Table 449**

- VTIM: VLAN Tag Inverse Match Enable – bit 17 in VLAN_Tag_Ctrl Register.
- HFO: Hash Filter enabled for Outer VLAN Tag Comparison . bit 25 and bit 27 in VLAN_Tag_Ctrl Register.
- HFI: Hash Filter enabled for Inner VLAN Tag Comparison – bit 25 and bit 27 in VLAN_Tag_Ctrl Register.
- PFO – Perfect Filter comparison enabled for Outer VLAN Tag - Any of the MAC_VLAN_Tag_Filter Registers is enabled (bit 16 is set) and programmed for Outer VLAN Tag comparison (bit 20 is set to 0).

- PFI – Perfect Filter comparison enabled for Inner VLAN Tag - Any of the MAC_VLAN_Tag_Filter Registers is enabled (bit 16 is set) and programmed for Inner VLAN Tag comparison (bit 20 is set to 1).
  OTS – Outer VLAN Tag Filter Status
  ITS – Inner VLAN Tag Filter Status

Table 449 OTS and ITS Bit Values with At Least 1 Perfect Filter Enabled

VTIM	HFO	HFI	PFO	PFI	OTS	ITS
0	0	0	0	1	0	1/0
0	0	0	1	0	1/0	0
0	0	0	1	1	1/0	1/0
0	1	0	1	1	1/0	1/0
0	1	0	1	0	1/0	0
0	1	0	0	1	1/0	1/0
0	0	1	1	1	1/0	1/0
0	0	1	1	0	1/0	1/0
0	0	1	0	1	0	1/0
1	0	0	0	1	0	1/0
1	0	0	1	0	1/0	0
1	0	0	1	1	1/0	1/0
1	1	0	1	1	1/0	1/0
1	1	0	1	0	1/0	0
1	1	0	0	1	1/0	1/0
1	0	1	1	1	1/0	1/0
1	0	1	1	0	1/0	1/0
1	0	1	0	1	0	1/0

Table 450 shows the possible values of status bits (OTS and ITS) when none of the perfect filters are enabled and only the VLAN Hash Filter is enabled.

Table 450 OTS and ITS Bit Values with Only VLAN Hash Filter Enabled

VTIM	HFO	HFI	OTS	ITS
0	0	0	0	0
0	1	0	1/0	0
0	0	1	1/0	1/0
1	0	0	1/0	0
1	1	0	1/0	0
1	0	1	1/0	1/0

With no perfect filters enabled, any VLAN packet is considered to have bypassed the perfect filter. If Hash Filter is enabled for one of the Tags, then the respective Status bit depends on the Filter's result. The Status bits are set to 0 if VLAN Hash Filter is also not enabled.

If the value of ITS/OTS is shown as 1/0; then it indicates that the final result is dependent on the enabled relevant filter's result.

Example 1: The second row of table 1-1 indicates that at least one Perfect Filter is enabled for Outer VLAN tag comparison and none of the filters are enabled for Inner VLAN tag comparison. Inverse VLAN Filtering is not enabled. The bit OTS is given as 1/0. If the received frame passes atleast one of the enabled Outer VLAN Tag filters then the bit is set to 1. If the frame doesn't pass any of the enabled Outer VLAN Tag filters, then the bit is set to 0.

Example 2: Last Row of Table 1-1 indicates that Inverse Filtering is enabled, Hash Filter and atleast one perfect filter is enabled for Inner VLAN Tag comparison, then if the received frame's Inner VLAN tagmismatches with both the Hash Filter and all the enabled Perfect filters, then the frame will have the ITS bitset to 1 else it is set to 0. OTS will be set to 0 as no comparison is performed.

44.3.7.4.5 Stripping

DWC_ether_qos supports the stripping feature in which the stripping operation is enabled on Inner VLAN Tags in received packets. This field indicates the stripping operation on inner VLAN Tag in received packet:

Each of the VLAN Tags has individual control over stripping. The programming options of Always strip, never strip, strip on pass and strip on fail are available. Inner or Outer VLAN Tag Stripping is based on the pass or fail results of the individual tag. If a tag is bypassed by all the relevant filters, stripping is not applicable for the tag.

- If strip on Pass is enabled for the outer VLAN Tag, then the stripping occurs only if the Outer VLAN tag has passed the relevant Filters. The Outer VLAN Tag Filter Result bit will be set.
- If strip on Fail is enabled for the outer VLAN Tag, then stripping occurs only if the Outer VLAN Tag has failed relevant filters. The Outer VLAN Tag Filter Result Bit will be reset.
- If the Outer VLAN tag of the received frame is bypassed by the entire filter (no comparison has been made), then the tag is not stripped, though the Status Bit is still 0.

As multiple filters are enabled, it is possible that the received VLAN frame could have matched any one or more of the filters. The VLAN Tag's value is not always deterministic from the filter status bits.

If the application strips the VLAN Tag based on the filter result, it might lose the VID. So the suggested use is, if Stripping is enabled for any of the tags, the tag can be put in the status. For this the application will have to enable the respective "VLAN Tag in Status" bit - 24 or 31 in the MAC VLAN Tag Control Register.

44.3.8 Using IEEE 1588 Timestamp Support

44.3.8.1 Using IEEE 1588 Timestamp Support

The IEEE 1588 defines a Precision Time Protocol (PTP) which enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects.

The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

44.3.8.1.1 Introduction to IEEE 1588 Timestamp Support

The IEEE 1588-2002 supports PTP transported over UDP/IP and IEEE 1588-2008 supports PTP transported over Ethernet.

The supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP and IEEE 1588-2008 supports PTP transported over Ethernet. The ; provides programmable support for both standards.

The controller supports the following features:

- Supports both timestamp formats
- Provides an option to take snapshot of all packets or only PTP type packets
- Provides an option to take snapshot of only event messages
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Provides an option to select the node to be a master or slave for ordinary and boundary clock
- Identifies the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Provides an option to measure sub-second time in digital or binary format

44.3.8.1.2 Description of IEEE 1588 Timestamp Support

Clock Types

 supports different clock types defined in IEEE 1588-2008.

The ; supports the following clock types:

- Ordinary Clock
- Boundary Clock
- End-to-End Transparent Clock
- Peer-to-Peer Transparent Clock

Ordinary Clock

The ordinary clock has a single PTP state and a single physical port. In a domain, an ordinary clock supports a single copy of the protocol.

The ordinary clock in a domain supports a single copy of the protocol. The ordinary clock has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

- Sends and receives PTP messages. The timestamp snapshot can be controlled as described in MAC_Timestamp_Control.
- Maintains the data sets such as timestamp values

Table 451 shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

Table 451 Ordinary Clock: PTP messages for Snapshot

Master	Slave
Delay_Req	SYNC

For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in MAC_Timestamp_Control.

Boundary Clock

A boundary clock, typically has several physical ports communicating with the network.

The boundary clock typically has several physical ports communicating with the network. The messages related to synchronization, master-slave hierarchy, and signaling terminate in the protocol engine of the boundary clock and such messages are not forwarded. The PTP message type status given by the MAC helps you to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

- The clock data sets are common to all ports of the boundary clock.
- The local clock is common to all ports of the boundary clock.

End-to-End Transparent Clock

End-to-End transparent clock supports end-to-end delay measurement mechanism between the slave clocks and the master clock.

The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the Ingress port to the Egress port.

The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot must be taken at both Ingress and Egress ports only for the messages mentioned in **Table 452**. You can take the snapshot by setting the SNAPTYPSEL bits to 10 in the MAC_Timestamp_Control register.

Table 452 End to end Transparent Clock: PTP messages for Snapshot

PTP Messages
SYNC
Delay_Req

Peer-to-Peer Transparent Clock

In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer.

The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in **Table 453**.

Table 453 Peer-to-Peer Transparent Clock: PTP Messages for Snapshot

PTP Messages
SYNC
Pdelay_req
Pdelay_Resp

You can take the snapshot by setting the SNAPTYPESEL bit to 11 in MAC_Timestamp_Control register.

Peer-to-Peer PTP Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 supports peer-to-peer PTP (Pdelay) message in addition to the SYNC, Delay Request, Follow-up, and Delay Response messages.

Figure 700 shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

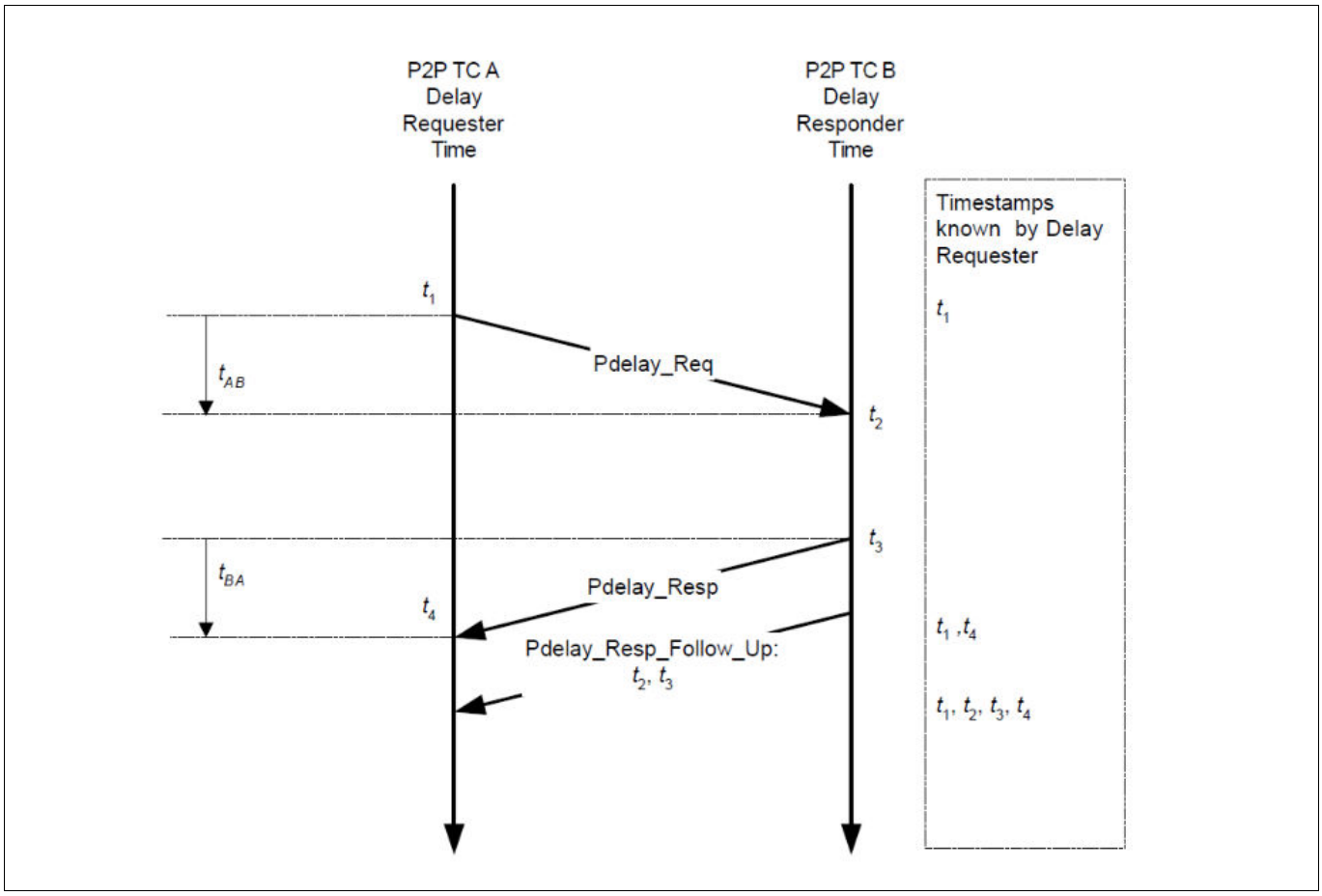


Figure 700 Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction

As shown in **Figure 700**, the propagation delay is calculated in the following way:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t1) for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a timestamp (t2) for this message.
3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t3) for this message.
   To minimize errors because of any frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:
   a) Difference between the timestamps t2 and t3 in the Pdelay_Resp message
   b) Difference between the timestamps t2 and t3 in the Pdelay_Resp_Follow_Up message
   c) Timestamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively
4. Port 1 generates a timestamp (t4) on receiving the Pdelay_Resp message.
5. Port 1 uses all four timestamps to compute the mean link delay.

Timestamp Correction

According to the IEEE 1588 specification, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network.

Because the reference timing source (the PTP clock clk_ptp_ref_i) is different from the MAC Tx or Rx clock, the captured timestamp must be corrected for latency issues because of synchronization. In addition, latency issues between the internal snapshot point and the recommended capture point (the boundary between the node and the network), must also be corrected.

Ingress Correction

In the Receive side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (Ingress Correction Value) programmed in the Ingress Correction register.

The value that needs to be programmed in the ingress correction register is calculated as follows:

The timestamp correction because of synchronization is compensated by adding INGRESS_SYNC_CORR to the synchronized timestamp value as follows:

```
INGRESS_SYNC_CORR = -(2*PTP_CLK_PER)
```

The latency correction between the message timestamp point and the internal timestamp snapshot point is done by subtracting the latency value (INGRESS_LATENCY) with the captured timestamp as follows:

```
Ingress Correction = INGRESS_SYNC_CORR - INGRESS_LATENCY
```

Ingress correction is performed by programming the TSIC field in the MAC Timestamp Ingress correction register. The ingress correction is always negative and has a unit of nanoseconds. The value is represented in complement form as follows:

When TSCTRLSSR bit in MAC_Timestamp_Control register is set, this has an accuracy of 1 ns.

It is represented by setting bit 31 to 1 and bits 30:0 containing 10^9 - <ingress_correction_value> represented in binary. For example, if the required correction value is -5 ns, then the programmed value is 0xBB9A_C9FB

When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 ns. It is represented by setting bit 31 to 1 and bits 30:0 containing 2^31 - <ingress_correction_value> represented in binary.

Egress Correction

In the Transmit side the timestamp captured at the internal snapshot point is corrected for latency and synchronization by adding the correction value (Egress Correction Value) programmed in the Egress Correction register.

The value to be programmed in the egress correction register is calculated as follows:

The timestamp correction because of synchronization is compensated by adding EGRESS_SYNC_CORR to the synchronized timestamp value as follows:

When Enable one step timestamp feature is selected,

```
EGRESS_SYNC_CORR = (1*PTP_CLK_PER + 4*TX_CLK_PER)
```

Otherwise,

```
EGRESS_SYNC_CORR = -(2*PTP_CLK_PER)
```

The egress latency correction between the recommended capture point and the internal timestamp snapshot point is done by adding the latency value (EGRESS_LATENCY) with the captured timestamp as follows:

```
Egress Correction = EGRESS_SYNC_CORR + EGRESS_LATENCY
```

Egress correction is performed by programming the TSEC field in the MAC Timestamp Egress correction register. The egress correction can be positive or negative and has a unit of nanoseconds. Negative values are represented in complement form as follows:

When TSCTRLSSR bit in MAC_Timestamp_Control register is set, this has an accuracy of 1 ns.

If the correction is positive, it is represented by setting bit 31 to 0 and bits 30:0 containing <egress_correction_value> represented in binary. The value must not exceed 0x 3B9A_C9FF. If the correction is negative, it is represented by setting bit 31 to 1 and bits 30:0 containing 10^9 - <egress_correction_value> represented in binary.

For example, if the required correction value is -5 ns, then the programmed value should be 0xBB9A_C9FB

When TSCTRLSSR bit in MAC_Timestamp_Control register is reset, this has an accuracy of ~0.466 ns. If the correction is positive, it is represented by setting bit 31 to 0 and bits 30:0 containing <egress_correction_value> represented in binary. The maximum value is 0x7FFF_FFFF. If the correction is negative, it is represented by setting bit 31 to 1 and bits 30:0 containing 2^31 - <egress_correction_value> represented in binary.

Frequency Range of Reference Timing Clock

The timestamp information is transferred across asynchronous clock domains, from the MAC clock domain to the application clock domain.

Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is 4 clock cycles of GMII or MII and 3 clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

Maximum PTP Clock Frequency

The maximum PTP clock frequency is limited by the maximum resolution of the reference time (1 ns resulting in 1 GHz) and the timing constraints achievable for logic operating on the PTP clock.

In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

Minimum PTP Clock Frequency

The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time to the GMII or MII clock domain.

This relationship is given in the following equation:

```
3 * PTP clock period + 4 * GMII/MII clock period <= Minimum gap between two SFDs
```

The GMII or MII clock frequency is fixed by IEEE specification. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in **Table 454**.

Notes

1. *It is recommended that you use a clock that has constant frequency, preferably a divided application clock.*
2. *When IEEE 1588 timestamp feature is enabled with internal timestamp, use a PTP clock frequency which is greater than 5 MHz. This is because the 8-bit MAC_Sub_Second_Increment register limits the minimum PTP frequency that can be used to ~4 MHz.*

Table 454 Minimum PTP Clock Frequency Example

Mode	Minimum Gap Between Two SFDs	Minimum PTP Frequency with External Timestamp Input	Minimum PTP Frequency with Internal Timestamp
10 Mbps full duplex	168 MII clocks(128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	~45 KHz	5 MHz
10 Mbps half duplex	48 MII clocks(8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	170 KHz	5 MHz
100 Mbps full duplex	168 MII clocks(128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	~0.5 MHz	5 MHz
100 Mbps half duplex	48 MII clocks(8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	4.55 MHz	5 MHz
1000 Mbps full duplex	84 GMII clocks(64 clocks for a 64-byte packet + 12 clocks of min IFG + 8 clocks of preamble)	~4.7 MHz	5 MHz
1000 Mbps half duplex	24 GMII clocks(4 clocks for a JAM pattern sent just after SFD because of collision + 12 IFG + 8 preamble)	18.75 MHz	18.75 MHz

PTP Processing and Control

The IEEE 1588-2008 specification supports peer-to-peer (PTP) messages in addition to the SYNC, Delay Request, Follow-up, and Delay Response messages. This topic discuses the common message header format as defined in IEEE 1588-2008.

Table 455 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008.

Table 455 Message Format Defined in IEEE 1588-2008

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField (*)								1	32
logMessageInterval								1	33

(*) – control Field is used in version 1. In version 2, message Type field is used for detecting different message types.

There are some fields in the Ethernet payload that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP Packets over IPv4
- PTP Packets over IPv6
- PTP Packets over Ethernet

PTP Packets over IPv4

The Ethernet payload contains certain fields that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for PTO packets over IPv4 and ethernet, and framers over IPv6 packets. This topic discusses fields that are matched to control snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2.

Table 456 provides information about the fields that are matched to control snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4.

Table 456 IPv4-UDP PTP Packet Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Packet Type	12, 13	0x0800	IPv4 datagram
IP version and Header Length	14	0x45	IP version is IPv4
Layer 4 Protocol	23	0x11	UDP
IP Multicast Address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81(or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed: • 224.0.1.129 • 224.0.1.130 • 224.0.1.131 • 224.0.1.132
IP Multicast Address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex)0xE0, 0x00, 0x00, 0x6B (Hex)	• PTP Primary multicast address: 224.0.1.129 • PTP Pdelay multicast address: 224.0.0.107
UDP Destination Port	36, 37	0x013F, 0x0140	• 0x013F: PTP event messages • 0x0140: PTP general messages
PTP Control Field (IEEE 1588 version 1)	74	0x00, 0x01, 0x02, 0x03, 0x04	• 0x00: SYNC • 0x01: Delay_Req • 0x02: Follow_Up • 0x03: Delay_Resp • 0x04: Management

Table 456 IPv4-UDP PTP Packet Fields Required for Control and Status (cont'd)

Field Matched	Octet Position	Matched Value	Description
PTP Message Type Field (IEEE 1588 version 2)	42 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	• 0x0: SYNC • 0x1: Delay_Req • 0x2: Pdelay_Req • 0x3: Pdelay_Resp • 0x8: Follow_Up • 0x9: Delay_Resp • 0xA: Pdelay_Resp_Follow_Up • 0xB: Announce • 0xC: Signaling • 0xD: Management
PTP Version	43 (nibble)	0x1 or 0x2	• 0x1: Supports PTP version 1 • 0x2: Supports PTP version 2

a. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only)

This is based on the IEEE 1588-2008, Annex D and the message format defined in the following table.

Table 457 Message Format Defined in IEEE 1588-2008

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField (*)								1	32
logMessageInterval								1	33

PTP Packets over IPv6

The Ethernet payload contains certain fields that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for PTO packets over IPv4 and ethernet, and framers over IPv6 packets. This topic discusses fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2.

Table 458 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4.

Table 458 IPv6-UDP PTP Packet Fields Required for Control and Status

Field Matched	Octet Position	Matched Value	Description
MAC Packet Type	12, 13	0x86DD	IP datagram
IP Version	14 (Bits [7:4])	0x6	IP version is IPv6
Layer 4 Protocol	20[1] (*)	0x11	UDP
PTP Multicast Address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	• PTP Primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) • PTP Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP Destination Port	56, 57a	0x013F, 0x140	• 0x013F: PTP event message • 0x0140: PTP general messages
PTP Control Field (IEEE 1588 version 1)	94a	0x00, 0x01, 0x02, 0x03, or 0x04	• 0x00: SYNC • 0x01: Delay_Req • 0x02: Follow_Up • 0x03: Delay_Resp • 0x04: Management (version1)
PTP Message Type Field (IEEE 1588 version 2)	62a (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	• 0x0: SYNC • 0x1: Delay_Req • 0x2: Pdelay_Req • 0x3: Pdelay_Resp • 0x8: Follow_Up • 0x9: Delay_Resp • 0xA: Pdelay_Resp_Follow_Up • 0xB: Announce • 0xC: Signaling • 0xD: Management
PTP Version	63 (nibble)	0x1 or 0x2	• 0x1: Supports PTP version 1 • 0x2: Supports PTP version 2

1) The Extension Header is not defined for PTP packets.

This is based on the IEEE 1588-2008, Annex D and the message format defined in the following table.

Table 459 Message Format Defined in IEEE 1588-2008

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6

Table 459 Message Format Defined in IEEE 1588-2008 (cont'd)

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField (*)								1	32
logMessageInterval								1	33

PTP Packets over Ethernet

The Ethernet payload contains certain fields that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for PTO packets over IPv4 and ethernet, and framers over IPv6 packets. This topic discusses fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2.

Table 460 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format.

Table 460 Ethernet PTP Packet Fields Required for Control And Status

Field Matched	Octet Position	Matched Value	Description
MAC Destination Multicast Address[1)]	0–5	01-1B-19-00-00-0001-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses[2)]: • 01-1B-19-00-00-00 • 01-80-C2-00-00-0E[3)]
MAC Packet Type	12, 13	0x88F7	PTP Ethernet packet
PTP Control Field(IEEE 1588 version 1)	46	0x00, 0x01, 0x02, 0x03, or 0x04	• 0x00: SYNC • 0x01: Delay_Req • 0x02: Follow_Up • 0x03: Delay_Resp • 0x04: Management

Table 460 Ethernet PTP Packet Fields Required for Control And Status (cont'd)

Field Matched	Octet Position	Matched Value	Description
PTP Message Type Field (IEEE 1588 version 2)	14 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	• 0x0: SYNC • 0x1: Delay_Req • 0x2: Pdelay_Req • 0x3: Pdelay_Resp • 0x8: Follow_Up • 0x9: Delay_Resp • 0xA: Pdelay_Resp_Follow_Up • 0xB: Announce • 0xC: Signaling • 0xD: Management
PTP Version	15 (nibble)	0x1 or 0x2	• 0x1: Supports PTP version 1 • 0x2: Supports PTP version 2

1) The unicast address match of destination addresses (DA), programmed in MAC address 0 to 31, is used if the TSENMACADDR bit of MAC_Timestamp_Control register is set.
2) IEEE 1588-2008, Annex F
3) The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

Transmit Path Functions

The MAC captures a timestamp when the Start Packet Delimiter (SFD) of a packet is sent on the GMII or MII interface. The packets, for which the timestamps has to be captured can be controlled on per-packet basis.

Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. You need to specify the packets for which you want to capture timestamps. You can do this by using the following method based on your configuration:

- EQOS-AHB, EQOS-AXI, and EQOS-DMA Configurations
  You can specify the packets by using the control bits in the Transmit descriptor. The MAC returns the timestamp to the software inside the corresponding Transmit descriptor, thus connecting the timestamp automatically to the specific PTP packet.
  The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

Receive Path Functions

The MAC can be programmed to capture the timestamp of all packets received on the GMII or MII interface or to process packets to identify the valid PTP messages.

Use the following options of the MAC_Timestamp_Control register to control the snapshot of the time to be sent to the application:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)

- Enable the node to be a master or slave and select the snapshot typeThis feature controls the type of messages for which snapshots are taken.

Note: The DWC_ether_qos also supports the PTP messages over VLAN packets.

Table 461 Ethernet PTP Packet Fields Required for Control And Status

SNAPTYPSEL	TSMSTRENA	TSEVNTENA	PTP Messages
00	x	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00	0	1	SYNC
00	1	1	Delay_Req
01	x	0	SYNC, Follow_Up, Delay_Req, Delay_Resp,Pdelay_Req, Pdelay_Resp,Pdelay_Resp_Follow_Up
01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	x	x	SYNC, Delay_Req
11	x	x	Pdelay_Req, Pdelay_Resp

The following list describes how the timestamp is provided, depending on the configuration:

- EQOS-AHB, EQOS-AXI, and EQOS-DMA Configurations
  The DMA returns the timestamp to the software inside the corresponding Receive Descriptor. The extended status, containing the timestamp message status and the IPC status, is written in normal descriptor RDES1 and the snapshot of the timestamp is written in RDES0 and RDES1 fields of context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

44.3.8.2 Using IEEE 1588 System Time Source

44.3.8.2.1 Introduction to IEEE 1588 Time Source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

The internal reference time provides the following options for using the reference timing source in a node:

- External Timestamp Input
  This option takes an external 64-bit timing reference and its clock as input. The clock input is used to synchronize the timing reference to the MAC clock domain.
- Internal Reference Time (80-bit)
  This option takes only the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps.

44.3.8.2.2 Description of IEEE 1588 Time Source

External Timestamp Input

External timestamp input option takes an external 64-bit timing reference and its clock as input. The clock input is used to synchronize the timing reference to the MAC clock domain.

The 64-bit timing reference is split into the following two 32-bit signals:

- Upper 32-bits providing the time in seconds
- Lower 32-bits providing the time in nanoseconds

This timing reference is used to timestamp the packets.

Internal Reference Time

Internal reference time option takes the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps.

The timestamp has the following fields:

- UInteger48 seconds Field
  The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bits wide. For example, 2.000000001 seconds are represented as seconds Field = 0x0000_0000_0002.
- UInteger32 nanosecondsField
  The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 nanoseconds are represented as nanoSeconds = 0x0000_0001.
  The nanoseconds field supports the following two modes:
  - Digital rollover mode: In this mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds.
  - Binary rollover mode: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is ~0.466 ns per bit.

System Time Register Module

The system time generator module is an optional module. It is not available if external time updating is enabled.

The 64-bit time is maintained in this module and updated using the input reference clock (clk_ptp_ref_i). This time is the source for taking snapshots (timestamps) of Ethernet packets being transmitted or received at the GMII interface.

The system time counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the Timestamp Update register. For initialization, the system time counter is written with the value in the Timestamp Update register. For system time correction, the offset value is added to or subtracted from the system time.

In the fine correction method, the frequency drift of a slave clock (clk_ptp_ref_i) with respect to the master clock (as defined in IEEE 1588-2002) is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register, as shown in **Figure 701**. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

This algorithm is shown in **Figure 701**.

Note: You must connect a PTP clock with a frequency higher than the frequency required for the specified accuracy.

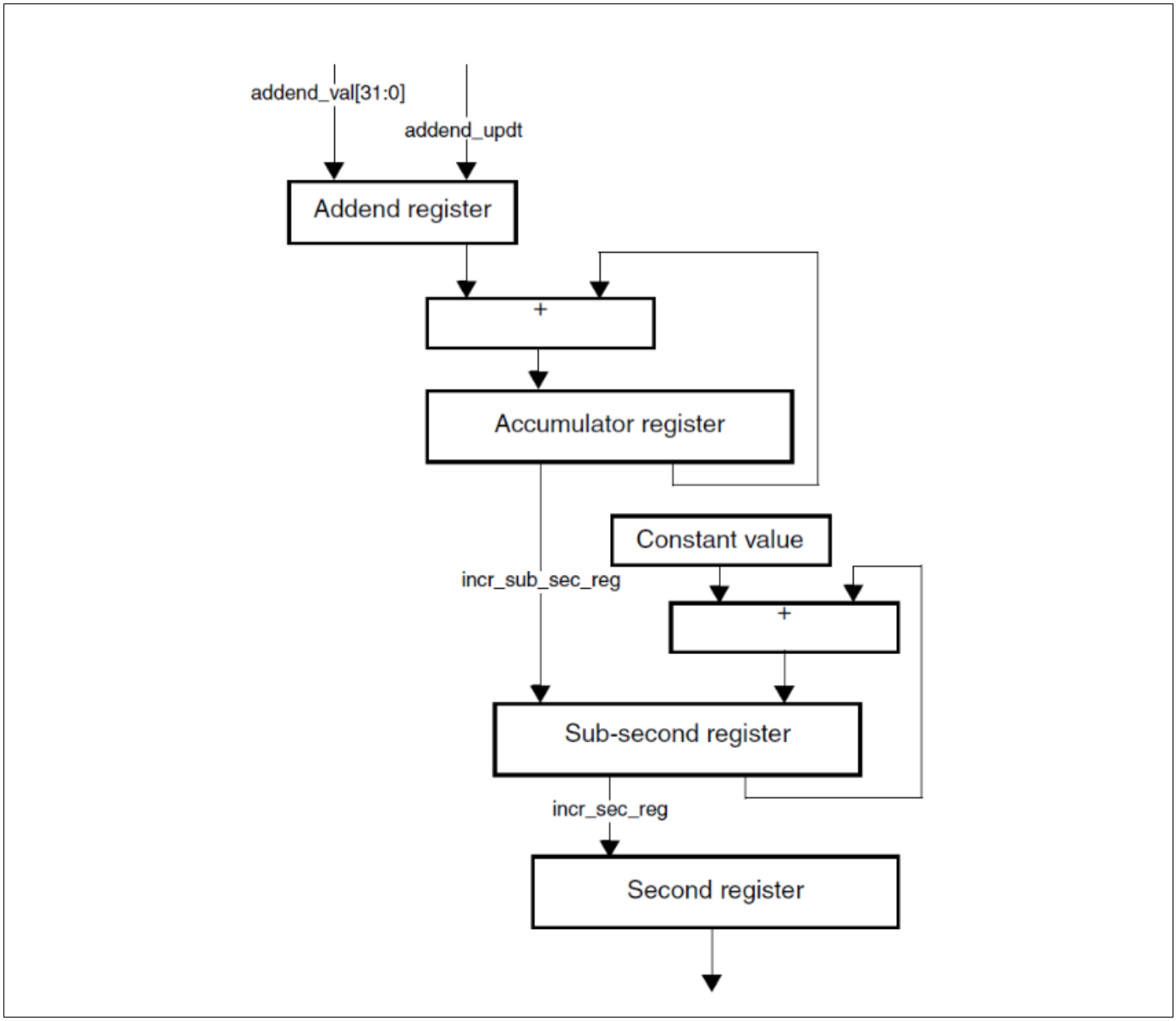


Figure 701 System Time Update Using Fine Method

The System Time Update logic requires a 50-MHz clock frequency to achieve 20-ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz / 50 MHz = 1.32. Therefore, the default addend value to be set in the register is 2^32 / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, or 1.3 and the value to set in the addend register is 2^32 / 1.30, or 0xC4EC4EC4. If the clock drifts higher, for example, to 67 MHz, the addend register must be set to 0xBF0B7672. When the clock drift is nil, the default addend value of 0xC1F07C1F (2^32 / 1.32) must be programmed.

In **Figure 701**, the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20 ns steps). When External Time Update is enabled, the optional System Time module is not available. Two different methods are used to update the System Time register depending on the configuration.

The software must calculate the drift in frequency based on the Sync messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

FreqCompensationValue0 = 2^32 / FreqDivisionRatio

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

- At time MasterSyncTimen the master sends the slave clock a Sync message. The slave receives this message when its local clock is SlaveClockTimen and computes MasterClockTimen as
  - $MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$
- The master clock count for current Sync cycle, MasterClockCountn is
  - $MasterClockCount_n = MasterClockTime_n – MasterClockTime_{n-1}$ (assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)
- The slave clock count for current Sync cycle, $SlaveClockCount_n$ is
  - $SlaveClockCount_n = SlaveClockTime_n – SlaveClockTime_{n-1}$
- The difference between master and slave clock counts for current Sync cycle, $ClockDiffCount_n$ is
  - $ClockDiffCount_n = MasterClockTime_n – SlaveClockTime_n$
- The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is
  - $FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$
- The frequency compensation value for Addend register, $FreqCompensationValue_n$ is
  - $FreqCompensationValue_n = FreqScaleFactor_n * FreqCompensationValue_{n-1}$

In theory, this algorithm achieves lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value from the master, the algorithm corrects it at the cost of more Sync cycles.

44.3.8.3 Using IEEE 1588 Higher Word Register

44.3.8.3.1 Introduction to IEEE 1588 Higher Word Register

The timestamp maintained in the MAC is 64-bit wide. The overflow to upper 16-bits of seconds register happens once in 130 years. The values of the upper 16-bits of the seconds field can be read from the CSR register.

This optional feature is provided to invoke an additional Higher Word Register.

44.3.8.4 Using Flexible Pulse-Per-Second Output

44.3.8.4.1 Introduction to Flexible Pulse-Per-Second Output

The GETH provides the flexibility to program the start or stop time, width, and interval of the pulse generated on the ptp_pps_o output.

Notes

1. *By default, is in the “Fixed Pulse-Per-Second Output” mode and indicates 1 second interval.*
2. *The frequency of the PPS output can be changed by setting the PPSCTRL0 field in the MAC_PPS_Control register.*

44.3.8.4.2 Description of Flexible Pulse-Per-Second Output

GETH provides features such as programming the start or stop time, with the flexible PPS output option.

GETH supports the following features with the flexible PPS output option:

- Programming the start or stop time in terms of system time.
- Programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time.
- Programming the stop time in advance, that is, you can program the stop time before the actual start time has elapsed.
- Programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of sub-second increment value programmed in the MAC_Sub_Second_Increment register. You can program the width of pulse from 1 to 2^32-1 units of sub-second increment value.
- Programming the interval, between the rising edges of PPS signal, in terms of number of units of sub-second increment value. You can program the interval between pulses from 1 to 2^32-1 units of sub-second increment value.
- Option to cancel the programmed PPS start or stop request.
- Error if the start or stop time being programmed has already elapsed.

Note: The PTP Reference clock mentioned in the following sections is the clock at which the system time gets updated. When the TSCFUPDT bit of MAC_Timestamp_Control register is set to 0, this clock is similar to

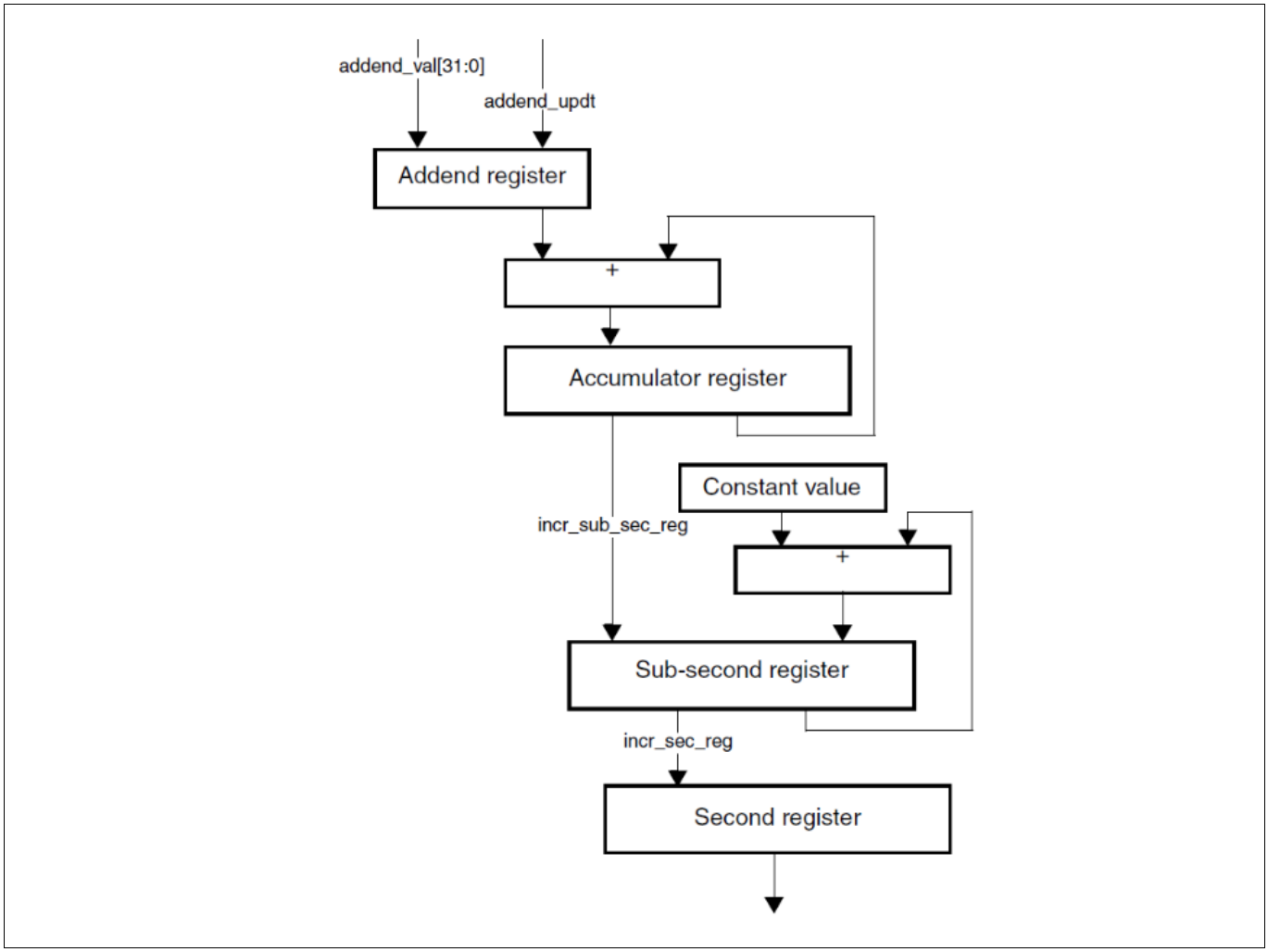


Figure 702 System Time Update Using Fine Method

PPS Start or Stop Time

The start time in the Target Time registers can be programmed initially.

You can initially program the start time in the Target Time registers. If you have enabled additional flexible PPS outputs, you need to program the following registers corresponding to an enabled flexible PPS output:

- MAC_PPS1_Target_Time_Seconds and MAC_PPS1_Target_Time_Nanoseconds registers for second Flexible PPS output
- MAC_PPS2_Target_Time_Seconds and MAC_PPS2_Target_Time_Nanoseconds registers for third Flexible PPS output
- MAC_PPS3_Target_Time_Seconds and MAC_PPS3_Target_Time_Nanoseconds registers for fourth Flexible PPS output

If required, you can again program the start or stop time but you can do it only after the earlier programmed value is synchronized to the PTP clock domain. Bit 31 of MAC_PPS#_Target_Time_Nanoseconds register indicates that the synchronization is complete. This enables you to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, you should program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can

cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

PPS Width and Interval

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of sub-second increment value.

For example, to have a PPS pulse width of 40 ns and interval of 100 ns with PTP reference clock of 50 MHz, you should program the width and interval to values 2 and 5, respectively. You can achieve smaller granularity by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, you should program or update the interval and width of the PPS signal output.

44.3.8.5 Using One-Step Timestamp Feature

44.3.8.5.1 MAC Transmit PTP Mode

MAC updates different fields of the Transmit PTP packets.

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- correctionField in the PTP header of messages
- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 462 shows how the PTP mode is selected based on the settings of SNAPTYPSEL, TSMSTRENA, and TSEVNTENA bits of the MAC_Timestamp_Control register and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 462 MAC_Transmit_PTP_Mode_and_One-Step_Timestamping_Operation

Programming			Mode	Per Packet Control 1)			Messages Processed on Tx
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE 2)	OSTC 3)	TTS 4)	
X	X	X	N/A	1	X	X	Timestamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)
2'b00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
							Delay_Req (correction field for residence time and Egress Asym Cor)

Table 462 MAC_Transmit_PTP_Mode_and_One-Step_Timestamping_Operation (cont'd)

Programming			Mode	Per Packet Control 1)			Messages Processed on Tx
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE 2)	OSTC 3)	TTS 4)	
2'b00	0	1	Ordinary or Boundary Slave	1	1	X	Delay_Req (originTimestamp field)Delay_Req (correction field for Egress Asym cor)
2'b00	1	1	Ordinary or Boundary Master	0	1	X	Sync (originTimestamp field)Sync (correction field for sub-nanosecond cor)
2'b01	X	0	End-to-End Transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
						Ingress TS	Pdelay_Req (correction field for residence time and Egress Asym Cor)
						Ingress TS	Pdelay_Resp (correction field for residence time and Ingress Asym Cor)
2'b01	0	1	Ordinary or Boundary Slave with support for peer delay mechanismorPeer to Peer Transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)(applicable only for Peer to Peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field)Delay_Req (correction field for Egress Asym cor)
				1	1	X	Pdelay_Req (originTimestamp field)Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)

Table 462 MAC_Transmit_PTP_Mode_and_One-Step_Timestamping_Operation (cont'd)

Programming			Mode	Per Packet Control 1)			Messages Processed on Tx
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE 2)	OSTC 3)	TTS 4)	
2'b01	1	1	Ordinary or Boundary Master with support for peer delay mechanism	0	1	X	Sync (originTimestamp field)Sync (correction field for sub-nanosecond cor)
				1	1	X	Pdelay_Req (originTimestamp field)Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)
2'b10	X	X	End-to-End Transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
						Ingress TS	Delay_Req (correction field for residence time and Egress Asym Cor)
2'b11	X	X	Peer-to-Peer Transparent	0	1	Ingress TS	Sync (correction field for residence time and Ingress Asym cor)
				1	1	X	Pdelay_Req (originTimestamp field)Pdelay_Req (correction field for Egress Asym Cor)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and Ingress Asym Cor)

1) The per packet control values provided here are the recommended settings used by devices in typical PTP operation, for the programmed mode.

2) TTSE represents TTSE bit of transmit descriptor in EQOS-AXI configuration, TTSE bit in TX control word in EQOS-MTL configuration, mti_ena_timestamp_i signal in EQOS-CORE configuration. The TTSE function is independent of the OST function and the programmed operation mode for OST. The MAC captures and returns the timestamp when the TTSE bit is set.

3) OSTC represents OSTC bit of transmit descriptor in EQOS-AXI configuration, OSTC bit in TX control word in EQOS-MTL configuration, mti_ost_en_i signal in EQOS-CORE configuration.

4) TTS represents the timestamp value provided in the TTSH, TTSL fields of transmit descriptor in EQOS-AXI configuration, timestamp provided in the TTSH, TTSL fields in the TX control word in EQOS-MTL configuration, mti_ost_i signal in EQOS-CORE configuration.

Note: Residence time/ turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp.When sub-nanosecond feature is enabled, residence time calculation includes sub-nanosecond accuracy.Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

44.3.8.6 One-Step Time Stamping for PTP Over UDP

44.3.8.6.1 Introduction to One-Step Timestamping for PTP Over UDP Feature

One step timestamping for PTP over UDP feature is available in configuration where IEEE 1588 Timestamp Support option is selected. DWC_ether_qos supports one-step timestamp operation for PTP messages sent over UDP/IPv4 and UPD/IPv6. DWC_ether_qos supports the one step timestamp operation for PTP messages sent over UDP/IPv4 and UDP/IPv6. Because origin timestamp and/or correction field (based on the PTP message type and the programmed PTP mode) in the PTP message are updated as part of the one step timestamp operation, it also requires update to the checksum field of the PTP message sent over UDP/IP.

44.3.8.6.2 Checksum Update for One-Step Timestamping for PTP

DWC_ether_qos, updates checksum for a packet, in the MTL before the packet is sent to the MACwhere the PTP message is updated for one step timestamp operation.

In DWC_ether_qos, checksum is updated for a packet, in the MTL before the packet is sent to the MAC where the PTP message is updated for one step timestamp operation. PTP requires the timestamp to be captured while the packet (SFD of the packet) is on the line and one step timestamp requires the origin timestamp and/or correction field to be updated in the PTP header while the packet is being transmitted on the line. Because the PTP message is present in the UDP payload, it is not possible to update the checksum field in the UDP header for modifications made to the PTP message, because it is already transmitted by then. Instead, the following options are supported that are in line with specification mentioned in Annex D & E of IEEE 1588-2008:

- When PTP message is sent over UDP/IPv4, the UDP checksum may be set to 0. The application must set the UDP checksum to 0 and set CIC to 0 or 1 while providing the packet to the MAC and the MAC does not update checksum for changes in origin timestamp or correction field.
- When PTP message is sent over UDP/IPv6, a transmitting node extends the UDP payload of all PTP messages by 2 octets beyond the end of the PTP message. The contents of the UDP checksum field or the final 2 octets of the UDP payload might be modified to ensure that the UDP checksum remains intact after modification of PTP fields. The application must add the two extra bytes at the end of the PTP message while providing the packet to the MAC and the MAC updates these two bytes to keep the checksum correct.

A programmable option (CSC field in MAC_Timestamp_Control register) supports UDP with non-zero checksum for IPv4. When set, the two bytes at the end of the PTP message are updated to keep the UDP checksum correct, similar to PTP over UDP/IPv6. So, the application must add two bytes at the end of the PTP message sent to the DWC_ether_qos and DWC_ether_qos updates these two bytes to keep the checksum correct.

Updating the pad bytes to keep the checksum correct requires the old value of correction field and/or origin timestamp in the packet. To get better latency and area, the application can set the origin timestamp field to zero, for PTP message in particular modes in which DWC_ether_qos overwrites the origin timestamp. For example, DWC_ether_qos operating in Ordinary/Boundary master mode overwrites the origin timestamp field of the sync message, when OSTC is set. Because the sync message is originated by the port operating in Ordinary/Boundary master mode, it is possible for the application generating the message to set the origin timestamp field to zero.

Notes

1. *For PTP over UPD/IPv6 and PTP over UDP/IPv4 with CSC set, the DWC_ether_qos does not verify the presence of the two pad bytes, instead it overwrites the two bytes after the PTP. message*
2. *When Enable One step timestamp for PTP over UDP/IP feature is selected, Enable Transmit TCP/IP Checksum offload feature is automatically enabled and this is enabled only for Queue 0 in multiple transmit queue configurations. If the application requires OST support for PTP messages sent over UDP/IP for packets sent from queues other than 0, transmit checksum engine must be manually enabled for those transmit queues. If checksum engine is not enabled, no updates will be done for PTP messages sent overUDP/IP even though OSTC is set.*
3. *One-Step timestamp operation is not supported for tunneled packets*

44.3.8.7 Using IEEE 1588 Sub Nanoseconds Timestamp Support

The ingress and egress correction can have sub-nanoseconds accuracy.

44.3.8.7.1 Description of IEEE 1588 Sub Nanoseconds Timestamp Support

The sub-nanosecond part of the ingress and egress correction is programmed in the TSICSNS and TSECSNSfields of the MAC Timestamp Ingress Correction Subnanosecond and MAC Timestamp Egress CorrectionSubnanosecond registers, respectively.

In this case, the correction has an unit of nanosecond, multiplied by 2^8. The least significant 8 bits of thevalue should be programmed in the sub-nanoseconds register.

For example, if the required egress correction is 3.6 nS and TSCTRLSSR bit is set, then TSEC must be 0x3and TSECSNS must be 0x99 (0.6 * 256). Similarly, if the required ingress correction is -3.6 nS andTSCTRLSSR bit is set, then TSIC must be 0xBB9A_C9FC and TSICSNS must be 0x66 (fractional part of (10^9- 3.6) * 256).

44.3.9 Multiple Channels and Queues Support

44.3.9.1 Block Diagram of DWC_ether_qos Multiple Channels and Queues

The DWC_ether_qos supports up to eight queues and channels on Tx and Rx paths.

Figure 703 shows the architecture of DWC_ether_qos with multiple queues and channels.

Note: In multi queue/channel configurations, enable only Q0/CH0 on Tx and Rx for half-duplex operation. If you want to enable single queue/channel in full-duplex operation, any queue/channel can be enabled.

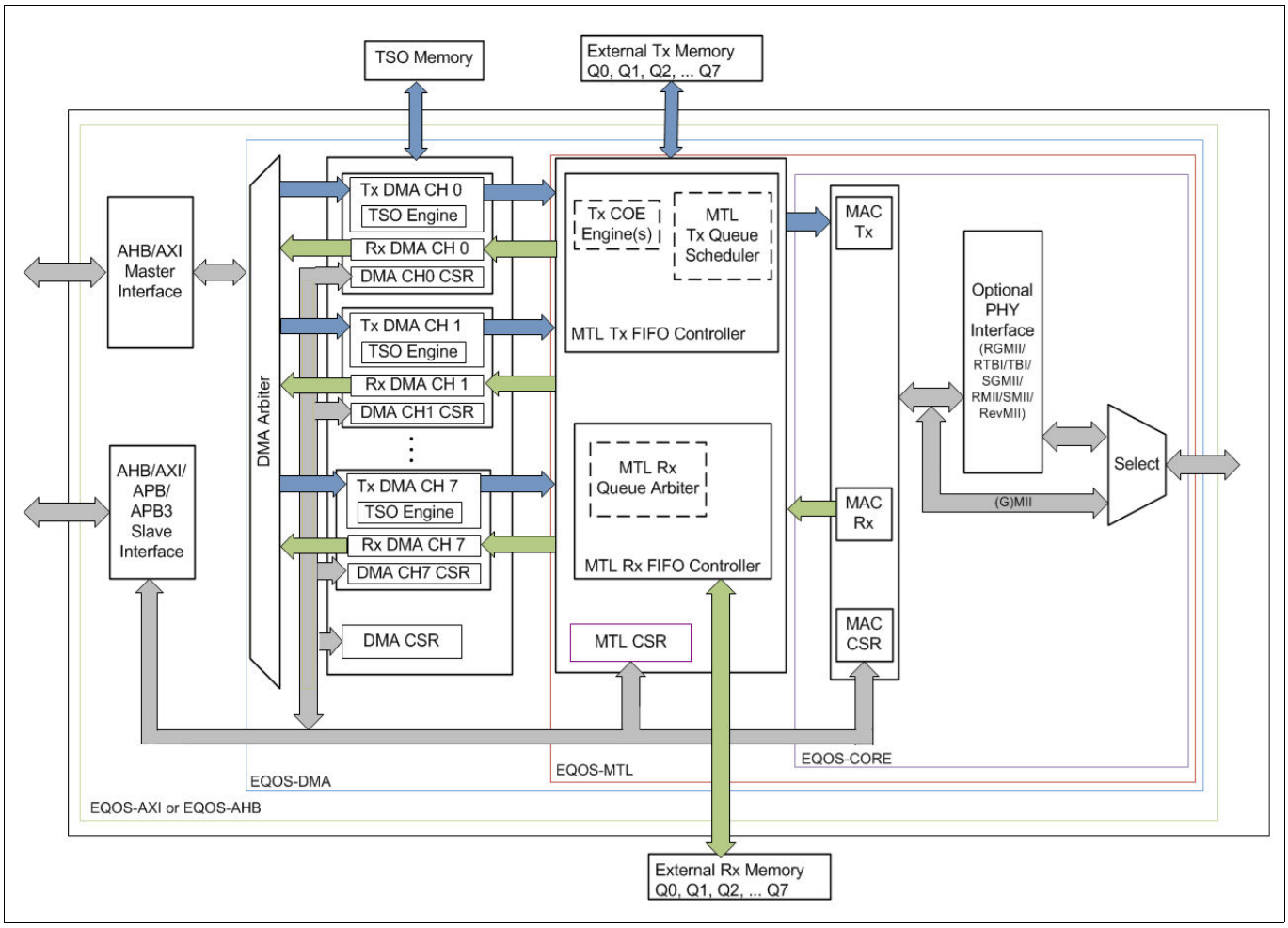


Figure 703 DWC_ether_qos with Multiple Channels and Queues Block Diagram

44.3.9.2 Multiple Queues and Channels Support in Transmit Path

The DWC_ether_qos supports up to eight Transmit queues.

The number of Tx channels is equal to the number of Tx queues. Depending on how the DWC_ether_qos is configured, the controller supports the following methods for handling transfers in the transmit path.

The following topics are discussed:

- Description of Fixed Priority Scheme in EQOS-AHB and EQOS-DMA
- Description of Weighted Strict Priority in EQOS-AHB and EQOS-DMA
- Description of Weighted Round Robin in EQOS-AHB and EQOS-DMA

44.3.9.2.1 Description of Fixed Priority Scheme in EQOS-AHB and EQOS-DMA

For AHB and DMA configurations, the fixed priority scheme is the default scheme.

The fixed priority scheme is the default priority scheme for the DMA channels. In fixed priority scheme, the channel with highest priority always wins the arbitration when it requests the bus. **Table 463** provides information about the priority levels of DMA channels.

Table 463 Fixed Priority Scheme for DMA Channels

Priority Level	Channel
0 (low)	Channel 0
1	Channel 1
2	Channel 2
3	Channel 3
4	Channel 4
5	Channel 5
6	Channel 6
7 (high)	Channel 7

44.3.9.2.2 Description of Weighted Strict Priority in EQOS-AHB and EQOS-DMA

You can use a Weighted Strict Priority (WSP) scheme for EQOS-AHB and EQOS-DMA configurations, where the weight corresponds to the number of burst transactions given to a channel in one arbitration cycle.

In Weighted Strict Priority (WSP), the weight corresponds to the number of burst transfers given to a channel in one arbitration cycle. The unused burst transfers of one or more channels are reallocated based on the priority. The channel with highest priority gets the unused burst transfer time before it is allocated to a channel with next highest priority.

The channel priorities are fixed. Channel 7 has the highest priority and Channel 0 has the lowest priority. When a channel uses the allocated burst transfers, the channel with next lower priority is processed. After processing the allocated bandwidth of all channels that had packets to transmit, any unused burst transfer time is allocated to the channel of the highest priority (if required), and then next highest priority (if required), and so on.

44.3.9.2.3 Description of Weighted Round Robing in EQOS-AHB and EQOS-DMA

EQOS-AHB and EQOS-DMA configurations support a Weighted Round Robin (WRR) priority scheme, in which the weight can be programmed through the DMA_CH#_Tx_Control register.

In Weighted Round Robin (WRR), all channels are serviced in round-robin order according to the weights settings. The TCW field of the DMA_CH#_Tx_Control register provides the weight for each Transmit channel as shown in **Table 464**.

Table 464 Weight for DMA Channels

TCW Field	Transmit Channel Weight
000	1
001	2
010	3
011	4
100	5

Table 464 Weight for DMA Channels (cont'd)

TCW Field	Transmit Channel Weight
101	6
110	7
111	8

The configured weights correspond to the number of burst transfers given to a channel in one arbitration cycle. The unused or excess burst transfers are distributed equally to all channels.

44.3.9.3 Multiple Queues in Receive Path

The DWC_ether_qos supports up to eight Receive channels; the number of Rx channels is independent of the Rx queues.

The DWC_ether_qos supports up to eight Receive channels in EQOS-AHB, EQOS-AXI, and EQOS-DMA configurations. The number of Rx channels is independent of the Rx queues.

In the Rx direction, the MTL Rx Controller selects the Rx DMA for which it is transferring or reading the data from the Rx FIFO memory. This scheduling is based on the programming done in the respective MTL_RxQ[x]_Control register.

Each Rx DMA indicates when it is ready to transfer data and the size of the burst-length (number of beats) that it has to transfer. The scheduler checks whether sufficient data (of requested burst length) is available to be transferred to these DMAs and then selects the Rx DMA that gets serviced using the programmed priorities.

44.3.9.3.1 Priority Scheme for Tx DMA and Rx DMA

The DWC_ether_qos DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin.

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channels for accessing descriptors and data buffers. The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The DA bit of the DMA_Mode register specifies the arbitration scheme (fixed or weighted round-robin) between the Tx and Rx DMA of a channel.

If you have enabled the Tx DMA and Rx DMA of a channel, you can specify which DMA gets the bus when the channel gets the control of the bus. You can set the priority between the corresponding Tx DMA and Rx DMA by using the TXPR field of the DMA_Mode register. For round-robin arbitration, you can use the PR field of the DMA_Mode register to specify the weighted priority between the Tx DMA and Rx DMA. The following table provides information about the priority scheme between Tx DMA and Rx DMA.

Table 465 Priority Scheme for Tx DMA and Rx DMA

Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Priority Scheme
x	x	x	0	1	Rx always has priority over Tx
0	0	0	0	0	Tx and Rx have equal priority. Rx gets the access first on simultaneous requests.
0	0	1	0	0	Rx has priority over Tx in ratio 2:1.
0	1	0	0	0	Rx has priority over Tx in ratio 3:1.
0	1	1	0	0	Rx has priority over Tx in ratio 4:1.
1	0	0	0	0	Rx has priority over Tx in ratio 5:1.
1	0	1	0	0	Rx has priority over Tx in ratio 6:1.
1	1	0	0	0	Rx has priority over Tx in ratio 7:1.

Table 465 Priority Scheme for Tx DMA and Rx DMA (cont'd)

Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Priority Scheme
1	1	1	0	0	Rx has priority over Tx in ratio 8:1.
x	x	x	1	1	Tx always has priority over Rx.
0	0	0	1	0	Tx and Rx have equal priority. Tx gets the access first on simultaneous requests.
0	0	1	1	0	Tx has priority over Rx in ratio 2:1.
0	1	0	1	0	Tx has priority over Rx in ratio 3:1.
0	1	1	1	0	Tx has priority over Rx in ratio 4:1.
1	0	0	1	0	Tx has priority over Rx in ratio 5:1.
1	0	1	1	0	Tx has priority over Rx in ratio 6:1.
1	1	0	1	0	Tx has priority over Rx in ratio 7:1.
1	1	1	1	0	Tx has priority over Rx in ratio 8:1.

Note: Bits 14, 13, 12, 11, and 10 are not valid if one of the paths (Transmit or Receive) is not present in a channel. The Transmit and Receive paths are always present in Channel 0.

44.3.9.4 Multiple Queues and Channels Support in EQOS-MTL

For EQOS-MTL configurations, queues are controlled through the MTL_TxQ#_Operation_Mode and MTL_rxQ#_Operation_Mode registers.

The DWC_ether_qos supports up to eight Tx and Rx queues. This section describes how multiple queues and channels are supported in MTL.

You can control a Tx queue through the MTL_TxQ#_Operation_Mode register. You can control an Rx queue through the MTL_RxQ#_Operation_Mode register.

44.3.9.4.1 Queue Memory

The DWC_ether_qos MTL Tx queues share one Tx memory while Rx queues share one Rx memory; individual queue sizes can be programmed in blocks of 256 bytes.

All MTL Tx queues share one Tx memory and all Rx queues share one Rx memory with programmable individual queue sizes in blocks of 256 bytes. The application can program the Tx queue size in the TQS field of MTL_TxQ#_Operation_Mode register and the Rx queue size in the RQS field of the MTL_RxQ#_Operation_Mode register.

The DWC_ether_qos supports the total Tx memory size of 128 KB (16 KB * 8) and total Rx memory size of 256 KB (32 KB * 8). The total memory size can be selected in powers of 2, that is, 1 KB, 2 KB, 4 KB, 8 KB, 16 KB, 32 KB, 64 KB, 128 KB, and 256 KB. In configurations with a single queue, the maximum Tx memory size is 32 KB and the maximum Rx memory size is 64 KB. In configurations with a single queue, the FIFOs size of 128 bytes, 256 bytes, and 512 bytes is also supported.

44.3.9.5 Rx Queue to DMA Mapping

The DWC_ether_qos controller supports two types of Rx queue to DMA mapping: Static Mapping and Dynamic (Per Packet) Mapping.

The packets in the MTL Rx Queues can be routed to any one of the multiple DMA channels by programming the following registers:

- MTL_RxQ_DMA_Map0 Register (for queues 0, 1, 2 and 3)
- MTL_RxQ_DMA_Map1 Register (for queues 4, 5, 6 and 7)

The following types of Rx Queue to DMA mapping is possible through programming:

- Static Mapping
- Dynamic (Per Packet) Mapping

44.3.9.5.1 Static Mapping

In Static Mapping mode, all packets of an Rx Queue are connected to a specific DMA channel.

In this mode, all of the packets of an Rx Queue are connected to a specific DMA channel. For example, all the packets from Rx Queue 0 can be routed to a DMA channel by programming Q0MDMACH (bit[3:0]) and Q0DDMACH (bit[7] = 0) of the MTL_RxQ_DMA_Map0 Register.

Similarly, packets from other Rx Queues can be routed to any DMA channel by programming register fields corresponding to each Queue.

44.3.9.5.2 Dynamic (Per Packet) Mapping

In Dynamic (per packet) Mapping mode, the destination DMA channel is decide by the MAC core receiver for each packet.

In this mode, the destination DMA channel of a packet being read from a Rx Queue is not constant but decided independently for each packet. For example, if you set the Q1DDMACH bit of the MTL_RxQ_DMA_Map0 register, the static mapping is disabled for Rx Queue 1 and the value in Q1MDMACH is ignored. The destination DMA channel is decided by the MAC Core receiver for each packet, depending on the following in decreasing order of priority:

1. Extended VLAN Based DMA Selection
   Extended routing is applicable only if the VLAN Filter has passed. Routing is only based on Perfect Filter Result. Each perfect filter has a DMA Channel Enable and a DMA Channel Number field which can be programmed. Routing is applicable for a filter, only if the DMA Channel Enable bit is set.The frame is routed to the smallest Matching Filter's DMA Channel provided it is enabled. If that filter's DMA Channel number is not enabled, the frame gets routed to Channel 0. For example, if a frame's VLAN tag matches filters 7, 3, and 1. Then the MAC checks if Filter 1's DMA Channel Number is enabled through programming. If yes, the frame gets routed to the programmed value; otherwise it gets routed to 0. When the inverse filter is enabled; is routed to the least mismatched filter's DMA channel number provided it is enabled.If the DMA Channel enable bit is not set, then the frame is routed based on DA based addressing or to Channel 0.If Hash Filter is also enabled, it is used to determine the Filter result only. Routing will still depend on the Perfect Filters enabled. If none of the perfect filters are enabled or if all of them are bypassed, then the VLAN Filter based routing will not take place. The frame will be routed via DA Based addressing or to Channel 0. If all the perfect filters give a fail result and the Hash Filter has passed, then the VLAN Filter result is a pass but routing will be based on DA Based Addressing or to Channel 0. Similar behavior is seen when inverse Filtering is enabled as well.
2. Ethernet DA-Based DMA Selection
   The DA address of the received packet is compared against the programmed DA values in MAC Address Registers. If the address matches any of the programmed values, the corresponding DCS field (when enabled) determines the destination DMA channel number.

If none of the previous operations is able to make a successful match/decision, then the packet is routed to DMA Channel 0 by default.

44.3.9.6 Selection of Tag Priorities Assigned to Tx and Rx Queues

The DWC_ether_qos controller supports assigning tag priorities to Tx and Rx Queues through programming registers.

The VLAN Tag priorities can be assigned to Tx Queues by programming the corresponding PSTQ# field in the following registers:

- MAC_TxQ_Prty_Map0 (for Tx Queues 0 to 3)
- MAC_TxQ_Prty_Map1 (for Tx Queues 4 to 7)

The bit corresponding to the VLAN Tag priority can be set in the PSTQ# field for assigning that priority to the Tx Queue. For example, if you want to assign VLAN Tag priority of 3 to Tx Queue 0, set bit [3] in PSTQ0 field.

The same VLAN Tag priority can be set for multiple Tx queues. The Tx packet association to particular Tx Queue is governed by the application, so the MAC does not route the packet based on Tx Queue priority mapping; it is only used for Pause Flow Control (PFC). The settings in the PSTQ# field determines the Tx Queues blocked for transmission when the PFC packet is received with corresponding VLAN Tag priorities enabled.

The VLAN Tag priorities can be assigned to Rx Queues by programming the PSRQ field in the corresponding MAC_RxQ_Ctrl2 register. The bit corresponding to the VLAN Tag priority can be set in the PSRQ field for assigning that priority to the Rx Queue. For example, if you want to assign VLAN Tag priority of 3 to Rx Queue 0, set bit [3] in PSRQ field of MAC_RxQ_Ctrl2 register. The VLAN Tag priority assigned to particular Rx Queue must be unique, that is, more than one Rx Queue cannot be assigned the same VLAN Tag priority. However, more than one VLAN Tag priorities can be assigned to same Rx Queue.

44.3.9.7 Rx Side Routing from MAC to Queues

The MAC routes the Rx packets to the Rx Queues based on following packet types in that order:

- Unicast/Multicast destination address packets that fail the destination address filter
- Multicast/Broadcast destination address packets that pass the destination address filter
- VLAN Tag Priority field in VLAN Tagged AV data packets
- AV Control packets
- VLAN Tagged IEEE 1588 PTP over Ethernet packets
- Untagged IEEE1588 PTP over Ethernet packets
- VLAN Tag Priority field in VLAN Tagged packets
- Untagged packets

The outer C-VLAN Tagged AV data Rx packets can be routed based on the priorities assigned to Rx Queues through PSRQ field in corresponding MAC_RxQ_Ctrl2 register and the corresponding Rx Queue is enabled for AV through RXQ#EN field in MAC_RxQ_Ctrl0 register. These packets may be single VLAN Tagged with C-VLAN type or Double VLAN Tagged with outer VLAN Tag of C-VLAN type when Double VLAN feature is enabled (EDVLP bit in MAC_VLAN_Tag register is set) with inner C-VLAN Tagged or inner S-VLAN Tagged when SVLAN processing is enabled (ESVL bit in MAC_VLAN_Tag register is set). This type of Rx packet routing is available when AV feature and Multiple Rx Queues are selected in the configuration.

The AV Control Rx packets can be routed based on the Rx Queue number specified in the AVCPQ field in MAC_RxQ_Ctrl1 register and corresponding Rx Queue is enabled for AV through RXQ#EN field in MAC_RxQ_Ctrl0 register. These packets may be single VLAN Tagged with C-VLAN type or Double VLAN Tagged with outer VLAN Tag of C-VLAN type when Double VLAN feature is enabled (EDVLP bit in MAC_VLAN_Tag register is set) with inner C-VLAN Tagged or inner S-VLAN Tagged when SVLAN processing is enabled (ESVL bit in MAC_VLAN_Tag register is set). This type of Rx packet routing is available when AV feature and Multiple Rx Queues are selected in the configuration.

The VLAN Tagged Rx packets can be routed based on the priorities assigned to Rx Queues through PSRQ field in corresponding MAC_RxQ_Ctrl2 register and the corresponding Rx Queue is enabled for DCB/generic through RXQ#EN field in MAC_RxQ_Ctrl0 register. This type of Rx packet routing is available when Multiple Rx Queues are selected in the configuration.

The untagged IEEE 1588 PTP over Ethernet Rx packets can be routed based on the Rx Queue number specified in the PTPQ field in MAC_RxQ_Ctrl1 register and the corresponding Rx Queue is enabled through RXQ#EN field in MAC_RxQ_Ctrl0 register. The VLAN tagged IEEE 1588 PTP over Ethernet Rx packets can be routed based on the priorities assigned to Rx Queues through PSRQ field in corresponding MAC_RxQ_Ctrl2 register or the Rx Queue number specified in the PTPQ field inMAC_RxQ_Ctrl1 register and the corresponding Rx Queue is enabled through RXQ#EN field in MAC_RxQ_Ctrl0 register. This is determined by programming in TPQC field of MAC_RxQ_Ctrl1 register. This type of Rx packet routing is available when IEEE 1588 Timestamp feature support and Multiple Rx Queues are selected in the configuration. The VLAN Tagged IEEE 1588 PTP over Ethernet Rx packets are detected only when 802.1AS mode is disabled (AV8021ASMEN bit in MAC_Timestamp_Control register is set to 0), otherwise this type of Rx packets are routed as generic VLAN Tagged Rx packets.

The multicast/broadcast destination address Rx packets that pass the destination address filter can be routed based on the Rx Queue number specified in the MCBCQ field of MAC_RxQ_Ctrl1 register when enabled through MCBCQEN bit of MAC_RxQ_Ctrl1 register and the corresponding Rx Queue is enabled through RXQ#EN field in MAC_RxQ_Ctrl0 register. This type of Rx packet routing is available when Multiple Rx Queues are selected in the configuration.

The untagged Rx packets can be routed based on the Rx Queue number specified in the UPQ field of MAC_RxQ_Ctrl1 register and the corresponding Rx Queue is enabled through RXQ#EN field in MAC_RxQ_Ctrl0 register. This type of Rx packet routing is available when Multiple Rx Queues are selected in the configuration.

The unicast destination address Rx packets that fail the destination address filter can be routed based on the Rx Queue number specified in the UFFQ field of MAC_RxQ_Routing_Ctrl register when enabled through UFFQE bit of MAC_RxQ_Routing_Ctrl register, RA bit of MAC_Packet_Filter register is set to 1 and the corresponding Rx Queue is enabled through RXQ#EN field in MAC_RxQ_Ctrl0 register. This type of Rx packet routing is available when Multiple Rx Queues are selected in the configuration.

The multicast destination address Rx packets that fail the destination address filter can be routed based on the Rx Queue number specified in the MFFQ field of MAC_RxQ_Routing_Ctrl register when enabled through MFFQE bit of MAC_RxQ_Routing_Ctrl register, RA bit of MAC_Packet_Filter register is set to 1 and the corresponding Rx Queue is enabled through RXQ#EN field in MAC_RxQ_Ctrl0 register. This type of Rx packet routing is available when Multiple Rx Queues are selected in the configuration.

If the Rx packet cannot be classified in any of the defined packet type for routing, it will be routed through Rx Queue 0.

44.3.9.8 Rx Side Arbitration Between DMA and MTL

The DWC_ether_qos controller supports Rx side arbitration between the DMA and the MTL.

After completing the current packet processing, the DMA Channel Controller fetches the next descriptor. After the descriptor fetching is complete, the DMA Channel Controller evaluates the amount of data to be to be transferred to the Rx Buffer based on the programmed PBL and Rx Buffer Length. Accordingly, it requests the MTL to transfer the data.

After servicing the current request, the MTL Rx Queue arbitration scheme selects Rx Queue based on the arbitration scheme (RAA field of the MTL_Operation_Mode register) and the weights programmed in the Queue <n> Receive Control Register. The arbitration is done among Queues for which DMA is ready to service. After the Rx Queue is selected, PBL (Programmable Burst Length) amount of data is read out from that Queue and is routed to the Rx DMA Channel based on the Rx Channel selection criteria.

The arbitration takes place when every PBL data transfer is completed and descriptors are ready for the processing from at least one DMA Channel.

44.3.9.9 Tx Side Arbitration between DMA and MTL

The DWC_ether_qos support Tx side arbitration between the DMA and the MTL.

Because the number of Tx DMA channels is always equal to the number of Tx Queues in the MTL, they are always mapped directly. For instance, each Tx DMA pushes data into its respective Tx Queue assigned to it. The main reason for this strategy is to optimize and increase the efficiency of the DMA data transfer, as well as to simplify the TX Checksum Engines in the MTL.

The data inside each Tx Queue is stored in packets. Hence, if two DMAs are allowed to transfer data into the same queue, when a Tx DMA starts a packet transfer, the other DMA cannot transfer data unless the previous packet is completely pushed-in. This means that the second DMA remains idle until the first packet is transferred. Hence, each DMA is always connected directly to its corresponding Tx Queue.

44.3.9.10 Audio Video Bridging

44.3.9.10.1 Introduction to AV Feature

The DWC_ether_qos supports the AV data transfer in 100 Mbps and 1000 Mbps modes. The AV feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

- IEEE 802.1Qav-2009: Allows the bridges to provide time-sensitive and loss-sensitive real-time audio video data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources.
- IEEE 802.1Qat-2009: Allows the network resources to be reserved for specific traffic streams traversing a bridged local area network.
- IEEE 802.1AS-2011: Specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration.

Note: In EQOS-CORE configurations, the AV feature is limited to identifying the AV Type payloads in the Ethernet packet and indicating it on the MAC Receive Interface (MRI) through mri_q_sel_o signal along with the SOP. For more information, see "Receive Path Functions".

44.3.9.10.2 Transmit Path Functions

When the DWC_ether_qos controller supports the AV Bridging feature, you can enable and disable certain functions associated with the transmit path.

When you select the AV feature, the Transmit paths of Queue 0 and the highest enabled queue are enabled by default.

The Transmit path of Queue 0 supports the strict priority algorithm, and it is used for best-effort traffic. For a queue, the strict priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for MTL Tx FIFO is enabled, the strict priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of size equal to the programmed threshold limit.

The Transmit paths of additional queues support traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

- The queue contains one or more packets.
- The credit for the queue is positive as per the algorithm.

You can disable the credit-based shaper algorithm for all queues or lower-priority queues. For example, you can either disable the credit-based shaper algorithm for Queue 1 and Queue 2 or only for Queue 1. You should not disable the credit-based shaper algorithm for Queue 2 and enable it for Queue 1. When you disable the credit-based shaper algorithm for a queue, the channel uses the default strict priority algorithm.

Each Transmit DMA has a separate descriptor chain for fetching the transmit data. The Transmit channel that gets the access to the system bus depends on the DMA arbiter.

The Transmit path has a shared FIFO (MTL layer) for each queue. The data fetched by the DMA is put in the respective part of the FIFO. The MTL Tx Queue Scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for a queue, the corresponding queue is selected for transmission if the following conditions are true:

- If the packet is available in the channel and has a positive or zero credit.
- If the higher priority queue has no packet waiting in the FIFO.

If the credit-based shaper algorithm is disabled for all queues, the packet to be transmitted from a queue is selected based on the priority scheme described in the following table.

Table 466 Weight for DMA Channels

TCW Field	Transmit Channel Weight
000	1
001	2
010	3
011	4
100	5
101	6
110	7
111	8

44.3.9.10.3 Receive Path Functions

When the DWC_ether_qos controller supports the AV Bridging feature, you can enable and disable certain functions associated with the receive path.

When you select the AV feature, the Receive path of Queue 0 is enabled by default. All traffic is received on this channel. You can enable the Receive paths of additional queues. By enabling the Receive paths of multiple channels, you can demultiplex the received data to send the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if it is an AV packet and its corresponding VLAN Priority tag value. This status is updated in the Extended Status field of the Receive descriptor as explained in "Receive Descriptor". All received packets with the EtherType field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

- AV data packets: The AV data packets are always tagged. The tagged AV data packets are received based on the programmed priority value. To specify the channel to which an AV packet with a given priority must be sent, program Bits[15:8] in the Transmit Flow Control Register of the corresponding queue.
- AV control packets: The AV control packets can be either tagged or untagged. The untagged AV control packets are received on Queue 0 by default. To receive these packets on any other queue, you can program Bits[2:0] of the MAC_RxQ_Ctrl1 register. Similar to the AV data packets, the tagged AV control packets are received based on the programmed priority value.

In addition to the AV packets, you can receive the untagged PTP packets on any queue. By default, the PTP packets (tagged or untagged) are received on Queue 0. To receive these packets on any other queue, you need to program Bits[6:4] of the MAC_RxQ_Ctrl1 register.

44.3.9.10.4 Credit-Based Sharper Algorithm

When the DWC_ether_qos supports the AV feature, the MTL Queue Scheduler uses the credit-based sharper algorithm. You can program additional queues to use this algorithm.

The MTL Queue Scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in Queue 0. You can program the additional queues to use the credit-based shaper algorithm.

The following sections provide information about how you can implement the credit-based shaper algorithm:

- Credit Value
- idleSlopeCredit and sendSlopeCredit Values
- Bandwidth Status

Credit Value

The credit value is part of the credit-based sharper algorithm used by the MTL Queue Scheduler.

The credit value is accumulated every transmit clock cycle, that is, 40 ns for 100 Mbps and 8 ns for 1000 Mbps. The credit to be added or subtracted per cycle can be fractional based on the required idleSlope and sendSlope values, as described in **Table 467**.

Table 467 Credit Value Per Transmit Cycle Example

Mode	Values	Description
100 Mbps	• portTransmitRate = 100 Mbps • idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class) • sendSlope = 30 Mbps	• credit = 2.8 bits accumulates per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted. • credit = 1.2 bits drains per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO based on the slot number in the transmit descriptor (if enabled) or the bandwidth availability on the AMBA AHB application bus.

The credit for a queue builds up only when the packet is available but it cannot be transmitted because the MAC is sending a packet from another queue. The DWC_ether_qos supports another mode in which the credit can build up in advance for a queue in which no packet is available in respective part of the FIFO. This enables sending a burst of high priority traffic in a queue as soon as the data is available. You can enable this mode with Bit 1 of the CBS Control registers of corresponding queues.

When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no packet waiting in a queue and other queues are transmitting. When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

idleSlopeCredit and sendSlopeCredit Values

The idleSlopeCredit and sendSlopeCredit values are programmable.

The software must program the idleSlopeCredit and sendSlopeCredit values. The programmed values should be the credit accumulated or drained per clock cycle scaled by 1024, such as, 2.8 x 1024 = 2867 and 1.2 x 1024 = 1229. In addition, the software must program the hiCredit and loCredit values, scaled by 1024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12,000 bits and 3,036 bits respectively, the values to be programmed in the hiCredit and loCredit registers of the corresponding channel are 12000 x 1024 bits and two's complement of 3036 x 1024, respectively.

Bandwidth Status

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the CBS status registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The CBS status register gives the average number of bits transmitted during the previous programmed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programmed in Bits[6:4] of the CBS control register of the respective queue. For example, if you have programmed two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programmed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue.

A lower priority queue, which is using the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue, which is using the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, Queue 1 and Queue 2 use the credit-based shaper algorithm (with reserved bandwidth of 50% and 25%, respectively) and Queue 0 uses the strict-priority algorithm. If Queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by Queue 0. Queue 2 cannot exceed the reserved bandwidth of 25%.

44.3.9.10.5 Slot Number Function with Audio Video Bridging

Introduction to Slot Number Function

When the DWC_ether_qos controller supports the AV feature, you can program the slot number where the DMA fetches data from system memory in the Transmit Descriptor 3 (TDES3).

When the AV feature is enabled, you can use the slot number function to schedule the data fetching from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals.

You can program the slot number at which the DMA should fetch the data from system memory in the Transmit Descriptor Word 3 (TDES3). This 4-bit field allows the application to schedule data up to 16 slots of 125 µs each. This field is applicable only for the AV channels.

Description of Slot Number Function

You can enable the DMA to fetch data only if it matches the current slot or next two slots.

When the DMA fetches a Tx descriptor, it compares the slot number of the Tx descriptor with the internally generated reference slot number. The reference slot number is updated every time the interval programmed in SIV field of DMA_CH[n]_Slot_Function_Control_Status register in terms of microseconds of the IEEE 1588 system time elapses. In addition, the slot interval counter is initialized to zero when the seconds field of the system time is incremented, that is, the sub-second counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state until there is a match.

To enable the DMA to fetch the data only if it matches the current slot or the next two slots, you can program Bit 1 of the Slot Function Control and Status register of the corresponding DMA channel.

Note: If the slot number in the descriptor is less than the reference slot number, the DMA takes it as a future slot.

You can enable the check for slot number by setting Bit 0 of the Slot Function Control and Status register of corresponding DMA channel. When this check is not enabled, the packets are fetched immediately after the descriptor is read. In addition, Bits [19:16] indicate the value of the reference slot number in DMA.

44.3.9.11 Queue Modes

You can specify queue modes depending on the type of feature supported.

You can enable a Tx queue for generic/DCB, AV, or all types of traffic. When you select multiple Tx queues without selecting the DCB feature, the queues are enabled for generic queuing using WRR or WSP algorithms.

To enable a Tx queue for AV, you need to select the Enable Audio Video Bridging option and then the Enable Support for AV in Tx Queue # option. The queuing is based on the CBS or SP algorithms.

You can enable an Rx queue for generic, DCB, AV, or all types of traffic. A particular queue enabled for generic/DCB or AV based routing is determined by the RXQ#EN field of corresponding queue. The following list explains how Rx queues are enabled based on the selected features:

- Multiple Rx queues without the DCB feature
  When you select multiple Rx queues without selecting DCB feature, all queues are enabled for generic queuing based on the VLAN Tag priority. The VLAN Tag priority should match the PSRQ field of the MAC_RxQ_Ctrl2 register. By default untagged packets are routed to Receive queue specified in UPQ field of MAC_RxQ_Ctrl1 register. Queue 0 is the default value of UPQ field. You can override the default value with any other value, for the UPQ field. The Rx packets can also be routed to a particular DMA channel based on the DCS field of perfectly-matched MAC Address register.
- Multiple Rx queues with AV feature
  When you select the Enable Audio Video Bridging option, AV is enabled for all selected Rx queues. The queuing is done based on the VLAN Tag priority for DCB data packets. The VLAN Tag priority should match the PSRQ field of the MAC_RxQ_Ctrl2 register. The Rx packets can also be routed to a particular DMA channel based on the DCS field of perfectly-matched MAC Address register. The AV control packets (tagged or untagged) are routed based on the AVCPQ field of the MAC_RxQ_Ctrl1 register. The PTP over Ethernet packets are routed based on the AVPTPQ field of MAC_RxQ_Ctrl1 register.

44.3.9.12 Programming Guidelines for Disabling Flow Control in AV Queues

The DWC_ether_qos supports programming of same priorities for AV and DCB enabled Rx queues, when both coexist. For example, priority 1 can be programmed in PSRQ[n] field corresponding to Rx Queue enabled for AV in MAC_RxQ_Ctrl2 register and same priority 1 can be programmed in PSRQ[n] field corresponding to Rx Queue enabled for DCB in MAC_RxQ_Ctrl2 register, at the same time. When AV and DCB operation coexists, it requires that the Flow Control is disabled for AV enabled queues. See, “Disabling Flow Control for AV Enabled Queues”.

44.3.9.13 Queue Priorities

The Tx queue priorities can be programmed through the MAC_TxQ_Prty_Map0 and MAC_TxQ_Prty_Map1 registers. The Rx queue priorities are programmed through the Rx Flow Control register.

You can program the priority of a Tx queue in the MAC_TxQ_Prty_Map0 and MAC_TxQ_Prty_Map1 registers. You can program the priority of an Rx queue corresponding field of MAC_RxQ_Ctrl2 register. The priority should be assigned in the following order:

1. AV queue (high priority)
2. Best-effort queue (low priority)

The software should put packets with correct priorities in the respective programmed queue on Tx side. The MAC uses the programmed priorities for blocking the Tx queues when a PFC packet is received. If a single queue is selected for multiple priorities and PFC is enabled, the entire queue is paused if one or more priorities in the queue are paused.

44.3.10 Using TCP/IP Offloading Features

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. Therefore, the MAC has an optional Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, and error detection in the Receive path.

By using the Checksum Offload features, the software can offload the task of checksum insertion and checking to the hardware. This feature helps in improving the throughput.

In the transmit path MAC calculates the checksum and inserts it in the Tx packet. In the receive path, MAC verifies the received checksum with the internally calculated checksum and provides the status.

It is possible to split the received packet such that the header and the payload are stored in different buffers in system memory.

44.3.10.1 Using Transmit Checksum Offload Engine

44.3.10.1.1 Introduction to the Transmit Checksum Offload Engine

The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, using which, the software can offload the task of checksum insertion to the hardware. In the transmit path MAC calculates the checksum and inserts it in the Tx packet. This feature helps in improving the throughput.

44.3.10.2 Description of the Transmit Checksum Offload Engine

The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, when this function is enabled, the Tx FIFO automatically operates in the store-and-forward mode even if the DWC_ether_qos is configured for Threshold (cut-through) mode.

You must make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter. The reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid dead-lock. When reading starts, the COE fails and consequently all succeeding packets may get corrupted because of improper recovery. Therefore, you must enable the checksum insertion only in the packets that are less than the number of bytes, given by the following equation:

```
Packet size < TxQSize - ((PBL + 7) * 4)
```

Where:

- TxQSize is indicated by TQS field of MTL_TxQ#_Operation_Mode register
- PBL = TxPBL field in the DMA_CH#_TX_Control register in all DMA configurations

See IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

44.3.10.2.1 IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the Transmit status (Bit 0 of TDES3 Normal Descriptor (Write-Back Format). This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 Header Length field.
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4.
- For IPv46datagrams:
  - The Ethernet type is 0x86dd but the IP header Version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

44.3.10.2.2 TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 16'h0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 16’h0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector. This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC Transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and

no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 468 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Table 468 Transmit Checksum Offload Engine Functions for Different Packet Types

Packet type	Hardware IP header checksum insertiion	Hardware TCP/UDP/ICMP checksum insertion
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Not Applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: • Hop-by-hop options (in IPv6 main header) • Hop-by-hop options (in IPv6 extension header) • Destinations options • Routing (with segment left 0) • Routing (with segment left > 0) • TCP, UDP, or ICMP • Authentication • Any other next header field in main or extension headers	 • Not Applicable • Not Applicable • Not Applicable • Not Applicable • Not Applicable • Not Applicable • Not Applicable • Not Applicable	 • Yes • No • Yes • No • No • Yes • No • No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: • IPv4 packet in an IPv4 tunnel • IPv6 packet in an IPv4 tunnel	 • Yes (IPv4 tunnel header) • Yes (IPv4 tunnel header)	 • No • No
IPv6 Tunnels: • IPv4 packet in an IPv6 tunnel • IPv6 packet in an IPv6 tunnel	 • Not Applicable • Not Applicable	 • No • No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled)	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled)	Not Applicable	Yes

Table 468 Transmit Checksum Offload Engine Functions for Different Packet Types (cont'd)

Packet type	Hardware IP header checksum insertiion	Hardware TCP/UDP/ICMP checksum insertion
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not Applicable	No

44.3.10.3 Using Receive Checksum Offload Engine

44.3.10.3.1 Introduction to the Receive Checksum Offload Engine

DWC_ether_qos provides the Checksum Offload Engine that is used to detect any error in an IPv4 or IPv6 packet in the receive path.

The MAC has a Checksum Offload Engine (COE) to support packet error detection in the Receive path. In the receive path, MAC verifies the received checksum with the internally calculated checksum and provides the status.

44.3.10.3.2 Description of the Receive Checksum Offload Engine

The Receive Checksum Offload engine is used to detect errors in IP packets by calculating the header checksum and further matching it with the received header checksum. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

Here, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the Enable Double VLAN Processing option is selected and the EDVLP bit of the MAC_VLAN_Tag register is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 469 describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status, as described in Table 3-13 on page 174.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 469 Receive Checksum Offload Engine Functions for Different Packet Types

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Not Applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: • Hop-by-hop options (in IPv6 main header) • Hop-by-hop options (in IPv6 extension header) • Destinations options • Routing (with segment left 0) • Routing (with segment left > 0) • TCP, UDP, or ICMP • Any other next header field in main or extension headers	• Not Applicable • Not Applicable • Not Applicable • Not Applicable • Not Applicable • Not Applicable • Not Applicable	• Yes • No • Yes • Yes • No • Yes • No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: • IPv4 packet in an IPv4 tunnel • IPv6 packet in an IPv4 tunnel	• Yes (IPv4 tunnel header) • Yes (IPv4 tunnel header)	• No • No
IPv6 Tunnels: • IPv4 packet in an IPv6 tunnel • IPv6 packet in an IPv6 tunnel	• Not Applicable • Not Applicable	• No • No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled)	Yes	Yes

Table 469 Receive Checksum Offload Engine Functions for Different Packet Types (cont'd)

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled)	Not Applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not Applicable	No

44.3.11 Splitting Header on Receive

Optionally, the incoming receive packet header can be split such that the header and the payload are stored in different buffers in the system memory. Header splitting is also useful for non-IP packets such as AVB traffic which are generally multicast type. This enables the application to store the packet header of AVB streams and payload for AVTP (Audio Video Transport Protocol) packets and the DMA can process the header and payload of the received packets separately. The table below shows how DMA processes a packet depending upon its type and the value of the SPLM bit.

Table 470 Receive Checksum Offload Engine Functions for Different Packet Types

Packet type	SPLM	Description
TCP or UDP Packet	00 (L3/L4 Split)	The DMA writes the Ethernet header + IP header + TCP or UDP header into the header buffer.
IP packet (notTCP/UDP)		The DMA writes the Ethernet header + IP header into the header buffer.
Non-IP packet		The DMA does not split the header and payload
IP packet and non IP packet	01 (L2 Split)	The DMA writes the Ethernet header based on split offset (SPLOFST) into the header buffer.
IP packet	10 (Combination of L2 or L3/L4Split)	L3/L4 split
Non-IP packet		L2 split
NA	11	Reserved

Note: L3/L4 split is applicable for IP packets that are either untagged or VLAN stripped.

The IP header includes IPv4 options in case of a IPv4 packet, and IPv6 extension headers in case of IPv6 frames.

44.3.12 Implementing Low-Power Modes

44.3.12.1 Implementing Energy Efficient Ethernet

44.3.12.1.1 Introduction to Energy Efficient Ethernet (EEE)

EEE is an operational mode that allows a controller to operate in low-power mode.

This topic provides an overview of power management through EEE.

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps, 1000 Mbps, and 10 Gbps. The DWC_ether_qos supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows power saving by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

44.3.12.1.2 Description of EEE

Transmit Path Functions

The transmit path functions include tasks that the MAC must perform to make the PHY to enter the LPI state.To enter this state, the software must set the MAC_LPI_Control_Status.LPIEN bit to initiate the LPI protocol.

In the transmit path, the software must set the LPIEN bit of the MAC_LPI_Control_Status register to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER field of MAC_LPI_Control_Status register. The PHY Link Status bit of the LPI Control and Status Register indicates the link status of the PHY.

Note: According to the Energy Efficient Ethernet standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode. However, the MAC can stop the GTX_CLK clock during the LPI state in the GMII (1000) code.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN
2. Asserts TX_ER
3. Sets TXD[3:0] to 0x1 (for 100 Mbps) or TXD[7:0] to 0x01 (for 1,000 Mbps)

   Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

4. Updates the status (TLPIEN bit of MAC_LPI_Control_Status register) and generates an interrupt

To bring the PHY out of the LPI state, that is, when the software resets the LPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern
2. Starts the LPI TW TIMERThe MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the MAC_LPI_Timers_Control register.
3. Updates the LPI exit status (TLPIEX bit of the MAC_LPI_Control_Status register) and generates an interrupt

Figure 704 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

Notes

1. *The MAC does not stop the TX_CLK clock. You can stop this clock (as shown in **Figure 704**) if your PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o signal to 1. The sbd_tx_clk_gating_ctrl_o signal is asserted after nine Tx Clock Cycles, one Pulse Synchronizer delay (BCM22), and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal is dependent on the LPITCSE bit of the MAC_LPI_Control_Status register.*
2. *If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern and so the Tx Clock cannot be gated.*
3. *In the EQOS-CORE configuration, when the Tx clock gating is done during the LPI mode, you cannot use the LPITXA bit of the MAC_LPI_Control_Status register.*
4. *If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.*
5. *If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.*

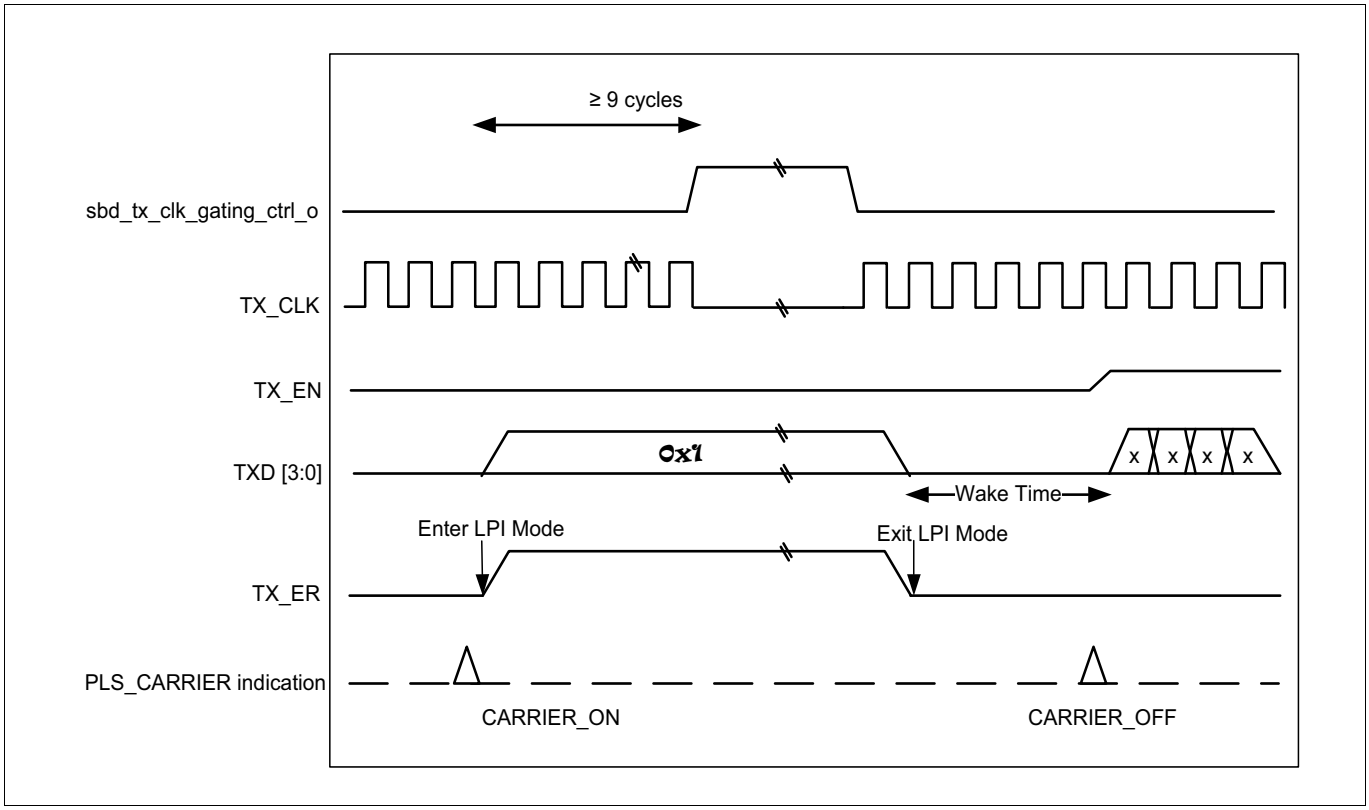


Figure 704 LPI Transitions (Transmit)

Automated Entry/Exit of LPI mode in TX Path

The MAC transmitter can be programmed to enter and exit LPI IDLE mode automatically based on whether it is IDLE for a specific period of time or has a packet to transfer. These modes are enabled and controlled by MAC_LPI_Control_Status register.

When LPITXA (Bit[19]) and LPITXEN (Bit[16]) of MAC_LPI_Control_Status register are set, the MAC transmitter enters LPI IDLE state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter will exit the LPI IDLE state and clear the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition to the above, when Bit[20] (LPIATE) is also set, the MAC transmitter will enter LPI IDLE state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in MAC_LPI_Entry_Timer. In this mode also, the MAC transmitter will exit the LPI IDLE state as soon as any of the functions becomes non-idle. However, the LPITXEN bit is not cleared but remains active so that reentry to LPI IDLE state is possible without any software intervention when the MAC becomes idle again.

When both LPIATE and LPITXA bits are cleared, you can directly control the entry and exit of LPI IDLE state by programming the LPITXEN bit.

Receive Path Functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER
2. The PHY sets RXD[7:0] to 0x01
3. The PHY de-asserts RX_DV

   Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

4. The MAC updates the RLPIEN bit of the MAC_LPI_Control_Status register and immediately generates an interrupt

Notes

1. *If the LPI pattern is detected for a very short duration (that is, less than two cycles of Rx clock), the MAC does not enter the Rx LPI mode.*
2. *If the duration between end of the current Rx LPI pattern and start of the next Rx LPI pattern, is very short (that is, less than two cycles of Rx clock), then the MAC exits and again enters the Rx LPI mode. The MAC does not give the Rx LPI Exit and Entry interrupts.*

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the MAC_LPI_Control_Status register and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 705 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

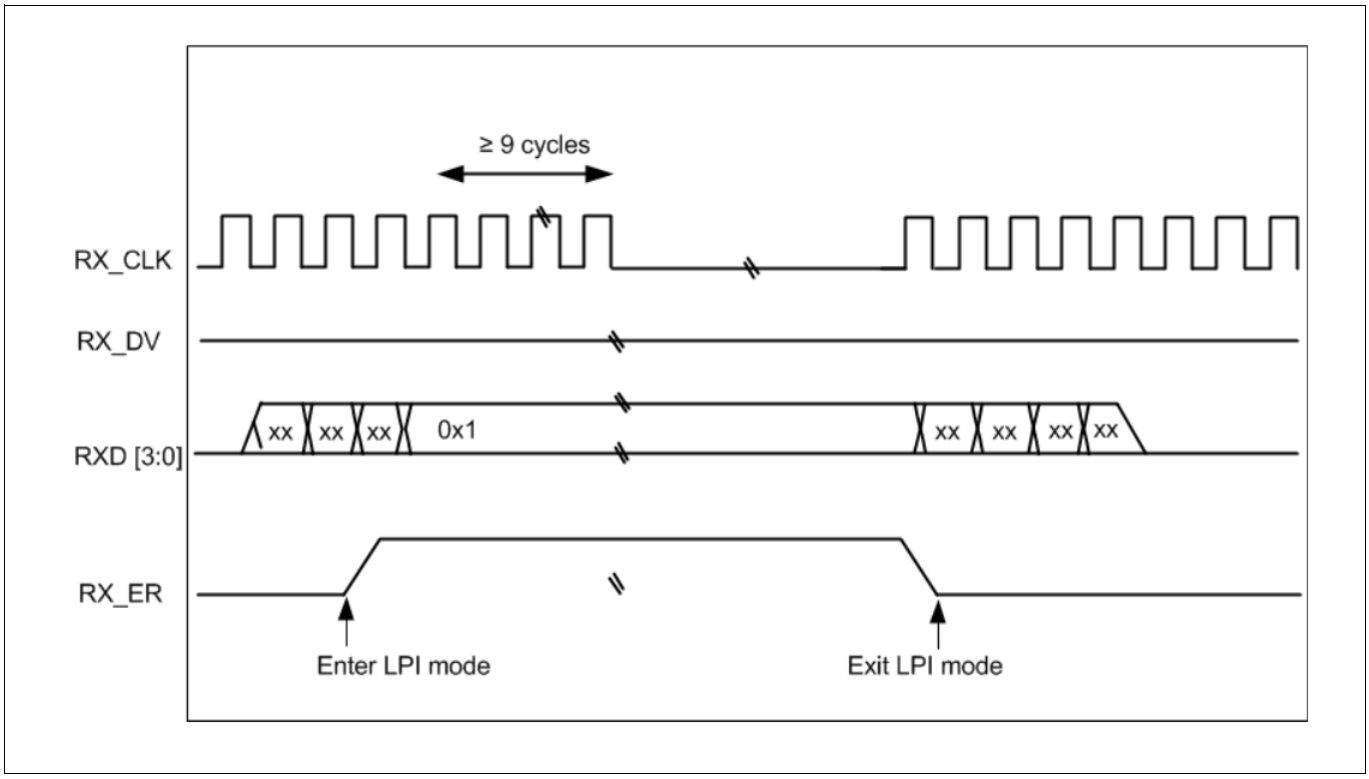


Figure 705 LPI Transitions (Receive)

Notes

1. *If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in **Figure 705**.*
2. *If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC receiver comes out of the LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver again enters the LPI state.*
3. *If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.*
4. *When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.*

44.3.12.1.3 LPI Timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers.

Following are the LPI timers that are loaded with the respective values from the MAC_LPI_Timers_Control and MAC_LPI_Entry_Timer registers:

- LPI LS TIMER
  The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. You can enable the monitoring of the LUD bit of the MAC_PHYIF_Control_Status register by setting the PLSEN bit of MAC_LPI_Control_Status register.
  The link status is indicated to the MAC by the LNKSTS bit of the MAC_PHYIF_Control_Status register or the value programmed by the software in the PLS bit of the MAC_LPI_Control_Status register. If the link status is not available in the MAC_PHYIF_Control_Status, the software should get the PHY link status by reading the PHY register and accordingly update the PLS bit.
  This timer is cleared every time the link goes down. It starts to increment when the link is up again and

continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in Bits[25:16] of the MAC_LPI_Timers_Control register. The GMII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. Therefore, the software can program up to 1023 milliseconds.

- LPI TW TIMER
  The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in Bit[15:0] of MAC Register 53 (LPI Timers Control Register). The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count.
  The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 µ s.
- LPI AUTO ENTRY TIMER
  This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC Transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in MAC_LPI_Control_Status register is set.

Note: Program the PLS bit of MAC_LPI_Control_Status to 1'b0 before switching between the GMII and MII modes. This resets the internal timers. If the mode is changed after the LPI LS TIMER or LPI TW TIMER starts, the change in the Tx clock frequency can result in incorrect timeout.

44.3.12.1.4 LPI Interrupt

This topic describes the low-power idle (LPI) mode interrupt signal that MAC generates when it enters or exits the LPI state.

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt mci_intr_o (sbd_intr_o in EQOS-DMA configurations) is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the MAC_LPI_Control_Status register.

When the MAC exits the Rx LPI state, then in addition to the mci_intr_o (sbd_intr_o in EQOS-DMA configurations), the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. You can use the lpi_intr_o signal to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that you can stop the application clock when the MAC is in the LPI state. If you do not want to gate-off the application clock during the Rx LPI state, you can leave the lpi_intr_o signal unconnected and use the mci_intr_o (sbd_intr_o in EQOSDMA configurations) signal to detect Rx LPI exit.

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately afterthe MAC_LPI_Control_Status register is read. This is because the clear signal, generated in CSRclock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is atleast four clock cycles of Rx clock and can be significant when the DWC_ether_qos is operating in the 10Mbps mode.

44.3.12.2 Implementing Power Management Through Magic Packet Detection

44.3.12.2.1 Power Management (PMT) Through Magic Packet Detection

You can configure a controller to function in low-power mode and power it on only when it receives a magic packet addressed to it from the network.

This section provides an overview of power management through magic packet detection.

The PMT block supports the reception of magic packets as wake up packets. Power can be saved by gating off the CSR clock and the Tx Clock during power-down mode. The PMT block does not perform the clock gate function, but generates interrupts for magic packets that the MAC receives.

Note: The magic packet feature is implemented based on the Magic Packet Technology white paper.

The magic packet is based on a method that uses the magic packet technology from Advanced Micro Device to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network.

The MAC checks only those magic packets that are addressed to the MAC or a multicast address (including broadcast address) to determine whether these packets meet the wake-up requirements. The magic packets that pass the address filtering (unicast or multicast (including broadcast) address) are checked to determine whether they meet the remote wake-up packet data format of six bytes of all ones followed by a Unicast MAC Address (that matches the value in MAC Address 0) appearing 16 times.

The application enables the magic packet wake-up by writing 1 to the MGKPKTEN bit of the MAC_PMT_Control_Status register. The PMT block constantly monitors each packet addressed to the node for a specific magic packet pattern. Each packet received is checked for a 48’hFF_FF_FF_FF_FF_FF pattern following the destination address, source address, and Length/Type fields. The PMT block then checks the packet for 16 repetitions of the MAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the PMT block again scans the 48’hFF_FF_FF_FF_FF_FF pattern in the incoming packet. The 16 repetitions can be anywhere in the packet, but must be preceded by the synchronization stream (48’hFF_FF_FF_FF_FF_FF). The device can also accept a multicast packet, as long as the 16 duplications of the MAC address are detected. If the number of repetitions of 8'hFF are more than six, the PMT block checks for 16 repetitions of the MAC address without any breaks or interruptions, after the last 6 repetitions of 8'hFF.

If the MAC address of a node is 48'h00_11_22_33_44_55, the MAC scans for the following data sequence:

```
Destination Address Source Address Length/Type..... FF FF FF FF FF FF00 11 22
33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 5500 11 22 33 44
55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 5500 11 22 33 44 55 00
11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 5500 11 22 33 44 55 00 11 22
33 44 55 00 11 22 33 44 55 00 11 22 33 44 55…CRC
```

DWC_ether_qos checks the remote wake-up packet only for length error, FCS error, dribble bit error, GMII error, collision. In addition, the remote wake-up packet is checked to ensure that it is not a runt packet. Even if the remote wake-up packet is more than 512 bytes long, if the packet has a valid CRC value, DWC_ether_qos considers it a valid packet.

The magic packet detection is updated in the MAC_PMT_Control_Status register for the received magic packet. A PMT interrupt to the Application triggers a read to the MAC_PMT_Control_Status register to determine whether a magic packet has been received.

Notes

1. *The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit of MAC_Configuration register and PWE bit in MAC_Watchdog_Timeout register.*
2. *The value programmed in DCRCC bit of MAC_Ext_Configuration register is applicable to a magic packet only when Enable IPv4 ARP Offload is selected in the configuration.*

PMT Block Registers

Use the MGKPKTEN bits of the MAC_PMT_Control_Status register to generate power management events. The application should program these bits. You can access the PMT registers in the similar manner as you access the MAC CSR registers.

44.3.12.3 Implementing Power Managemetn Through Remote Wake-Up Packet Detection

44.3.12.3.1 Power Management (PMT) Throught Remote Wake-Up Packet Detection

You can configure a controller to function in low-power mode and power it on only when it receives a remote wake-up packet addressed to it from the network.

This topic provides an overview of power management through remote wake-up packet detection.

The power management block supports the reception of network (remote) wake-up packets as wake up packets. Power can be saved by gating off the CSR clock and the Tx Clock during power down mode. The PMT block does not perform the clock gate function, but generates interrupts for remote wake-up packets that the MAC receives.

More number of Remote Wake-up Filters facilitate a wider range of packets being detected as remote wake-up packets.

Note: The remote wake-up packet feature is implemented based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

When the MAC is in sleep mode and the remote wake-up bit is enabled in the MAC_PMT_Control_Status, the normal operation is resumed after a remote wake-up packet is received. The application writes all eight wake-up filter registers, by performing a sequential Write to address (00C4H). The application enables remote wake-up by writing 1 to the RWKPKTEN bit of the MAC_PMT_Control_Status register.

The PMT block supports 16 programmable filters that allow support of different receive packet patterns. If the incoming packet passes the address filtering of Filter Command, and if Filter CRC-16 matches the CRC of the incoming pattern, the MAC identifies the packet as a wake-up packet.

The Filter Offset (see Filter i Offset) determines the offset from which the packet is to be examined. The Filter Byte Mask (see Filter i Byte Mask) determines which bytes of the packet must be examined. The 31st bit of Byte Mask must be set to zero.

The remote wake-up CRC block determines the CRC value that is compared with Filter CRC-16. The remote wake-up packet is checked only for length error, FCS error, dribble bit error, GMII error, collision. In addition, the remote wake-up packet is checked to ensure that it is not a runt packet. Even if the remote wake-up packet is more than 512 bytes long, if the packet has a valid CRC value, it is considered valid. The remote wake-up packet detection is updated in the PMT Control and Status register for every remote wakeup packet received. A PMT interrupt to the application triggers a Read to the PMT Control and Status register to determine reception of a remote wake-up packet.

Notes

1. *The watchdog timeout limit for a remote wake-up frame is 2,048 bytes irrespective of the value programmed in WD bit of MAC_Configuration register and PWE bit in MAC_Watchdog_Timeout register.*
2. *The value programmed in DCRCC bit of MAC_Ext_Configuration register is applicable to a remote wake-up frame only when Enable IPv4 ARP Offload is selected in the configuration.*

PMT Block Registers

Use the RWKPKTEN bits of the MAC_PMT_Control_Status register to generate power management events. The application should program these bits. You can access the PMT registers in the similar manner as you access the MAC CSR registers.

wkuppktfilter_reg0	Filter 0 Byte Mask							
wkuppktfilter_reg1	Filter 1 Byte Mask							
wkuppktfilter_reg2	Filter 2 Byte Mask							
wkuppktfilter_reg3	Filter 3 Byte Mask							
wkuppktfilter_reg4	RSVD	Filter 3 Command	RSVD	Filter 2 Command	RSVD	Filter 1 Command	RSVD	Filter 0 Command
wkuppktfilter_reg5	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset	
wkuppktfilter_reg6	Filter 1 CRC - 16				Filter 0 CRC - 16			
wkuppktfilter_reg7	Filter 3 CRC - 16				Filter 2 CRC - 16			
wkuppktfilter_reg8	Filter 4 Byte Mask							
wkuppktfilter_reg9	Filter 5 Byte Mask							
wkuppktfilter_reg10	Filter 6 Byte Mask							
wkuppktfilter_reg11	Filter 7 Byte Mask							
wkuppktfilter_reg12	RSVD	Filter 7 Command	RSVD	Filter 6 Command	RSVD	Filter 5 Command	RSVD	Filter 4 Command
wkuppktfilter_reg13	Filter 7 Offset		Filter 6 Offset		Filter 5 Offset		Filter 4 Offset	
wkuppktfilter_reg14	Filter 5 CRC - 16				Filter 4 CRC - 16			
wkuppktfilter_reg15	Filter 7 CRC - 16				Filter 6 CRC - 16			
wkuppktfilter_reg16	Filter 8 Byte Mask							
wkuppktfilter_reg17	Filter 9 Byte Mask							
wkuppktfilter_reg18	Filter 10 Byte Mask							
wkuppktfilter_reg19	Filter 11 Byte Mask							
wkuppktfilter_reg20	RSVD	Filter 11 Command	RSVD	Filter 10 Command	RSVD	Filter 9 Command	RSVD	Filter 8 Command
wkuppktfilter_reg21	Filter 11 Offset		Filter 10 Offset		Filter 9 Offset		Filter 8 Offset	
wkuppktfilter_reg22	Filter 9 CRC - 16				Filter 8 CRC - 16			
wkuppktfilter_reg23	Filter 11 CRC - 16				Filter 10 CRC - 16			
wkuppktfilter_reg24	Filter 12 Byte Mask							
wkuppktfilter_reg25	Filter 13 Byte Mask							
wkuppktfilter_reg26	Filter 14 Byte Mask							
wkuppktfilter_reg27	Filter 15 Byte Mask							
wkuppktfilter_reg28	RSVD	Filter 15 Command	RSVD	Filter 14 Command	RSVD	Filter 13 Command	RSVD	Filter 12 Command
wkuppktfilter_reg29	Filter 15 Offset		Filter 14 Offset		Filter 13 Offset		Filter 12 Offset	
wkuppktfilter_reg30	Filter 13 CRC - 16				Filter 12 CRC - 16			
wkuppktfilter_reg31	Filter 15 CRC - 16				Filter 14 CRC - 16			

Figure 706 Remote Wake-Up Packet Filter Register

44.3.12.3.2 PMT Interrupt Signals

This topic describes the power management (PMT) interrupt signal that MAC generates when it enters or exits the power-down mode.

The PMT interrupt signal is asserted when a valid remote wake-up packet is received. In addition to the mci_intr_o (sbd_intr_o in EQOS-DMA configurations), the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that you can stop the application clock when the MAC is in the power-down mode.

The pmt_intr_o signal is generated in the Rx clock domain. It is not cleared immediately when the PMT Control and Status register is read. This is because the resultant clear signal has to cross to the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the DWC_ether_qos is operating in the 10 Mbps mode.

When software resets the PWRDWN bit in Remote Wake-Up Packet Detection register, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

44.3.12.4 System Considerations During Power Down

This topic describes the recommended power-down and wake-up sequence when the controller is running in low-power mode.

The DWC_ether_qos neither gates nor stops clocks when the power-down mode is enabled. Power saving by clock gating must be done outside the core by the application. The receive data path must be clocked with clk_rx_i during the power-down mode, because it is involved in the magic packet or remote wake-up packet detection. However, the application path clock can be gated during the power-down mode. The transmit path clock must be gated during the power-down mode only if ARP Offload is not enabled.

The recommended power-down and wake-up sequence is as follows:

1. Disable the Transmit DMA (if applicable) and wait for any previous packet transmissions to complete. These transmissions can be detected when Transmit Interrupt [Bit 0 of the DMA_CH0_Status register] is received.
2. Disable the MAC transmitter and MAC receiver by clearing the appropriate bits in the MAC Configuration register.
3. Wait until the Receive DMA empties all the packets from the Rx FIFO to system memory. You can do this by reading the appropriate bits of Debug registers in the DMA and MTL CSR space.
4. Enable Power-Down mode by appropriately configuring the PMT registers.
5. Enable the MAC Receiver and enter the Power-Down mode.
6. Gate the application and transmit clock (transmit clock must be gated only if ARP Offload is not enabled) inputs to the core (and other relevant clocks in the system) to reduce power and enter Sleep mode.

   Note: On receiving a valid remote wake-up packet, the DWC_ether_qos asserts the pmt_intr_o signal and exits the Power-Down mode.

   Note: On receiving the interrupt, the system must enable the application and transmit clock inputs to the core.

7. Read the PMT Status register to clear the interrupt, then enable the other modules in the system and resume normal operation.

44.3.13 Using MAC Management Counters

44.3.13.1 Introduction to MAC Management Counters

The DWC_ether_qos supports storing the statistics about the received and transmitted packets in registers that are accessible through the application.

The counters in the MAC Management Counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the MAC Control Interface (MCI). Each register is 32-bits wide. The write data is qualified with the corresponding mci_be_i signals. Therefore, non-32-bit accesses are allowed as long as the address is word-aligned. The MMCs are accessed using transactions, in the same way the CSR address space is accessed.

The MMC counters are free running. There is no separate enable for the counters to start. If a particular MMC counter is present in the RTL, it starts counting when corresponding packet is received or transmitted. The Receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets, dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, set Bit 0 in the “MAC_Packet_Filter” register.

The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets. This gathering is only enabled when you select the Full Checksum Offload Engine in coreConsultant.

You can select the MMC counters individually during configuration. The addresses for counters that are not selected become reserved. The width of each MMC counter is 32-bit, by default. You can individually change the width to 16-bit for each selected MMC counter during configuration.

For MMC register details, see registers “MMC_Control” to “RxICMP_Error_Octets”.

44.3.13.2 Address Assignments

The MMC registers follow a certain naming convention and their descriptions use certain terminologies that you should know.

The MMC register naming conventions are as follows.

- “tx” as a prefix or suffix indicates counters associated with transmission
- “rx” as a prefix or suffix indicates counters associated with reception
- “_g” as a suffix indicates registers that count only good packets
- “_gb” as a suffix indicates registers that count packets regardless of whether they are good or bad.

The following definitions define the terminology used in MMC register descriptions.

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber Timeout
  - No Carrier or Loss of Carrier
  - Late Collision
  - Packet Underflow
  - Excessive Deferral
  - Excessive Collision
- Received packets are considered “good” if none of the following errors exists:

  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbps only)
  - Length error (non-Type packet only)
  - Out of Range (non-Type packet only, longer than 1518 bytes)
  - GMII_RXER Input error
- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN Frame maxsize = 1,522
  - Jumbo Frame maxsize = 9,018
  - JumboVLAN Frame maxsize = 9,022
- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the **Table 471** table.

Table 471 Size of the Maximum Receive Packet

JE	S2KP	GPSLCE	EDVLP	Untagged Frame maxsize in bytes	Single VLAN Frame maxsize in bytes	Double VLAN Frame maxsize in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

Notes

1. *The MMC counters registers at the following offset addresses are of type Read-Only and have the default value of 0:*
   0x0714 to 0x0778
   0x0780 to 0x07E4
   0x0810 to 0x0844
   0x0850 to 0x0884
2. *The RX MMC counters and RX IPC MMC counters are updated only for packets that pass the Destination Address filter except when the UCDBC bit is set for broadcast packets in the MMC_Control register.*

44.3.14 Flow Control

This chapter describes the flow control for Transmit and Receive paths.

Note: Priority Flow Control (PFC) is not supported.

44.3.14.1 Transmit Flow Control

The Transmit Flow Control involves transmitting Pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end.

44.3.14.1.1 Flow Control in Full-Duplex Mode

DWC_ether_qos supports full-duplex flow control operations. In full-duplex mode, the DWC_ether_qos uses one of the following packet types for flow control:

- IEEE 802.3x Pause packets
- Priority flow control (PFC) packets

The PFC packets are used only when the Enable Data Center Bridging option is selected. The PFCE bit ofMAC_Rx_Flow_Ctrl register determines whether PFC packets or IEEE 802.3x Pause Control packets are used for flow control. If the PFCE bit is set, the MAC sends the PFC packets.

Pause Packet Structure

The table below describes the fields of a Pause packet.

Table 472 Size of the Maximum Receive Packet

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control opcode	Contains 0001 for IEEE 802.3x Pause Control packets; 0101 for PFC packets
PT	Contains Pause time specified in the PT field of the MAC_Q#_Tx_Flow_Ctrlregister

Pause Packet Control

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Similarly, when the mti_flowctrl_i signal is asserted, the MAC generates and transmits a single Pause packet. If the mti_flowctrl_i signal remains asserted at a configurable number of slot times before the Pause time runs out, the MAC transmits a second Pause packet. This process is repeated as long as the mti_flowctrl_i signal remains active. If the mti_flowctrl_i signal goes inactive prior to the sampling time, the MAC transmits a Pause packet with zero Pause time (if the DZPQ bit in MAC_Q#_Tx_Flow_Ctrl register is set to 0) to indicate to the remote end that the Receive buffer is ready to receive new data packets.

For PFC packets, you can specify the priority, pause time, and other controls for a queue in MAC_Q#_Tx_Flow_Ctrl register. The MAC supports independent flow control for each Rx queue. There is one trigger input for each Rx queue. Therefore, when multiple triggers come simultaneously, the MAC sends one PFC packet for each trigger. Based on the hardware trigger or software trigger through respective queues, the MAC sends a PFC packet with programmed Pause Time and priority. If multiple priorities are programmed in same Rx queue, multiple priorities are set for the PFC packet with same Pause Time values. A separate pause timer is available for each Rx queue.

44.3.14.1.2 Flow Control in Half-Duplex Mode

In half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

The table below describes the flow control in the Tx path for Queue 0 based on the setting of the following bits:

- EHFC bit of MTL_RxQ0_Operation_Mode register
- TFE bit of MAC_Q0_Tx_Flow_Ctrl register
- DM bit of MAC_Configuration register

Flow control is similar for all queues.

Table 473 Size of the Maximum Receive Packet

EHFC	TFE	DM	Description
x	0	x	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs back-pressure when Bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1.
1	1	0	The MAC transmitter performs back-pressure when Bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx performs back-pressure when Rx Queue level crosses the threshold set by Bits[10:8] of MTL_RxQ0_Operation_Mode register.
0	1	1	The MAC transmitter sends the Pause packet when Bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1.
1	1	1	The MAC transmitter sends the Pause packet when Bit 0 of MAC_Q0_Tx_Flow_Ctrl register is set or the sideband signal sbd_flowctrl_i is 1. In addition, the MAC Tx sends a Pause packet when Rx Queue level crosses the threshold set by Bits[10:8] of MTL_RxQ0_Operation_Mode register.

44.3.14.2 Triggering Transmit Flow Control

The Transmit Flow Control involves transmitting Pause packets in full-duplex mode and backpressure in half-duplex mode to control the flow of packets from the remote end. The application can request the MAC to send a Pause packet or initiate back-pressure by using either of the following method:

The application sets the FCB_BPA bit in the corresponding MAC_Q#_Tx_Flow_Ctrl register.

The flow control is triggered based on the following:

Rx Queue Threshold: The flow control operation of the MAC is enabled when the EHFC bit of corresponding MTL_RxQ#_Operation_Mode register is set. The flow control signal to the MAC is asserted when the fill level of the Rx queue crosses the threshold configured in RFA field of MTL_RxQ#_Operation_ Mode register. This flow control signal is de-asserted when the fill-level of the queue falls below the threshold configured in the RFD field.

The hardware flow control generated based on the Rx Queue threshold crossing condition is applicable only when the Rx queue size is 4,096 bytes or more.

Note: The RFS sets the threshold at which the flow control is triggered and the MAC schedules to send a PAUSE frame to be sent by the local transmitter. The remote transmitter can continue to send packets until it receives this PAUSE packet. So the RXQ needs space to take in this data even after the flow-control is triggered, to avoid overflow and loss of packets. The max-space required that is calculated theoretically can be 2 max-sized packets + a little more assuming that the read from the RXFIFO is not occurring during this whole time. Theoretically, this makes RFA=4 or more, which implies that RxQ must have at the least 4KB size. Practically, the system works with RFA=2 (Full – 2KB, with very little probability of overflow). If you have FIFO size of less than 4K, then with RFA=2, the flow control will be triggered with just 1000B in the RxQ which is even less than max packet size. The RFA field MTL_RxQ[n]_Operation_Mode register is used for activating the Flow Control. These bits control the threshold (fill-level of Rx queue) at which the flow control is activated. For Example: If your RXQ size is

2K and you set the RFA to 1K; as soon as you receive 1K, the Flow Control is activated. If you are sending a packet of 1500 bytes (which is allowed as per spec), you will be triggering flow-control even before you receive a complete packet. This will reduce the throughput considerably. If your RXQ size is 4K, you can set the RFA to 2K, so that you are able to receive the whole packet.

44.3.14.3 Receive Flow Control

In the Receive path, the Flow Control is functional only in the full-duplex mode. If any Pause packet is received in the half-duplex mode, the packet is considered as a normal control packet.

Note: Receive pause packets should have a frame size of 64 bytes.

44.3.14.3.1 Description of Receive Flow Control

The Receive Flow Control is implemented by the MAC based on the bit value of the respective register, and the destination address and different fields of the received packet.

The table below describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of MAC_Rx_Flow_Ctrl register
- DM bit of MAC_Configuration register

Table 474 Size of the Maximum Receive Packet

TFE	DM	Description
0	x	The MAC receiver does not detect the received Pause packets.
1	0	The MAC receiver does not detect the received Pause packets butrecognizes such packets as Control packets.
1	0	The MAC receiver detects or processes the Pause packets andresponds to such packets by stopping the MAC transmitter.

You can enable the PFC packet detection by setting the PFCE bit in the MAC_Rx_Flow_Ctrl register. If PFC packet detection is enabled, the MTL Tx queue corresponding to the received priority is blocked when the PFC packet is received. If PFC packet detection is not enabled in configurations with multiple queues and RFE bit is enabled, the MAC transmitter is blocked when the 802.3x Pause packet is received.

The following list describes the Rx flow control:

1. The MAC checks the destination address of the received Pause packet for either of the following:
   a) Multicast destination address: The DA matches the unique multicast address specified for the control packet (48'h0180C2000001).
   b) Unicast destination address: The DA matches the content of the MAC Address Register 0 and the UP bit of MAC_Rx_Flow_Ctrl register is set. If the UP bit is set and the MAC processes Pause packets with unicast destination address in addition to the unique multicast address.
2. The MAC decodes the following fields of the received packet:
   a) Type field: This field is checked for 16'h8808.
   b) Opcode field: This field is checked for 16'h0001 (Pause packet) or 16'h0101 (PFC packet).
   c) Pause Time or Pause Time Vector field: The Pause time (for Pause packet) is captured to determine the time for which transmitter needs to be blocked. The Pause Time Vector field (for PFC packet) is captured to determine the time for which the MTL Tx queue corresponding to the received priority needs to be blocked.Priority Enable Vector field: This field is valid only for PFC packets. It is captured to determine the MTL Tx queue corresponding to the received priority.

3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter does one of the following:
   a) For 802.3x Pause packets, the MAC pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times).
   b) For PFC packets, the MAC blocks the Tx queues corresponding to the priority. Based on the priorities assigned to the Tx queues in MAC_TxQ_Prty_Map0 and MAC_TxQ_Prty_Map1 registers, the MAC asserts the respective bit in the mti_disable_txq_o signal and loads the Pause timer corresponding to the priority. The Tx queue corresponding to the priority is blocked till the Pause timer expires. In addition to the PFC-based flow control operation, when PFCE bit is set, all bits of the mti_disable_txq_o signal are asserted while in LPI mode so that the ETS or CBS scheduler start over again after exiting the LPI mode.
4. The MAC transfers the received control packet to the application based on the setting of the PCF field in MAC_Packet_Filter register.

If subsequent Pause or PFC packets are received before the earlier Pause Time expires, the MAC updates thePause Timer with new value.

44.3.14.4 Enabling Receive Flow Control

To enable Pause Flow Control, set the RFE bit in the MAC_Rx_Flow_Ctrl register.

44.3.15 Using Loopback Mode

44.3.15.1 Guidelines for Using Loopback Mode

You need to follow certain guidelines to use the loopback feature of the DWC_ether_qos controller.

The following are some guidelines for using the loopback mode:

- Enable loopback only with the full-duplex mode. In half-duplex mode, the carrier sense signal (crs) or collision (col) signal inputs get sampled which may result into issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip (for example, in FPGA setup), you should externally generate the Tx and Rx clocks and provide these clocks to the MAC.
- Do not loop back big packets. Big packets may get corrupted in the loopback FIFO.

The Transmit and Receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the phy_txd_o data to the Receive path. The asynchronous FIFO is 10-bits (6-bits in 10/100 Mbps mode) wide to accommodate phy_txd_o, phy_txen_o, and phy_txer_o. The depth of FIFO is five in 1000 Mbps mode and nine in 10/100 Mbps mode. The FIFO is free-running to write on the write clock (clk_tx_i) and read on every read clock (clk_rx_i).

At the start of each packet read out of the FIFO, the Write and Read pointers get re-initialized to have an offset of 2 (4 in 10/100 Mbps mode). This avoids overflow or underflow during a packet transfer. This also ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between (G)MII Transmit and Receive clock frequencies. Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO.

At the end of every received packet, the Receive Protocol Engine module generates received packet status and sends it to the Receive Packet Controller module. The control, missed packet, and filter fail status are added to the Receive status in the Receive Packet Controller module.

The MAC does not process ARP or PMT packets that are looped back.

44.3.15.2 Enabling Loopback Mode

The MAC supports Loopback of transmitted packets to its receiver.

To enable this feature, program the LM bit of the MAC_Configuration register.

You can enable loopback for all PHY interfaces. The data is always looped back on the MII or GMII interface irrespective of which PHY interface is selected. The loopback data is also passed through the corresponding interface block.

44.3.16 Interrupts from the MAC

Various events in optional modules of the MAC can generate interrupts.

Interrupts can be generated from the MAC as a result of various events in the optional modules.

In EQOS-DMA and EQOS-AHB configurations, these interrupt events are combined with the events in the DMA on the sbd_intr_o signal.

The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The MAC_Interrupt_Status register describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt on the mci_intr_o or sbd_intr_o signals by setting the corresponding mask bits in the MAC_Interrupt_Enable register.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt. For example, when set high, Bit 0 of the MAC_Interrupt_Status register indicates that the link status on the RGMII, SGMII, or SMII interface has changed. You must read the MAC_PHYIF_Control_Status register to clear this interrupt event.

The interrupts from the RGMII, and PCS blocks and the optional General Purpose Inputs are combined (OR’ed) and given as the GLI bit in the DMA_Interrupt_Status register.

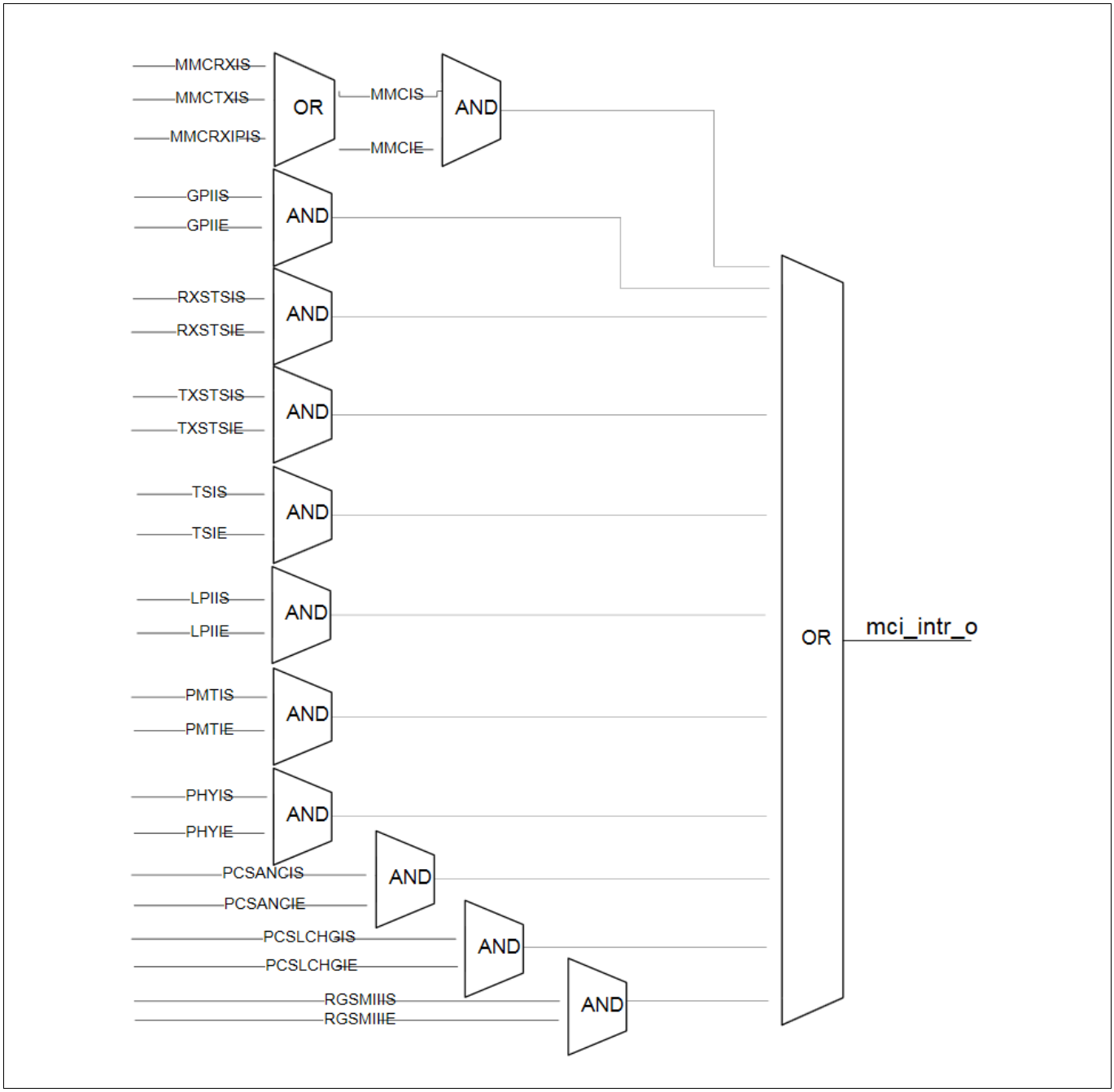


Figure 707 MAC Interrupt Enabling Scheme

Note: By default, the MAC interrupt status bits are cleared when the register that contains the source of the interrupt is read. If RCWE bit in MAC_CSR_SW_Ctrl register is programmed to 1, the MAC interrupt status bits are cleared when the bit that contains the source of the interrupt is explicitly written to 1.

44.3.17 Descriptors

44.3.17.1 Overview of Descriptors

The DWC_ether_qos controller support two types of descriptors: Norma and Context.

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory. The DWC_ether_qos supports the following two types of descriptors:

- Normal Descriptor: Normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.
- Context Descriptor: Context descriptors are used to provide control information applicable to the packet to be transmitted.

Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

Note: There is no limit for the number of descriptors that can be used for a single packet.

44.3.17.2 Descriptor Structure

The DWC_ether_qos supports the ring structure for a DMA descriptor, which separates descriptors by the Word, DWord, or LWord number programmed in the DMA_CH#_Control.DSL register field.

The DWC_ether_qos supports the ring structure for DMA descriptor as shown in **Figure 708**.

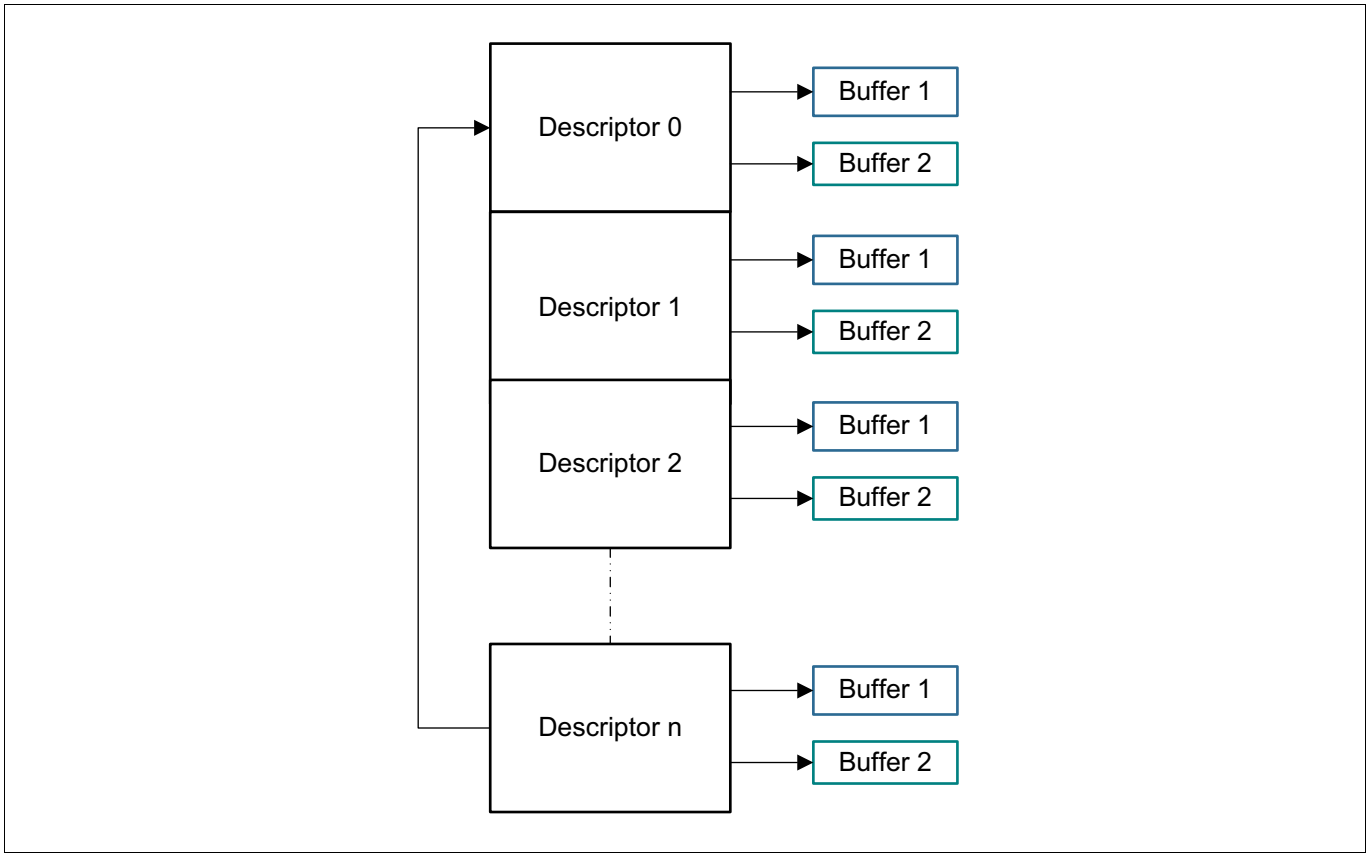


Figure 708 Descriptor Ring Structure

In Ring structure, descriptors are separated by the Word, DWord, or LWord number programmed in the DSL field of the DMA_CH#_Control register. The application needs to program the total ring length, that is, the total number of descriptors in ring span in the following registers of a DMA channel:

- Transmit Descriptor Ring Length Register (DMA_CH#_TxDesc_Ring_Length)
- Receive Descriptor Ring Length Register (DMA_CH#_RxDesc_Ring_Length)

The Descriptor Tail Pointer Register contains the pointer to the descriptor address (N). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (N – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA goes into the Suspend mode when this condition occurs. The application must perform a write to the Descriptor Tail pointer register and update the tail pointer so that the following condition is true:

```
Current Descriptor Pointer < Descriptor Tail Pointer;
```

The DMA automatically wraps around the base address when the end of ring is reached, as shown in the following figure.

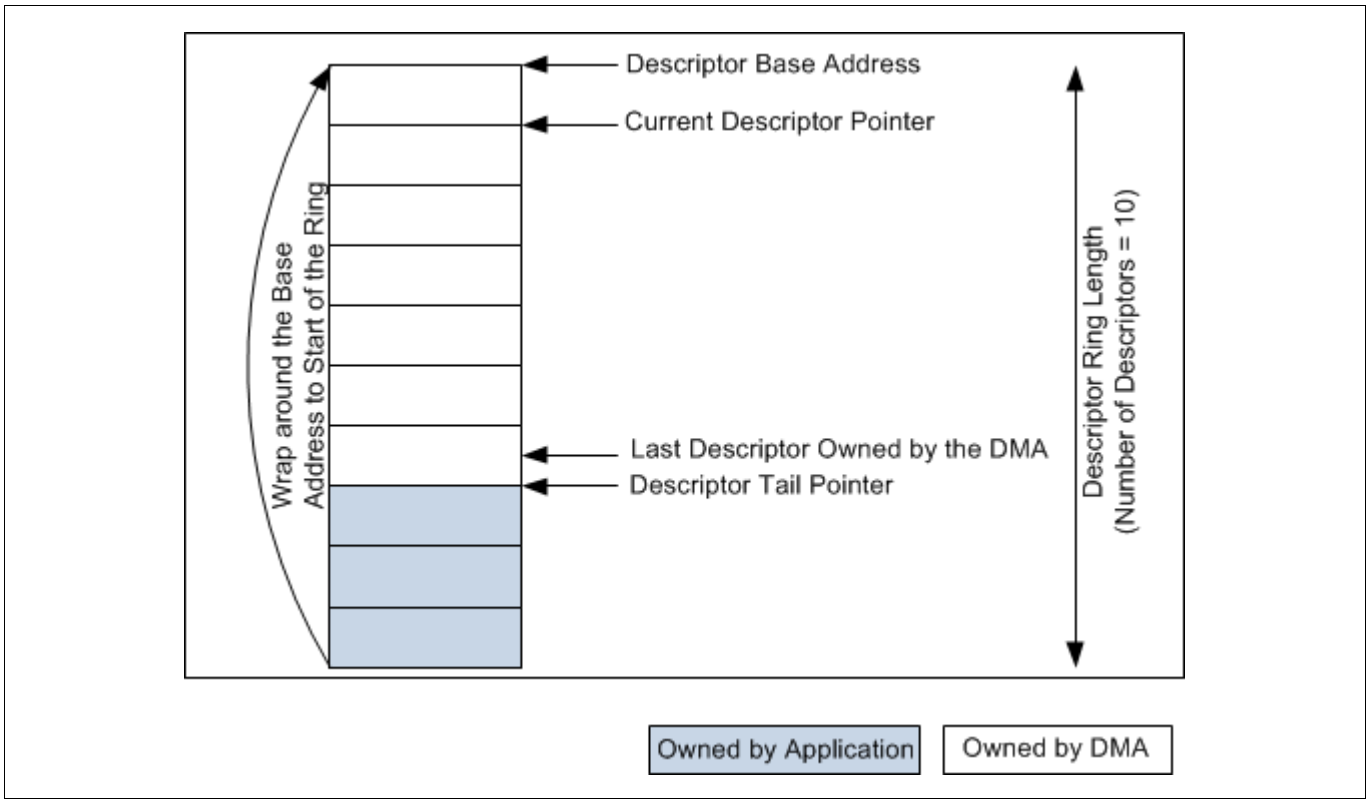


Figure 709 DMA Descriptor Ring

For descriptors owned by the application, the OWN bit of DES3 is reset to 0. For descriptors owned by the DMA, the OWN bit is set to 1. If the application has only one descriptor in the beginning, the application sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA processes the first descriptor and then waits for the application to advance the tail pointer.

44.3.17.3 Split Header Support

You can enable the optional split header support by programming the DMA_CH#_Control.SPH register bit, which allows the DMA to process the header and payload of a received packet separately.

Split Header support is an optional feature. You can enable the split header feature by selecting the Enable Split Header Feature option and setting the SPH bit in the DMA_CH#_Control register.

The DMA can process the header and payload of received packets separately.

Table 475 shows how DMA processes a packet depending upon its type.

Table 475 Header Split Points

Packet Type	Description
TCP or UDP Packet	The DMA writes the Ethernet header + IP header + TCP or UDP header into the header buffer
IP packet (not TCP/UDP)	The DMA writes the Ethernet header + IP header into the header buffer
Non-IP packet	The DMA does not split the header and payload

The IP header includes IPv4 options in case of a IPv4 packet, and IPv6 extension headers in case of IPv6 frames. The points at which the header is split are shown in **Figure 710**.

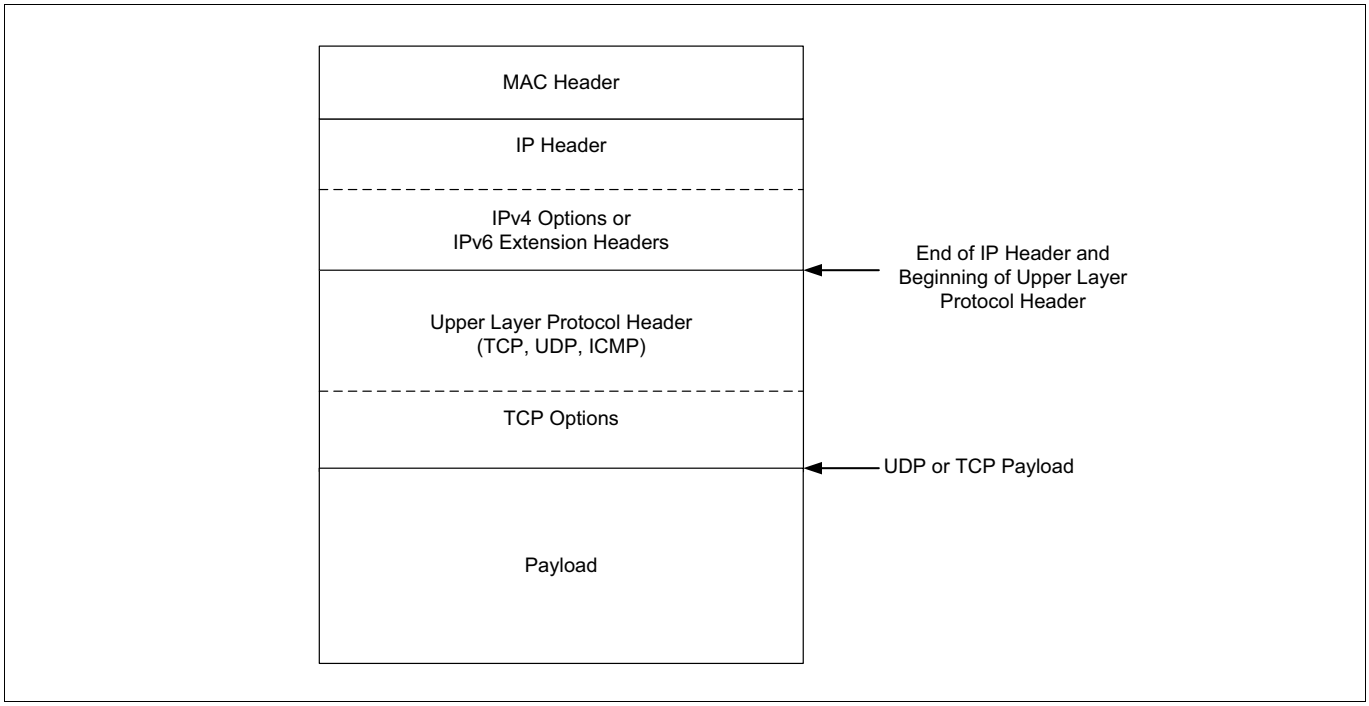


Figure 710 Header Split Points

44.3.17.3.1 Descriptor Structure with Split Header Feature

The DWC_ether_qos supports the split header structure for descriptors.

Figure 711 shows the descriptor structure without the Split Header feature.

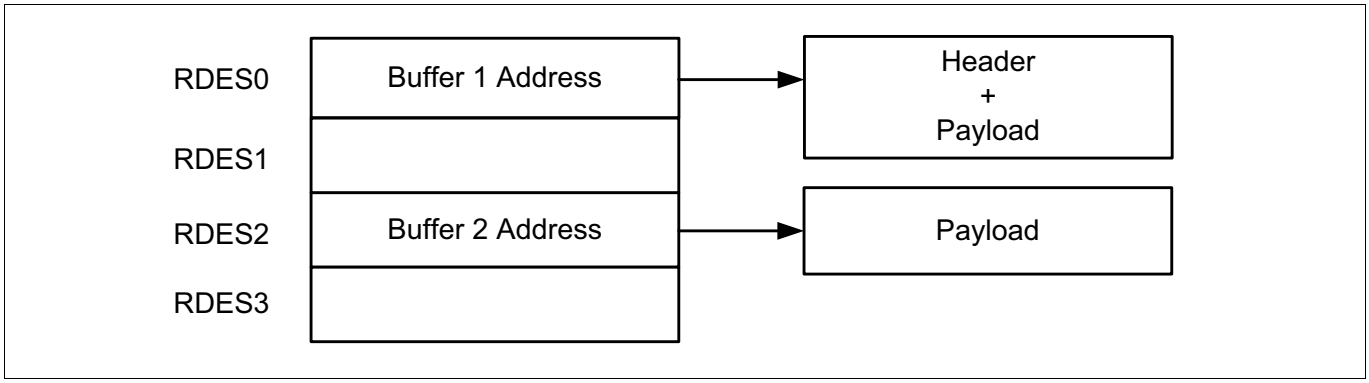


Figure 711 Descriptors without Split Header Feature

Figure 712 shows the descriptor structure with the Split Header feature.

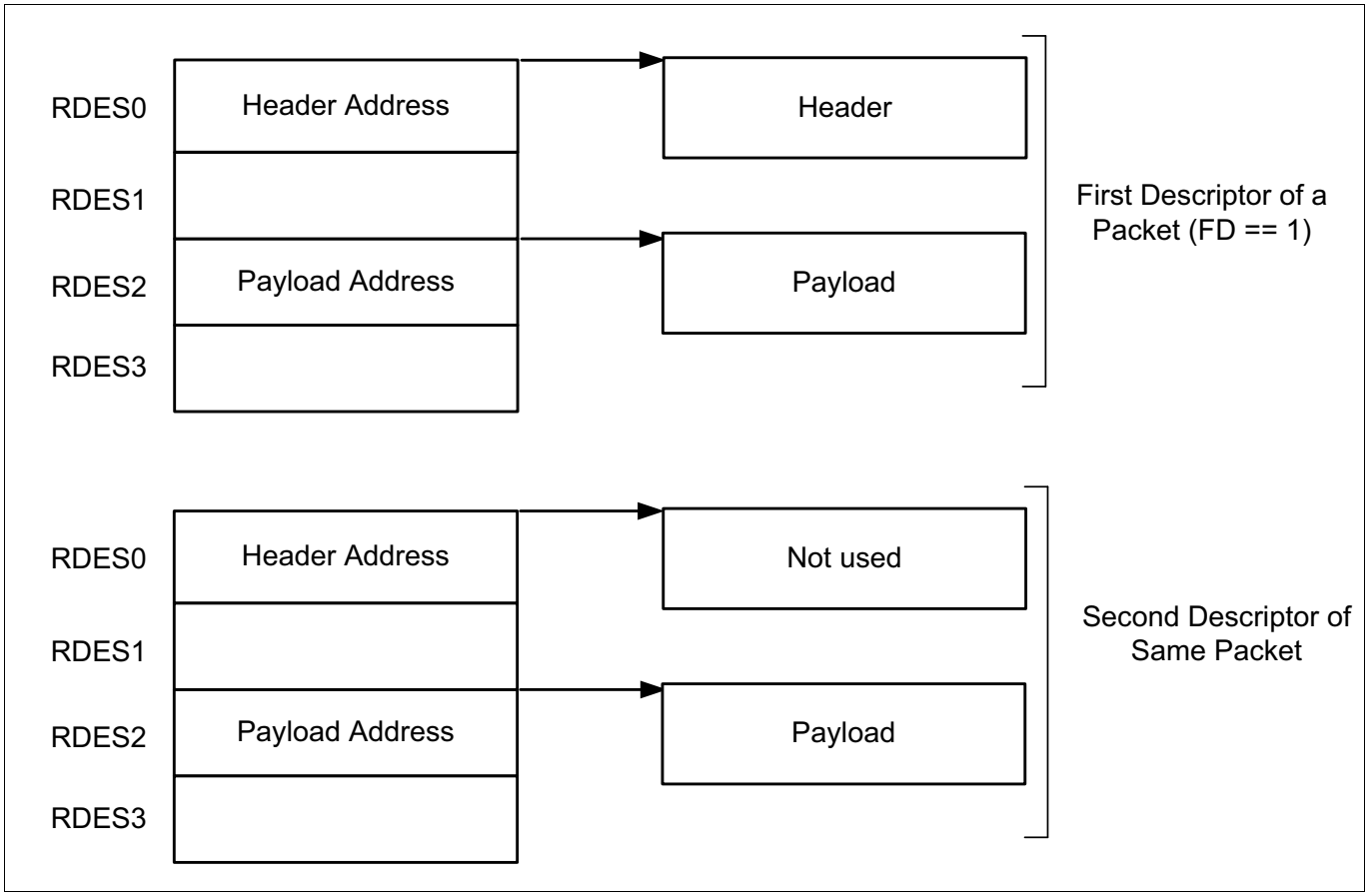


Figure 712 Descriptors with Split Header Feature

The DMA writes the header of the received packet by using the header address to which the RDES0 in the first descriptor is pointing (FD bit of RDES3 is set). The DMA writes the payload of the received packet into the buffer address to which the RDES2 is pointing. For subsequent descriptors (FD is set to 0), the address to which RDES0 (Header address) is pointing is not used. The payload is written only to buffers to which the RDES2 (payload address) is pointing.

The DMA writes the header length in RDES2 of the first receive descriptor (RDES3[29] (FD Bit) is set) for the packet. The packet length is written in RDES3 of the last receive descriptor (RDES3[28] (LD Bit) set). The buffer length for the payload is set by the driver through the RBSZ field in the corresponding DMA Channel Register 2 (Receive Control Register). The DMA fills receive buffers fully in all except the last descriptor. The header length is taken to

be the value based on the bits programmed in the MAC Extended Configuration Register HDSMS field (Bits [22:20]).

44.3.17.4 Descriptor Endianness

You must align the descriptor address to the configured bus width in terms of endianness.

Figure 713 shows the normal Receive and Transmit descriptors for 32-bit data bus when the endian mode of data bus and descriptors is little-endian.

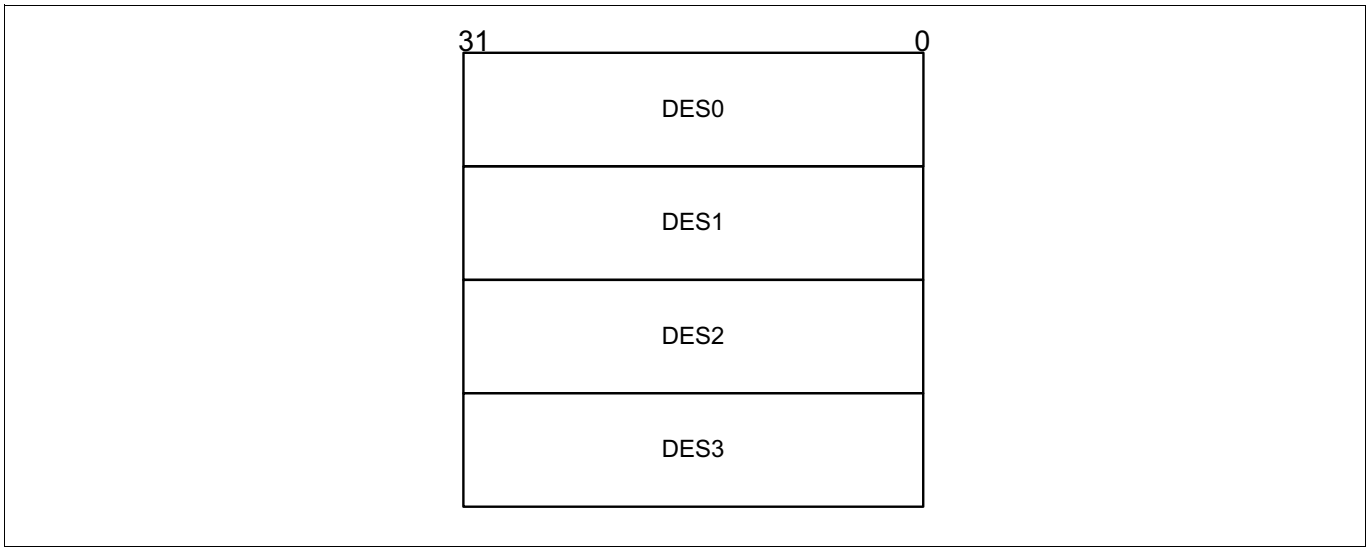


Figure 713 Rx/Tx Descriptors in Same-Endian Mode for 32-Bit Little-Endian or Reverse Endian Mode for 32-Bit Big-Endian Data Bus

44.3.17.5 Transmit Descriptor

The DWC_ether_qos support two types of Transmit Descriptor formats: Read and Write-back.

The DMA in DWC_ether_qos requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit packet basis. The Transmit Normal descriptor has two formats: Read format and Write-Back format

44.3.17.5.1 Transmit Norma Descriptor (Read Format)

Figure 714 shows the Read Format for a Transmit normal descriptor. **Table 476** through **Table 479** describe the read format for the Transmit Normal Descriptors: TDES0, TDES1, TDES2, and TDES3.

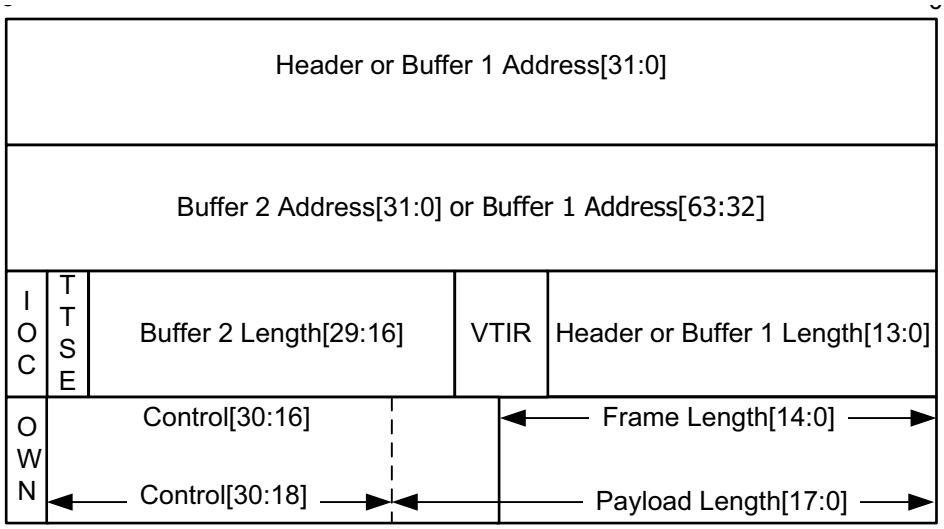


Figure 714 Transmit Descriptor Read Format

Table 476 TDES0 Normal Descriptor (Read Format)

Bit	Name	Description
31:0	BUF1AP	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1.

Table 477 TDES1 Normal Descriptor (Read Format)

Bit	Name	Description
31:0	BUF2AP	Buffer 2 or Buffer 1 Address Pointer This bit indicates the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation for the buffer address alignment.In 40- or 48-bit addressing mode, these bits indicate the most-significant 8- or 16- bits of the Buffer 1 Address Pointer.

Table 478 TDES2 Normal Descriptor (Read Format)

Bits	Name	Description
31	IOC	Interrupt on Completion This bit sets the TI bit in the DMA_CH#_Status register after the present packet has been transmitted.
30	TTSE/TMWD	Transmit Timestamp Enable This bit enables the IEEE1588 time stamping for Transmit packet referenced by the descriptor, if TSE bit is not set.
29:16	B2L	Buffer 2 Length The driver sets this field. When set, this field indicates Buffer 2 length.

Table 478 TDES2 Normal Descriptor (Read Format) (cont'd)

Bits	Name	Description
15:14	VTIR	VLAN Tag Insertion or Replacement These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN Tag Insertion, Replacement, or Deletion is enabled for the packet. The following list describes the values of these bits: • 2'b00: Do not add a VLAN tag. • 2'b01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. • 2'b10: Insert a VLAN tag with the tag value programmed in the MAC_VLAN_Incl register or context descriptor. • 2'b11: Replace the VLAN tag in packets with the tag value programmed in the MAC_VLAN_Incl register or context descriptor. This option should be used only with the VLAN packets. These bits are valid when the Enable SA and VLAN Insertion on Tx option is selected while configuring the core.
13:0	B1L	Buffer 1 Length For Header length only bits [9:0] are taken. The size 13:0 is applicable only when interpreting buffer 1 length.

Table 479 TDES3 Normal Descriptor (Read Format)

Bits	Name	Description
31	OWN	Own Bit When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
30	CTXT	Context TypeThis bit should be set to 1'b0 for normal descriptor.
29	FD	First Descriptor When this bit is set, it indicates that the buffer contains the first segment of a packet.
28	LD	Last Descriptor When this bit is set, it indicates that the buffer contains the last segment of the packet. When this bit is set, the B1L or B2L field should have a non-zero value.

Table 479 TDES3 Normal Descriptor (Read Format) (cont'd)

Bits	Name	Description
27:26	CPC	CRC Pad Control This field controls the CRC and Pad Insertion for Tx packet. This field is valid only when the first descriptor bit (TDES3[29]) is set. The following list describes the values of Bits[27:26]: • 2'b00: CRC and Pad Insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packet of length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. • 2'b01: CRC Insertion (Disable Pad Insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit Buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. • 2'b10: Disable CRC Insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. • 2'b11: CRC Replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. This field is valid only for the first descriptor. Note: When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.
25:23	SAIC	SA Insertion Control These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must set the CRC Pad Control bits appropriately when SA Insertion Control is enabled for the packet. Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement.The following list describes the values of Bits[24:23]: • 2'b00: Do not include the source address • 2'b01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. • 2'b10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. • 2'b11: Reserved These bits are valid in the EQOS-DMA, EQOS-AXI, and EQOS-AHB configurations when the Enable SA and VLAN Insertion on Tx option is selected while configuring the core and when the First Segment control bit (TDES3 [29]) is set. This field is valid only for the first descriptor.

Table 479 TDES3 Normal Descriptor (Read Format) (cont'd)

Bits	Name	Description
22:19	SLOTNUMorTHL	SLOTNUM: Slot Number Control Bits in AV Mode These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the Transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field DMA_CH#_Slot_Function_Control_Status. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels. THL: TCP/UDP Header Length If the TSE bit is set, this field contains the length of the TCP header. The minimum value of this field must be 5. This field is valid only for the first descriptor.
18	RES_18	Reseved
17:16	CIC/TPL	Checksum Insertion Control or TCP Payload LengthThese bits control the checksum calculation and insertion. The following list describes the bit encoding: • 2'b00: Checksum Insertion Disabled. • 2'b01: Only IP header checksum calculation and insertion are enabled. • 2'b10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. • 2'b11: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is valid when the Enable Transmit TCP/IP Checksum Offload option is selected and the TSE bit is reset. When the TSE bit is set, this field contains the upper bits [17:16] of the TCP Payload length. This allows the TCP/UDP packet length field to be spanned across TDES3[17:0] to provide 256 KB packet length support. This field is valid only for the first descriptor.
15	TPL	Reserved or TCP Payload Length When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is Bit 15 of the TCP payload length [17:0]. This field is valid only when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected while configuring the core.
14:0	FL/TPL	Packet Length or TCP Payload Length This field is equal to the length of the packet to be transmitted in bytes. When the TSE bit is not set, this field is equal to the total length of the packet to be transmitted: `Ethernet Header Length + TCP /IP Header Length – Preamble Length – SFD Length + Ethernet Payload Length` When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length. This length does not include Ethernet header or TCP/IP header length.

44.3.17.5.2 Transmit Normal Descriptor (Write-Back Format)

The write-back format of the Transmit Descriptor includes timestamp low, timestamp high, OWN, and Status bits.

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 715 illustrates the write-back format of the Transmit Descriptor.

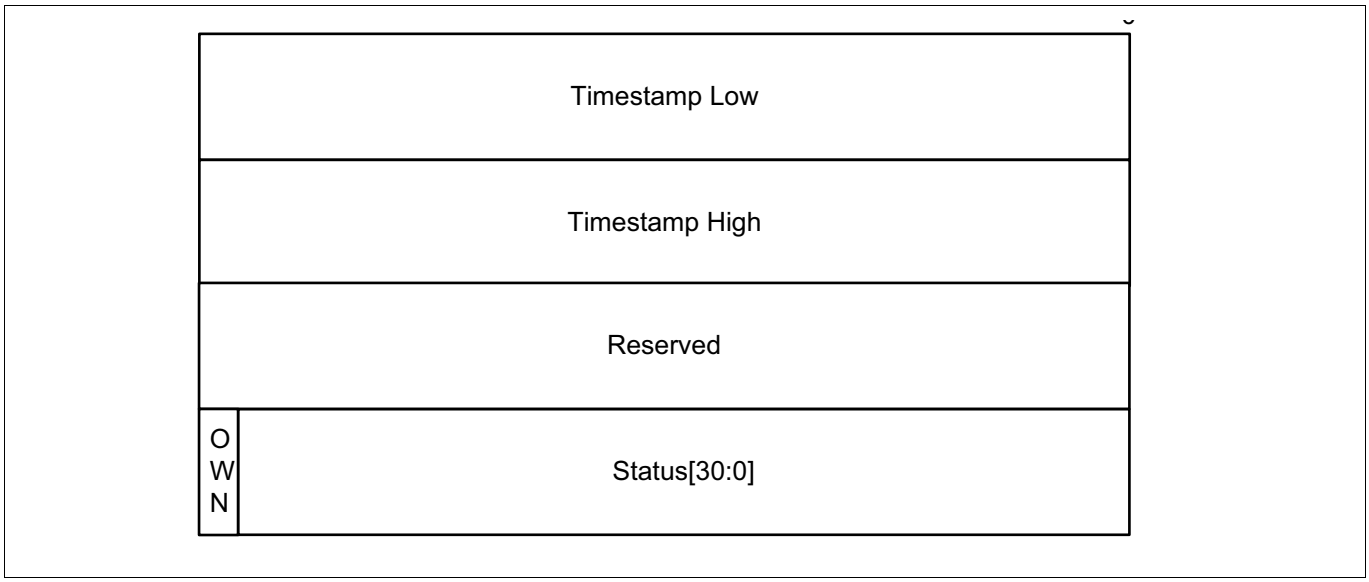


Figure 715 Transmit Descriptor Write-Back Format

As described in Table 480, this format is only applicable to the last descriptor of a packet.

Table 480 TDES0 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31:0	TTSL	Transmit Packet Timestamp Low The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.

As provided in Table 481, this format is only applicable to the last descriptor of a packet.

Table 481 TDES1 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31:0	TTSH	Transmit Packet Timestamp High The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding Transmit packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

As given in Table 482, this format is applicable only to the last descriptor of a packet.

Table 482 TDES2 Normal Descriptor (Write-Back Format)

Bit	Description
31:0	Reserved

As provided in Table 483, this format is applicable only to the last descriptor of a packet.

Table 483 TDES3 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31	OWN	Own Bit When this bit is set, it indicates that the DWC_ether_qos DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 'b0.
30	CTXT	Context Type This bit should be set to 'b0 for Normal descriptor.
29	FD	First Descriptor This bit indicates that the buffer contains the first segment of a packet.
28	LD	Last Descriptor This bit is set 'b1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
27:24	Rsvd	Reserved
23	DE	Descriptor Error When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the descriptor. Descriptor Errors can be: • Incorrect sequence from the context descriptor. For example, a location after the first descriptor for a packet. • All 1s • CTXT, LD, and FD bits set to 1. Note 1: When Descriptor Error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA will set the TI bit in the DMA_CH#_Status registerNote 2: Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.
22:18	Rsvd	Reserved
17	TTSS	Tx Timestamp Status This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES2 and TDES3 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set. This bit is valid only when IEEE1588 timestamping feature is enabled; otherwise, it is reserved.
16	Rsvd	Reserved

Table 483 TDES3 Normal Descriptor (Write-Back Format) (cont'd)

Bit	Name	Description
15	ES	Error Summary This bit indicates the logical OR of the following bits: • TDES3[0]: IP Header Error • TDES3[14]: Jabber Timeout • TDES3[13]: Packet Flush • TDES3[12]: Payload Checksum Error • TDES3[11]: Loss of Carrier • TDES3[10]: No Carrier • TDES3[9]: Late Collision • TDES3[8]: Excessive Collision • TDES3[3]: Excessive Deferral • TDES3[2]: Underflow Error
14	JT	Jabber Timeout This bit indicates that the MAC transmitter has experienced a jabber time-out. This bit is set only when the JD bit of the MAC_Configuration register is not set.
13	FF	Packet Flushed This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	Payload Checksum ErrorThis bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either because of insufficient bytes, as indicated by the Payload Length field of the IP Header or the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when Bus Error is detected during packet transfer. When the Full Checksum Offload engine is not enabled, this bit is reserved.
11	LoC	Loss of Carrier This bit indicates that Loss of Carrier occurred during packet transmission (that is, the gmii_crs_i signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the half-duplex mode.
10	NC	No Carrier This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	LC	Late Collision This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode and 512 byte times including Preamble and Carrier Extension in GMII mode). This bit is not valid if Underflow Error is set.

Table 483 TDES3 Normal Descriptor (Write-Back Format) (cont'd)

Bit	Name	Description
8	EC	Excessive Collision This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after first collision and the transmission of the packet is aborted.
7:4	CC	Collision Count This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	Excessive Deferral This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000 Mbps mode or Jumbo Packet enabled mode) if DC bit is set in the MAC_Configuration register.
2	UF	Underflow Error This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: • The DMA encountered an empty Transmit Buffer while transmitting the packet • The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the suspended state and sets the underflow bit corresponding to a queue in the MTL_Interrupt_Status register.
1	DB	Deferred Bit This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the half-duplex mode.
0	IHE	IP Header Error When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. This bit is valid only when Tx Checksum Offload is enabled. Otherwise, it is reserved. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

44.3.17.5.3 Transmit Context Descriptor

The context descriptor is used to provide the timestamps for one-step timestamp correction, VLAN Tag ID for VLAN insertion feature, and SA insertion bit for SA insertion.

The Transmit Context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction and VLAN Tag ID for VLAN insertion feature. Write back is done on a context descriptor only to reset the OWN bit.

Note: The VLAN Tag IDs and MSS values, provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA. When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet. The Inner VLAN Tag Control input is used only for the next packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

Figure 716 shows the format of the Transmit Context descriptor.

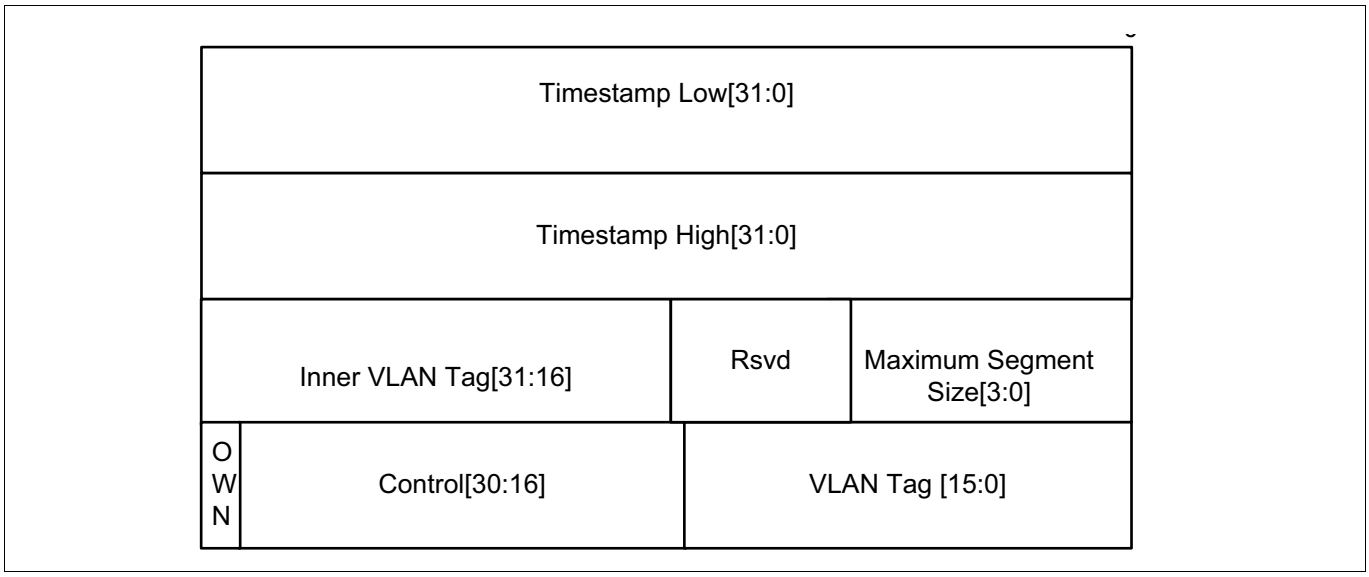


Figure 716 Transmit Context Descriptor Format

Table 484 describes the format of the TDES0 Context descriptor.

Table 484 TDES0 Context Descriptor

Bit	Name	Description
31:0	TTSL	Transmit Packet Timestamp Low For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

Table 485 provides the format of the TDES1 Context descriptor.

Table 485 TDES1 Context Descriptor

Bit	Name	Description
31:0	TTSH	Transmit Packet Timestamp High For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

Table 486 explains the format of the TDES2 Context descriptor.

Table 486 TDES2 Context Descriptor

Bit	Name	Description
31:16	IVT	Inner VLAN Tag When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.

Table 486 TDES2 Context Descriptor (cont'd)

Bit	Name	Description
15:14	Rsvd	Reserved
13:0	MSS	Maximum Segment Size When the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected, the driver can provide maximum segment size in this field. This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

Table 487 details the format of the TDES3 Context descriptor.

Table 487 TDES3 Context Descriptor

Bit	Name	Description
31	OWN	Own Bit When this bit is set, it indicates that the DWC_ether_qos DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit immediately after the read.
30	CTXT	Context Type This bit should be set to 1'b1 for Context descriptor.
29:28	Rsvd	Reserved
27	OSTC	One-Step Timestamp Correction Enable When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.
26	TCMSSV	One-Step Timestamp Correction Input or MSS Valid When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid.When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25:24	Rsvd	Reserved
23	DE	Descriptor Error When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the descriptor. Descriptor Errors can be: • Incorrect sequence from the context descriptor. For example, a location after the first descriptor for a packet. • All 1s • CTXT, LD, and FD bits set to 1. Note 1: When Descriptor Error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA will set the TI bit in the DMA_CH#_Status registerNote 2: Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.
22:20	Rsvd	Reserved

Table 487 TDES3 Context Descriptor (cont'd)

Bit	Name	Description
19:18	IVTIR	Inner VLAN Tag Insert or Replace When this bit is set, these bits request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. The following list describes the values of these bits: • 2'b00: Do not add the inner VLAN tag. • 2'b01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. • 2'b10: Insert an inner VLAN tag with the tag value programmed in the MAC_Inner_VLAN_Incl register or context descriptor. • 2'b11: Replace the inner VLAN tag in packets with the tag value programmed in the MAC_Inner_VLAN_Incl register or context descriptor. This option should be used only with the VLAN frames. These bits are valid when the Enable SA and VLAN Insertion on Tx and Enable Double VLAN Processing options are selected.
17	IVLTV	Inner VLAN Tag Valid When this bit is set, it indicates that the IVT field of TDES2 is valid.
16	VLTV	VLAN Tag Valid When this bit is set, it indicates that the VT field of TDES3 is valid.
15:0	VT	VLAN Tag This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the MAC_VLAN_Incl register is reset.

44.3.17.6 Receive Descriptor

The DWC_ether_qos controller supports two receive descriptors: normal and context.

The DMA in DWC_ether_qos attempts to read a descriptor only if the Tail Pointer is different from the Base Pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC. Otherwise, the performance of the DMA is impacted greatly because of the unavailability of the descriptors. In such situations, the RxFIFO in MTL becomes full and starts dropping packets.

The following Receive Descriptors are present:

- Normal descriptors
- Context descriptors

All RX descriptors are prepared by the software and given to the DMA as "Normal" Descriptors with the content as shown in Receive Normal Descriptor (Read Format). The DMA reads this descriptor and after transferring a received packet (or part of) to the buffers indicated by the descriptor, the Rx DMA will close the descriptor with the corresponding packet status. The format of this status is given in the "Receive Normal Descriptor (Write-Back Format)".

For some packets, the normal descriptor bits are not enough to write the complete status. For such packets, the RX DMA will write the extended status to the next descriptor (without processing or using the Buffers Pointers embedded in that descriptor). The format and content of this descriptor write back is described in "Receive Context Descriptor".

44.3.17.6.1 Receive Normal Descriptor (Read Format)

The read format for a Receive Normal descriptor is made up of a header or Buffer 1 address, reserved field, payload or Buffer 2 or Next Descriptor address, a 30-bit reserved filed, OWN bit, and an interrupt bit.

Figure 717 shows the Read Format for a Receive Normal Descriptor.

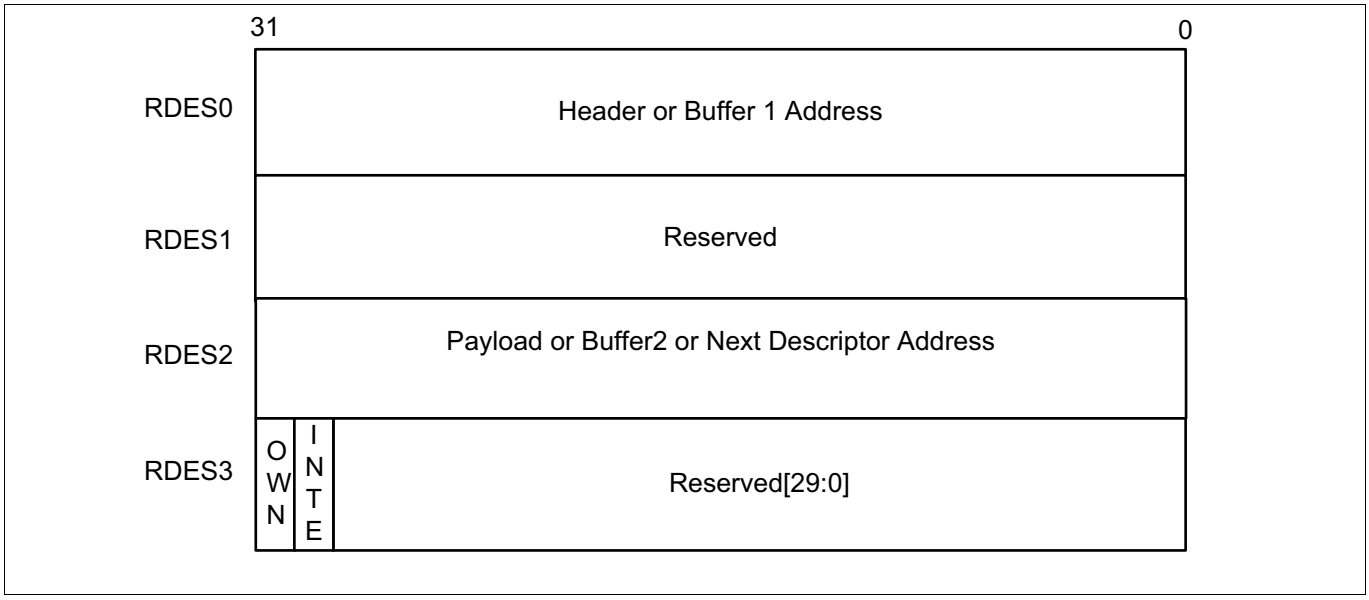


Figure 717 Receive Normal Descriptor Read Format

Note: In the Receive Descriptor (Read Format), if the Buffer Address field is all 0s, DWC_ether_qos does not transfer data to that buffer and skips to the next buffer or next descriptor.

Table 488 explains the read format of the RDES0 Normal Descriptor.

Table 488 RDES0 Normal Descriptor (Read Format)

Bit	Name	Description
31:0	BUF1AP	Header or Buffer 1 Address Pointer When the SPH bit of Control register of a channel is reset, these bits indicate the physical address of Buffer 1. When the SPH bit is set, these bits indicate the physical address of Header Buffer where the Rx DMA writes the L2/L3/L4 header bytes of the received packet.The application can program a byte-aligned address for this buffer which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a Write operation with RDES0[1:0] (or RDES0[2:0]/[3:0] in case of 64-/128-bit configuration) as zero. However, the packet data is shifted as per actual offset as given by buffer address pointer.If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

Table 489 describes the read format of the RDES1 Normal Descriptor.

Table 489 RDES1 Normal Descriptor (Read Format)

31:16	15	14	13	12	11:8	7	6	5	4	3	2:0
OPC	TD	TSA	PV	PFT	PMT	IPCE	IPCB	IPV6	IPV4	IPHE	PT

Table 490 provides the read format of the RDES2 Normal Descriptor.

Table 490 RDES2 Normal Descriptor (Read Format)

Bit	Name	Description
31:0	BUF2AP	Buffer 2 Address Pointer These bits indicate the physical address of Buffer 2.When the SPH bit of the DMA_CH#_Control register is set, the buffer address pointer must be bus width-aligned, that is, RDES2[3:0, 2:0, or 1:0] = 0 corresponding to 128, 64, or 32 bus width. LSBs are ignored internally. When the SPH bit of the DMA_CH#_Control register is reset, there is no limitations on the RDES2 value. However, the RxDMA uses the LS Bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores BUF2AP[3:0 or 2:0 or 1:0] (corresponding to 128- or 64- or 32-bit data-bus) and writes to the complete location.

Table 491 describes the read format of the RDES3 Normal Descriptor.

31	30	19-26	25	24	23:0
OWN	IOC	Rsvd	BUF2V	BUF1V	Rsvd

Table 491 RDES3 Normal Descriptor (Read Format)

Bit	Name	Description
31	OWN	Own Bit When this bit is set, it indicates that the DWC_ether_qos DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full
30	IOC	Interrupt Enabled on Completion When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29:26	Rsvd	Reserved
25	BUF2V	Buffer 2 Address Valid When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid.The application must set this bit so that the DMA can use the address, to which the Buffer 2 address in RDES2 is pointing, to write received packet data.
24	BUF1V	Buffer 1 Address Valid When set, this indicates to the DMA that the buffer 1 address specified in RDES1 is valid.The application must set this value if the address pointed to by Buffer 1 address in RDES1 canbe used by the DMA to write received packet data.
23:0	Rsvd	Reserved

44.3.17.6.2 Receive Normal Descriptor (Write-Back Format)

Figure 718 illustrates the write-back format for a Receive Normal descriptor.

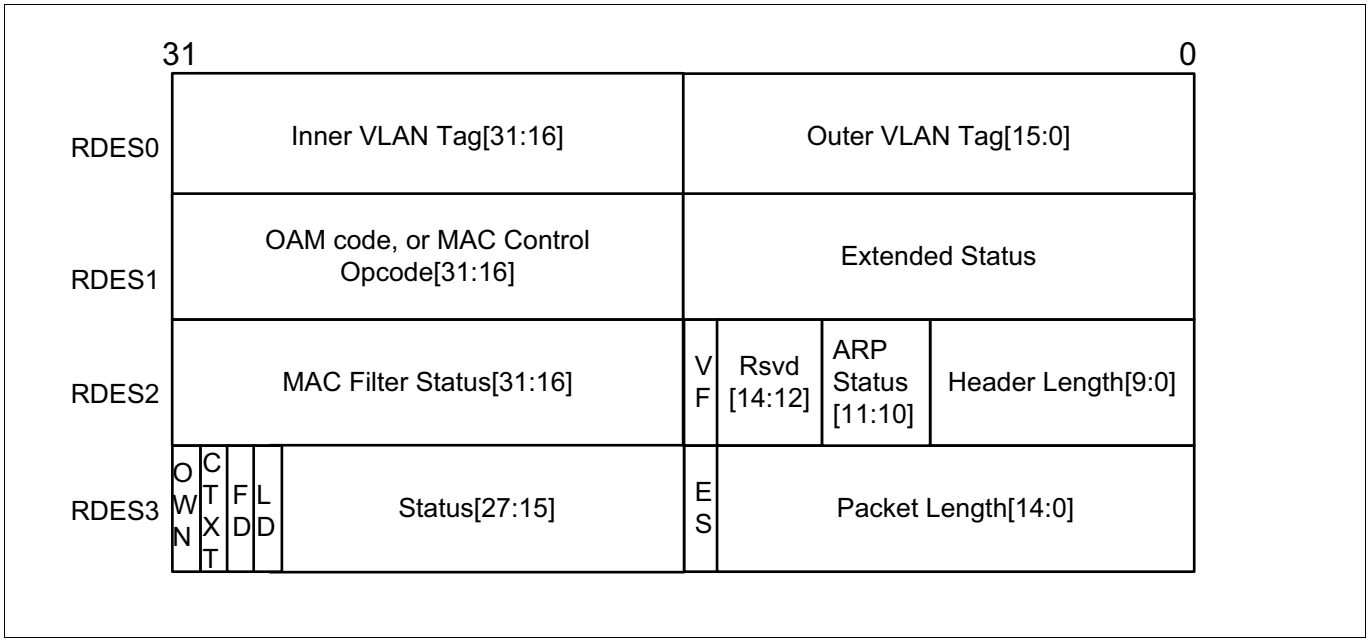


Figure 718 Receive Normal Descriptor (Write-Back Format)

Table 492 describes the write-back format for the RDES0 Normal Descriptor.

Table 492 RDES0 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31:16	IVT	Inner VLAN TagThis field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set.This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.
15:0	OVT	Outer VLAN TagThis field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

The Status fields in write-back format are valid only for the last descriptor (RDES3[28] is set). **Table 493** provides the details of the write-back format for RDES1 Normal Descriptor.

31:16	15	14	13	12	11:8	7	6	5	4	3	2:0
OPC	TD	TSA	PV	PFT	PMT	IPCE	IPCB	IPV6	IPV4	IPHE	PT

Table 493 RDES1 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31:16	OPC	OAM Sub-Type Code, or MAC Control Packet opcode OAM Sub-Type CodeIf Bits[18:16] of RDES3 are set to 3'b111, this field contains the OAM sub-type and code fields. MAC Control Packet opcode If Bits[18:16] of RDES3 are set to 3'b110, this field contains the MAC Control packet opcode field.
15	TD	Timestamp Dropped This bit indicates that the timestamp was captured for this packet but it got dropped in the MTL Rx FIFO because of overflow. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.
14	TSA	Timestamp Available When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
13	PV	PTP Version This bit indicates that the received PTP message has the IEEE 1588 version 2 format. When this bit is reset, it indicates the IEEE 1588 version 1 format. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.
12	PFT	PTP Packet Type This bit indicates that the PTP message is sent directly over Ethernet. This bit is available only when you select the Timestamp feature. Otherwise, this bit is reserved.
11:8	PMT	PTP Message Type These bits are encoded to give the type of the message received: • 0000: No PTP message received • 0001: SYNC (all clock types) • 0010: Follow_Up (all clock types) • 0011: Delay_Req (all clock types) • 0100: Delay_Resp (all clock types) • 0101: Pdelay_Req (in peer-to-peer transparent clock) • 0110: Pdelay_Resp (in peer-to-peer transparent clock) • 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) • 1000: Announce • 1001: Management • 1010: Signaling • 1011–1110: Reserved • 1111: PTP packet with Reserved message type These bits are available only when you select the Timestamp feature.

Table 493 RDES1 Normal Descriptor (Write-Back Format) (cont'd)

Bit	Name	Description
7	IPCE	IP Payload Error When this bit is set, it indicates either of the following: • The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. • The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. • The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.
6	IPCB	IP Checksum Bypassed This bit indicates that the checksum offload engine is bypassed. This bit is available when you select the Enable Receive TCP/IP Checksum Check feature.
5	IPV6	IPv6 header Present This bit indicates that an IPV6 header is detected. When the Enable Split Header Feature option is selected and the SPH bit of Control Register of a channel is set, the IPV6 header is available in the header buffer area to which RDES0 is pointing.
4	IPV4	IPV4 Header Present This bit indicates that an IPV4 header is detected. When the SPH bit of RDES3 is set, the IPV4 header is available in the header buffer area to which RDES0 is pointing.
3	IPHE	IP Header Error When this bit is set, it indicates either of the following: • The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. • The IP datagram version is not consistent with the Ethernet Type value. • Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either Bit 5 or Bit 4 is set. This bit is available when you select the Enable Receive TCP/IP Checksum Check feature.
2:0	PT	Payload Type These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): • 3'b000: Unknown type or IP/AV payload not processed • 3'b001: UDP • 3'b010: TCP • 3'b011: ICMP • 3'b110: AV Tagged Data Packet • 3'b111: AV Tagged Control Packet • 3'b101: AV Untagged Control Packet • 3'b100: IGMP if IPV4 Header Present bit is set else DCB (LLDP) Control Packet If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000.

31:27	26:19	18	17	16	15	14	13:11	10	9:0
Rsvd	MADRM	HF	DAF	SAF	OTS	ITS	Rsvd	ARPNR	HL

Table 494 RDES2 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31:27	Rsvd	Reserved
26:19	MADRM	MAC Address Match or Hash Value When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the hash value computed by the MAC. A packet passes the hash filter when the bit corresponding to the hash value is set in the hash filter register.
18	HF	Hash Filter Status When this bit is set, it indicates that the packet passed the MAC address hash filter. Bits[26:19] indicate the hash value.
17	DAF	Destination Address Filter Fail When this bit is set, it indicates that the packet failed the DA Filter in the MAC.
16	SAF	SA Address Filter Fail When this bit is set, it indicates that the packet failed the SA Filter in the MAC.
15	OTS	VLAN Filter Status When set, this bit indicates that the VLAN Tag of the received packed passed the VLAN filter.This bit is valid only when DWC_EQOS_ERVFE is not enabled.If DWC_EQOS_ERVFE is enabled, the bit is redefined as Outer VLAN Tag Filter Status (OTS). This bit is valid for both Single and Double VLAN Tagged frames
14	ITS	Inner VLAN Tag Filter Status (ITS) This bit is valid only when DWC_EQOS_ERVFE is enabled. This bit is valid only for Double VLAN Tagged frames, when Double VLAN Processing is enabled. For more information, see the Filter Status topic.
13:11	Rsvd	Reserved
10	ARPNR	ARP Reply Not Generated When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time). This bit is reserved when the Enable IPv4 ARP Offload option is not selected.
9:0	HL	L3/L4 Header Length This field contains the length of the header of the packet split by the MAC at L3 or L4 header boundary as identified by the MAC receiver. This field is valid only when the first descriptor bit is set (FD = 1). The header data is written to the Buffer 1 address of corresponding descriptor. If header length is zero, this field is not valid. It implies that the MAC did not identify and split the header. This field is valid when the Enable Split Header Feature option is selected.

Table 495 describes the write-back format for the RDES3 Normal Descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18:16	15	14:0
OWN	CTXT	FD	LD	RS2V	RS1V	RS0V	CE	GP	RWT	OE	RE	DE	LT	ES	PL

Table 495 RDES3 Normal Descriptor (Write-Back Format)

Bit	Name	Description
31	OWN	Own Bit When this bit is set, it indicates that the DWC_ether_qos DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full
30	CTXT	Receive Context Descriptor When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 1'b0 to this bit for normal receive descriptor. When CTXT and FD bits are used together, {CTXT, FD} • 00: Intermediate Descriptor • 01: First Descriptor • 10: Reserved • 11: Descriptor Error (due to all 1s) *Note: When Descriptor Error occurs, the Receive DMA closes the receive descriptor indicating Descriptor Error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data Receive DMA will set the CDE bit in DMA_CH#_Status register but not the RI bit even when IOC is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.*
29	FD	First Descriptor When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. See the CTXT bit description for details of using the CTXT bit and FD bit together.
28	LD	Last Descriptor When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
27	RS2V	Receive Status RDES2 Valid When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	Receive Status RDES1 Valid When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	Receive Status RDES0 Valid When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.

Table 495 RDES3 Normal Descriptor (Write-Back Format) (cont'd)

Bit	Name	Description
24	CE	CRC Error When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
23	GP	Giant Packet When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set).Note: Giant packet indicates only the packet length. It does not cause any packet truncation.
22	RWT	Receive Watchdog Timeout When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
21	OE	Overflow Error When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO.Note: This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.
20	RE	Receive Error When this bit is set, it indicates that the gmii_rxer_i signal is asserted while the gmii_rxdv_i signal is asserted during packet reception. This error also includes carrier extension error in the GMII and half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
19	DE	Dribble Bit Error When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
18:16	LT	Length/Type Field This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: • 3'b000: The packet is a length packet • 3'b001: The packet is a type packet. • 3'b011: The packet is a ARP Request packet type • 3'b100: The packet is a type packet with VLAN Tag • 3'b101: The packet is a type packet with Double VLAN Tag • 3'b110: The packet is a MAC Control packet type • 3'b111: The packet is a OAM packet type • 3'b010: Reserved

Table 495 RDES3 Normal Descriptor (Write-Back Format) (cont'd)

Bit	Name	Description
15	ES	Error Summary When this bit is set, it indicates the logical OR of the following bits: • RDES3[24]: CRC Error • RDES3[19]: Dribble Error • RDES3[20]: Receive Error • RDES3[22]: Watchdog Timeout • RDES3[21]: Overflow Error • RDES3[23]: Giant Packet This field is valid only when the LD bit of RDES3 is set.
14:0	PL	Packet Length These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when the LD bit of RDES3 is set and Overflow Error bits are reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. This field is valid when the LD bit of RDES3 is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current packet.

44.3.17.6.3 Receive Context Descriptor

The DMA can write to Receive Context Descriptor, which provices extended status related to he last receive packet. The descriptor is read only for the application.

This descriptor is read-only for the application. Only the DMA can write to this descriptor. The context descriptor provides information about the extended status related to the last received packet. The Bit 30 of RDES3 indicates the context type descriptor.

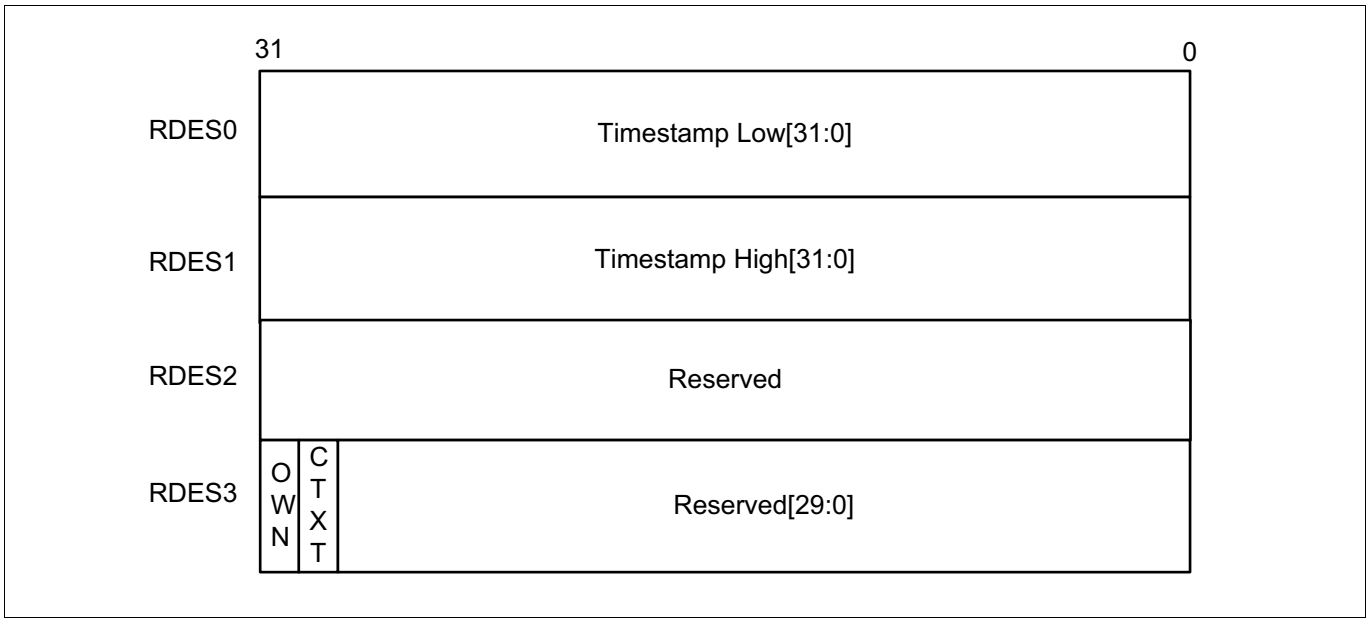


Figure 719 Receive Context Descriptor Format

Table 496 RDES0 Context Descriptor

Bit	Name	Description
31:0	RTSL	Receive Packet Timestamp Low The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.

Table 497 RDES1 Context Descriptor

Bit	Field	Description
31:0	RTSH	Receive Packet Timestamp High The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.

Table 498 RDES2 Context Descriptor

Bit	Description
31:0	Reserved

Table 499 RDES3 Context Descriptor

Bit	Name	Description
31	OWN	Own Bit When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: • The DMA completes the packet reception • The buffers associated with the descriptor are full
30	CTXT	Receive Context Descriptor When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 1'b1 to this bit for context descriptor. DMA writes 2’b11 to indicate a descriptor error due to all 1s. When CTXT and FD bits are used together, {CTXT, FD} • 00: Reserved • 01: Reserved • 10: Context Descriptor • 11: Descriptor Error *Note: When Descriptor Error occurs, the Receive DMA closes the receive descriptor indicating Descriptor Error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data Receive DMA will set the CDE bit in DMA_CH#_Status register but not the RI bit even when IOC is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.*

Table 499 RDES3 Context Descriptor (cont'd)

Bit	Name	Description
29	DE	Descriptor Error See the CTXT bit description for details of using theDE bit along with CTXT bit.
28:0	Rsvd	Reserved

44.3.18 Programming Sequences

Depending on the DWC_ether_qos configuration, the controller must be initialized in the proper sequence. Additionally, it is recommended that you follow programming guidelines for the supported features.

The DWC_ether_qos is programmed through the software registers.

Note: After a write operation, there should not be any further writes to the same location until the first write is updated. Otherwise, the second write operation will not get updated to the destination clock domain. Thus the delay between two writes to the same register location should be at least 6 cycles of the destination clock (PHY receive clock or PHY transmit clock).

44.3.18.1 Initializing DMA

You must perform a number of steps to initialize the DMA.

Complete the following steps to initialize the DMA:

Steps

1. Provide a software reset. This resets all of the MAC internal registers and logic (bit-0 of DMA_Mode).
2. Wait for the completion of the reset process (poll bit 0 of the DMA_Mode, which is only cleared after the reset operation is completed).
3. Program the following fields to initialize the DMA_SysBus_Mode register:
   a) AAL
   b) Fixed burst or undefined burst
   c) Burst mode values in case of AHB bus interface, OSR_LMT in case of AXI bus interface.
   d) If fixed length value is enabled, select the maximum burst length possible on the AXI Bus (bits [7:1])
4. Create a descriptor list for transmit and receive. In addition, ensure that the receive descriptors are owned by DMA (set bit 31 of descriptor TDES3/RDES3).
5. Note: The descriptor address from the start to the end of the ring must not cross the 4GB boundary.
   Program the Transmit and Receive Ring length registers (DMA_CH(#i)_TxDesc_Ring_Length (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) and DMA_CH(#i)_RxDesc_Ring_Length (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)). The ring length programmed must be at least 4.
6. Program the Transmit and Receive Ring length registers (DMA_CH(#i)_TxDesc_Ring_Length (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) and DMA_CH(#i)_RxDesc_Ring_Length (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)). The ring length programmed must be at least 4.
7. Note: For 40- or 48-bit addressing mode, program the higher address List registers (DMA_CH[n]_TxDesc_List_HAddress, DMA_CH[n]_RxDesc_List_HAddress).
   Program the settings of the following registers for the parameters like maximum burst-length (PBL) initiated by DMA, descriptor skip lengths, OSP in case of TxDMA, RBSZ in case of RxDMA, and so on:
   - DMA_CH(#i)_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1)
   - DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1)
   - DMA_CH(#i)_RX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)
8. Enable the interrupts by programming the DMA_CH(#i)_Interrupt_Enable (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register.
9. Start the Receive and Transmit DMAs by setting SR (bit 0) of the DMA_CH(#i)_RX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1) and ST (bit 0) of the DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register.
10. Repeat steps 4 to 9 for all the Tx DMA and Rx DMA channels selected in the hardware.

44.3.18.2 Initializing MTL Registers

The Transaction Layer (MTL) registers must be initialized to establish the transmit and receive operating modes and commands.

Complete the following steps to initialize the MTL registers:

Steps

1. Program the Tx Scheduling (SCHALG) and Receive Arbitration Algorithm (RAA) fields in MTL_Operation_Mode to initialize the MTL operation in case of multiple Tx and Rx queues.
2. Program the Receive Queue to DMA mapping in MTL_RxQ_DMA_Map0 and MTL_RxQ_DMA_Map1 registers.
3. Program the following fields to initialize the mode of operation in the MTL_TxQ0_Operation_Mode register.
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) in case of threshold mode
   b) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queue0
   c) Transmit Queue Size (TQS)
4. Program the following fields to initialize the mode of operation in the MTL_RxQ0_Operation_Mode register:
   a) Receive Store and Forward (RSF) or RTC in case of Threshold mode
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD)
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP)
   d) Receive Queue Size (RQS)
5. Repeat previous two steps for all MTL Tx and Rx queues selected in the configuration.

44.3.18.3 Initializing MAC

The MAC configuration registers establish the operating mode of the MAC. These registers must be initialized before initializing the DMA.

The following MAC Initialization operations can be performed after DMA initialization. If the MAC initialization is completed before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

Steps

1. Provide the MAC address registers: MAC_Address0_High and MAC_Address0_Low. If more than one MAC address is enabled in your configuration (during configuration in the coreConsultant), program the MAC addresses appropriately.
2. Program the following fields to set the appropriate filters for the incoming frames in the MAC_Packet_Filter register:
   a) Receive All
   b) Promiscuous mode
   c) Hash or Perfect Filter
   d) Unicast, multicast, broadcast, and control frames filter settings
3. Program the following fields for proper flow control in the MAC_Q0_Tx_Flow_Ctrl register:
   a) Pause time and other Pause frame control bits
   b) Transmit Flow control bits
   c) Flow Control Busy
4. Program the MAC_Interrupt_Enable register, as required, and if applicable, for your configuration.

5. Program the appropriate fields in the MAC_Configuration register. For ex: Inter-packet gap while transmission and jabber disable.
6. Set bit 0 and 1 in MAC_Configuration registers to start the MAC transmitter and receiver.

44.3.18.4 Performing Normal Receive and Transmit Operation

During normal operation of the DWC_ether_qos, normal and transmit interrupts are read, descriptors polled, the DMA is suspended (if it does not own descriptors), and values of current host transmitter or receiver descriptor pointers are read for debugging.

For normal operation, complete the following steps:

Steps

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptors, reading the status of the descriptor owned by the Host (either transmit or receive).
2. Set appropriate values for the descriptors, ensuring that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into SUSPEND state. The transmission or reception can be resumed by freeing the descriptors and writing the descriptor tail pointer to Tx/Rx tail pointer register (DMA_CH[n]_TxDesc_Tail_Pointer and DMA_CH[n]_RxDesc_Tail_Pointer).
4. The values of the current host transmitter or receiver descriptor address pointer can be read for the debug process (DMA_CH[n]_Current_App_TxDesc and DMA_CH[n]_Current_App_RxDesc).
5. The values of the current host transmit buffer address pointer and receive buffer address pointer can be read for the debug process (Register DMA_CH[n]_Current_App_TxBuffer and DMA_CH[n]_Current_App_RxBuffer).

44.3.18.5 Stopping and Starting Transmission

You can pause transmission by disabling the Transmit DMA, waiting for previous frame transmissions to complete, disabling the MAC transmitter and receiver, and disabling the Receive DMA.

Notes

1. *Do not change the configuration (such as duplex mode, speed, port, or loop back) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active.*
2. *Similarly, do not change the DMA-related configuration when Transmit and Receive DMA are active.*

Complete the following steps to pause the transmission for some time. The steps are provided for Channel 0.

Steps

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) Register.
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of MTL_TxQ0_Debug Register (TRCSTS is not 01 and TXQSTS=0).
3. Disable the MAC transmitter and MAC receiver by clearing Bit (RE) and Bit 1(TE) of the MAC_Configuration Register.
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of MTL_RxQ0_Debug Register, PRXQ=0 and RXQSTS=00).

5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in MTL_TxQ0_Debug Register and RXQSTS is 0 in MTL_RxQ0_Debug Register).
6. To restart the operation, first start the DMAs, and then enable the MAC Transmitter and Receiver.

44.3.18.6 Programming Guidelines for Switching to New Descriptor List in RxDMA

If you want to switch to a new descriptor list is allowed when the RxDMA is in the SUSPEND state, however, make sure you understand the guidelines before the switch.

Switching to a new descriptor list is different in the Rx DMA compared to the Tx DMA. Switching to a new descriptor list is permitted when the RxDMA is in SUSPEND state, as clarified by the following points:

- Generally, RxDMA prepares the descriptors in advance.
- If the RxDMA goes to SUSPEND due to descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames will be lost because of an RxFIFO Overflow. Therefore, the software is allowed to create a new descriptor list and program the RxDMA to start using it immediately, without going into STOP state.

44.3.18.7 Programming Guidelines for Multi-Channel, Multi-Queuing

44.3.18.7.1 Programming Guidelines for Multi-Channel Multi-Queuing - Transmit

You can program several aspects of a Transmit queue, such as specifying queue size, enabling a queue, and programming the scheduling method.

Complete the following steps:

Steps

1. Program the Transmit queue size in the TQS field of MTL_TxQ[n]_Operation_Mode register. Based on the value programmed in the TQS field, the size of the queue is determined.
2. For a queue to be used, the queue needs to be enabled in TXQEN in the corresponding MTL_TxQ[n]_Operation_Mode Register. In DMA configurations, the ST bit of DMA_CH[n]_Tx_Control Register and corresponding TXQEN in MTL_TxQ[n]_Operation Mode Register needs to be enabled.
3. The scheduling method needs to be programmed in SCHALG of MTL_Operation_Mode register.
4. Program the MTL_TxQ[n]_Quantum_Weight for generic or DCB queue as per the selected algorithm. In case of CBS algorithm in AVB queues, the MTL_TxQ[n]_ETS_Control, MTL_TxQ[n]_SendSlopeCredit, MTL_TxQ[n]_HiCredit and MTL_TxQ[n]_LoCredit registers also need to be programmed as required.
5. If DCB is enabled and PFC function is required, program MAC_TxQ_Prty_Map0 Register to assign a fixed priority to the queue. This priority assigned is used for determining if the corresponding queue should stop transmitting packet based on the received PFC packet.

44.3.18.7.2 Programming Guidelines for Multi-Channel Multi-Queuing - Receive

You can program several aspects of a Receive queue, such as specifying queue size, enabling a queue for AV or DCB, setting the packet types (so they can be routed to Receive queues), and defining arbitration and mapping if multiple DMA channels enabled.

Complete the following steps:

Steps

1. Program the Receive queue size in the RQS field of MTL_RxQ[n]_Operation_Mode Register. Based on the value programmed in RQS field, the size of the queue is determined.
2. Enable the Receive Queues 0 to 7 in the fields RXQ0EN to RXQ7EN in MAC_RxQ_Ctrl0 Register for AV or DCB. In DMA configurations, SR bit of statically or dynamically mapped DMA_CH[n]_Rx_Control Register and corresponding RXQ[n]_EN in MAC_RxQ_Ctrl0 Register needs to be enabled.
3. The MAC routes the Rx packets to the Rx Queues based on following packet types:
   a) AV PTP Packets: Based on the programming of AVPTPQ in MAC_RxQ_Ctrl1 Register.
   b) AV Untagged Control packets: Based on the programming of AVCPQ in MAC_RxQ_Ctrl1 Register.
   c) Data Center Bridging (DCB) related Link Layer Discovery Protocol (LLDP) packets. Program DCBCPQ in MAC_RxQ_Ctrl1 Register to indicate to MAC which queue should get the DCB packets.
   d) VLAN Tag Priority field in VLAN Tagged packets: Program PSRQ7-0 of the MAC_RxQ_Ctrl2 Register for the routing of tagged packets based on the USP (user Priority) field of the received packets to the Rx Queues 0 to 7.
   e) The AV tagged control and data packets are also routed based on PSRQ field of the MAC_RxQ_Ctrl2 register.
4. If multiple RX DMA channels are enabled, the following programming should be done for proper arbitration and mapping:
   a) Program the RAA field of MTL_Operation_Mode register to select the arbitration algorithm to decide which RxQ is read out from the RxFIFO memory.
   b) Program the MTL_RxQ[n]_Control to decide the weights and the packet arbitration for each RxQ.
   c) If static mapping is programmed in MTL_RxQ_DMA_Map[n] register (RXQ[n]DADMACH is reset to 0), bits RXQx2DMA and others need to be programmed to select the channel for which each queue is mapped.
   d) Set RXQ[n]DADMACH bit in MTL_RxQ_DMA_Map0 Register to select dynamic mapping of packets in each RxQueue.
   e) In dynamic channel mapping, the routing of a packet to a specific RxDMA channel is decided by the value of DCS field in the lowest MAC Address Register.
   Note: The priorities set in PSRQ7-0 should be unique.

44.3.18.8 Programming Guidelines for GMII Link State Transitions

There are two programming tasks with regards to GMII Link State Transitions when transmit and receive clocks are either running or stopped when the Link is down.

When the Link is down, there are two types of scenarios where you must perform programming steps:

- When transmit and receive clocks are running
- When transmit and receive clocks are stopped

44.3.18.8.1 Transmit and Receive Clocks Running when Link Down

When the Link is down but the Transmit and Receive clocks are still running, you need to need to disable the Transmit DMA and the MAC receiver, wait for previous frame transmissions to complete, disable the MAC transmitter, verify both Tx and Rx queues are empty, read PHY registers to determine latest configuration and then program MAC registers, and restart the Tx DMA, and then enable the MAC Transmit and Receiver.

The following steps explains how to program DWC_ether_qos when the link is down but the Transmit and Receive clocks are running:

Note: The steps are provided for Channel 0.

Steps

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) Register.
2. Disable the MAC receiver by clearing Bit 2 (RE) of MAC_Configuration Register.
3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of MTL_TxQ0_Debug Register (TRCSTS is not 01).Or, Flush the Tx FIFO for faster empty operation.
4. Disable the MAC transmitter by clearing Bit 1(TE) of the MAC_Configuration Register.
5. Make sure that both Tx Queue and Rx Queue are empty (TXQSTS is 0 in MTL_TxQ0_Debug Register and RXQSTS is 0 in MTL_RxQ0_Debug Register).
6. After the link is up, read the PHY registers to know the latest configuration and accordingly program the MAC registers.
7. Restart the operation by starting the Tx DMA, and then enabling the MAC Transmitter and Receiver.

44.3.18.8.2 Transmit and Receive Clocks Stopped when Link Down

When the Link is down and the clocks are stopped, you need to disable the MAC Transmitter and Receiver, wait until clocks are restored, wait for completion of transfer (in progress when clocks stopped), read PHY registers to determine latest operation mode and program the MAC registers, and then restart MAC Transmitter and Receiver.

Complete the following steps when the link is down and the Transmit and Receive clocks are stopped:

Note: The steps are provided for Channel 0.

Steps

1. Disable the MAC Transmitter and Receiver by clearing bits RE and TE of MAC_Configuration Register. This does not take effect immediately as the clocks are absent.
2. Wait until the link is up and the clocks are restored.
3. Wait for the completion of the transfer of any partial frame, if any, at time of stopping of Transmit/Receive clock. This can be checked by reading the MAC_Debug Register (should be all-zero). Some old packets may still remain in the TXFIFO as the MAC Transmitter is stopped.
4. Read the PHY registers to know the latest operating mode and accordingly program the MAC registers.
5. Restart the MAC Transmitter and Receiver by setting RE and TE bits.

44.3.18.9 Programming Guidelines for IEEE 1588 Timestamping

There are several programming tasks that can be performed after you enable the IEEE 1588 Timestamping feature: initializing timestamp counter, synchronizing or updating system time in one process, and synchronizing or updating system time to reduce system-time jitter.

There are three programming tasks that can be performed after you have enabled the IEEE 1588 Timestamping feature:

- Initializing the timestamp counter
- Synchronizing or updating the system time in one process (coarse correction method)
- Synchronizing or updating the system time to reduce system-time jitter (fine correction method)

44.3.18.9.1 Initialization Guidelines for System Time Generation

During the initialization of the DWC_ether_qos controller, you can enable the timestamp feature.

You can enable the timestamp feature by setting Bit 0 of the MAC_Timestamp_Control Register. However, it is essential that the timestamp counter be initialized after this bit is set. Complete the following steps during initialization:

Steps

1. Mask the Timestamp Trigger interrupt by clearing the bit 16 of MAC_Interrupt_Enable Register.
2. Set Bit 0 of MAC_Timestamp_Control Register to enable timestamping.
3. Program MAC_Sub_Second_Increment Register based on the PTP clock frequency.
4. If you are using the Fine Correction approach, program MAC_Timestamp_Addend and set Bit 5 of MAC_Timestamp_Control Register.
5. Poll the MAC_Timestamp_Control Register until Bit 5 is cleared.
6. Program Bit 1 of MAC_Timestamp_Control Register to select the Fine Update method (if required).
7. Program MAC_System_Time_Seconds_Update Register and MAC_System_Time_Nanoseconds_Update Register with the appropriate time value.
8. Set Bit 2 in MAC_Timestamp_Control Register.
   The timestamp counter starts operation as soon as it is initialized with the value written in the Timestamp Update registers.
   If one-step timestamping is enabled
   a) To enable one-step timestamping, program Bit 27 of the TDES3 Context Descriptor.
   b) Program registers MAC_Timestamp_Ingress_Asym_Corr and MAC_Timestamp_Egress_Asym_Corr to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper timestamping.

Example

Note: If timestamp operation is disabled by clearing Bit 0 of MAC_Timestamp_Control Register, you need to repeat all these steps to restart the timestamp operation.

44.3.18.9.2 Updating System Time in One Process

You can synchronize or updated the system time in one process by programmiing the Timestamp Update registers.

To synchronize or update the system time in one process (coarse correction method), complete the following steps:

Steps

1. Set the offset (positive or negative) in the Timestamp Update registers (MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update).
2. Set Bit 3 (TSUPDT) of the MAC_Timestamp_Control Register.
   The value in the Timestamp Update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

44.3.18.9.3 Updating System Time to Reduce System-Time Jitter

You can program the DWC_ether_qos controller to synchronize or update the system time to reduce system-time jitter, which is known as fine correction method).

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

Steps

1. With the help of the algorithm explained in the “System Time Register Module” section, calculate the rate by which you want to make the system time increments slower or faster.
2. Update the MAC_Timestamp_Addend with the new value and set Bit 5 of the MAC_Timestamp_Control Register.
3. Wait for the time for which you want the new value of the Addend register to be active. You can do this by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in MAC_PPS[n]_Target_Time_Seconds Register and MAC_PPS[n]_Target_Time_Nanoseconds Register.
5. Enable the Timestamp interrupt in bit 12 of MAC_Interrupt_Enable register.
6. Set bit 4 in Register MAC_Timestamp_Control.
7. When this trigger causes an interrupt, read MAC_Interrupt_Status Register.
8. Reprogram MAC_Timestamp_Addend Register with the old value and set bit 5 again.

44.3.18.10Programming Guidelines for AV Feature

There are several programming tasks you can perform after you enable the AV Feature: initializing the DMA (QOS-AHB configuration only), enabling slot number checking, enabling average bits per slot reporting, and disabling flow control for AV-enabld queues.

After you enable the AV feature in the DWC_ether_qos controller, there are several programming tasks to be followed:

- Initialzing the DMA for QOS-AHB configurations only
- Enabling slot number checking
- Enabling average bits per slot reporting
- Disabling flow control for AV-enabled queues (transmit and receive flow control)

44.3.18.10.1 Initializing the DMA

The first step in programming the AV feature in a QOS-AHB configuration is to initialize the DMA.

Use this initialization sequence only for QOS-AHB configuration with AV feature.

Steps

1. Provide a software reset to reset all QOS internal registers and logic (bit 0 in DMA_Mode register).
2. Wait for the completion of the reset process. Poll bit 0 of the DMA_Mode register), which is cleared only after the reset operation is completed.
3. Program the fields to initialize the DMA register by setting the values in DMA_Mode register.
4. Create a proper descriptor list for transmit and receive. In addition, ensure that the DMA owns the Transmit and Receive descriptors. When OSF mode is used, at least two TX descriptors are required.
5. Make sure that your software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors.
6. Program the Transmit and Receive Ring length registers (DMA_CH(#i)_TxDesc_Ring_Length (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) and DMA_CH(#i)_RxDesc_Ring_Length (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)). The ring length programmed must be at least 4.

7. Initialize receive and transmit descriptor list address with the base address of the transmit and receive descriptor (DMA_CH(#i)_TxDesc_List_Address (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1), DMA_CH(#i)_TxDesc_List_Address (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1)). In addition, you must program the Transmit and Receive tail pointer registers indicating to the DMA about the available descriptors (DMA_CH(#i)_TxDesc_Tail_Pointer (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) and DMA_CH(#i)_RxDesc_Tail_Pointer (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1)).
8. Program the following fields to initialize the mode of operation in the MTL_TxQ0_Operation_Mode register:
   a) Transmit Store And Forward (TSF)
   b) Transmit Threshold Control (TTC)
   c) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queu
   d) Transmit Queue Size (TQS)
9. Enable the interrupts by programming the DMA_CH(#i)_Interrupt_Enable (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register.
10. Repeat steps 4 through 9 for all additional channels of AV feature.
11. Program the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV Queues.
12. Start the Receive and Transmit DMA by setting bit 0 of the DMA_CH(#i)_TX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1) register and bit 0 of the DMA_CH(#i)_RX_Control (for i = 0; i <= DWC_EQOS_NUM_DMA_RX_CH-1) register.

44.3.18.10.2 Enabling Slot Number Checking

You can enable slot number checking feature if you want to specify the intervals at which the DMA channels mapped to AV Queues fetch the frames from the AHB/AXI system bus.

Before you begin

These steps should be completed after step 11 and before step 12 of “Initializing DMA”.

You can use the slot number check feature to specify the intervals at which the DMA Channels mapped to AV Queues fetch the frames from the AHB/AXI system bus. This feature is useful for a uniform and periodic transfer of the AV traffic from the host memory. The feature is available only when you enable timestamping and program the MAC_Sub_Second_Increment register. Complete the following steps to enable the slot number checking:

Steps

1. Enable timestamping by following the steps described in “Initialization Guideline for System Time Generation”.
2. Make sure that the SLOTNUM field (bits 22:19) of TDES3 Normal Descriptor (Read Format) contains a valid slot number. You can read the current reference slot number from the DMA_CH(#i)_Slot_Function_Control_Status (for i = 0; i <= DWC_EQOS_NUM_DMA_TX_CH-1).
3. Set Bit 0 (ESC) of the Slot Function Control and Status register of a channel to enable the slot number checking.

44.3.18.10.3 Enabling Average Bits Per Slot Reporting

You can enable the ability to report the average bits that are transmitted in a slot.

The CBS Status register of the additional AV channels provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting:

Steps

1. Enable timestamping by following the steps described in the “Initialization Guideline for System Time Generation” section.
2. Program Bits [6:4], SLC, of the MTL_TxQ[n]_ETS_Control register of a channel with the number of slots over which the average transmitted bits per slot need to be computed.
3. Enable Bit 9(ABPSSIE) of the MTL_Q[n]_Interrupt_Control_Status register of a channel to generate the average bits per slot interrupt.

   Note: The frequency of this interrupt depends on the value programmed in step 2. For example, when you program value 0 in the SLC field, the interrupt is generated at every 125 microsecond.

   Note: When not required, you can disable this interrupt to stop the interrupt flooding.
4. Read Bits [16:0], ABS, from the MTL_TxQ[n]_ETS_Status register of a channel on each interrupt.

   Note: The software can read the ABS bits in polling mode even if the ABPSIE bit is not enabled. When high, bit 1 (ABPSIS) of the MTL_TxQ[n]_ETS_Status register indicates that a new value is updated in the ABS field.

44.3.18.10.4 Disabling Transmit Flow Control for AV Enabled Queues

You can disable the transmit flow control for the AV-enabled queue.

You can disable the transmit flow control for the AV-enabled queues, as follows:

- Program the EHFC (Enable Hardware Flow Control) bit of corresponding Rx Queue's MTL_RxQ[n]_Operation_Mode register to 0.

44.3.18.10.5 Disabling Receive Flow Control for AV Enabled Queues

You can disable the receive flow control for the AV-enabled queue.

You can disable the receive flow control for the AV-enabled queues, as follows:

- Program the PSTQ[n] field corresponding to AV enabled Tx Queue in MAC_TxQ_Prty_Map0/1 register to 0.

44.3.18.11Programming Guidelines for Energy Efficient Ethernet

There are several programming tasks you can perform for the Energy Efficient Ethernet (EEE) feature: entering and exiting Tx LPI mode, and gating off CSR clock in LPI mode.

After you enable the Energy Efficient Ethernet (EEE) in the DWC_ether_qos controller, you can program the following:

- Entering and Exiting Tx LPI mode during the QoS initialization
- Gating off the CSR clock in the Rx LPI mode
- Gating off the CSR clock in the Tx LPI mode

44.3.18.11.1 Entering and Exiting the Tx LPI Mode

Energy Efficient Ethernet (EEE) enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of Physical layers to operate in the Low-Power Idle (LPI) mode. In the Transmit path, the software must set the LPIEN bit of the MAC_LPI_Control_Status register to indicate to the MAC to stop transmission and initiate the LPI protocol. You can configure the Energy Efficient Ethernet (EEE) feature in coreConsultant while initializing the core.

Before you begin

Complete the following steps during QoS core initialization:

Steps

1. Read the PHY register through the MDIO interface, check if the remote end has the EEE capability, and then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode.)
3. Program Bits[25:16] and Bits[15:0] in MAC_LPI_Timers_Control Register.
4. Read the link status of the PHY chip by using the MDIO interface and update Bit 17 of MAC_LPI_Control_Status accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Program the MAC_1US_Tic_Counter as per the frequency of the clock used for accessing the CSR slave port.
6. Program the MAC_LPI_Entry_Timer register (LPIET) with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set LPITE and LPITXA (bit[20:19]) of MAC_LPI_Control_Status register to enable the auto-entry into LPI and auto-exit of MAC from LPI state.
8. Program the MAC_1US_Tic_Counter as per the frequency of the clock used for accessing the CSR slave port.
9. Program the MAC_LPI_Entry_Timer register (LPIET) with the IDLE time for which the MAC should wait before entering the LPI state on its own.
10. Set LPITE and LPITXA (bit[20:19]) of MAC_LPI_Control_Status register to enable the auto-entry into LPI and auto-exit of MAC from LPI state.
11. Set Bit 16 of MAC_LPI_Control_Status Register to make the MAC Transmitter enter the LPI state.
12. When a packet is scheduled for transmission (when the TxDMA comes out of IDLE state or when a packet is presented at ATI or MTI interface), the MAC Transmitter exits LPI state automatically. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.
13. MAC Transmitter re-enters LPI state if it remains IDLE for LPIET time and sets the TLPIEN bit and the entry-exit cycle continues.
14. Reset LPITXEN in case the application wants to over-ride the auto-entry/exit modes and make the MAC Transmitter exit the LPI state directly.

Example

Notes

1. *To make the MAC enter the LPI state only after it completes the transmission of all queued frames in the Tx FIFO, you should set Bit 19 in MAC_LPI_Control_Status Register.*
2. *To switch off the GMII Transmit Clock during the LPI state, use the sbd_tx_clk_gating_ctrl_o signal for gating the clock input.*
3. *To switch off the CSR clock or power to the rest of the system during the LPI state, you should wait for the TLPIEN interrupt of MAC_LPI_Control_Status Register to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.*

44.3.18.11.2 Gating Off the CSR Clock in the Rx LPI Mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode. In the Rx LPI mode, the MAC RX enters the LPI mode and the RX LPI interrupt is set. The interrupt pin is asserted and cleared when the host reads the MAC LPI Control Status register.

Before you begin

The following operations are performed when the MAC receives the LPI pattern from the PHY:

Steps

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status [RLPIEN interrupt of MAC_LPI_Control_Status Register] is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the MAC_LPI_Control_Status Register.

Example

After the sbd_intr_o interrupt is asserted and the MAC Tx is also in the LPI mode, you can gate-off the CSR clock. If the MAC TX is not in the LPI mode when you gate off the CSR clock, the events on the MAC transmitter do not get reported or updated in the CSR.

For restoring the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when MAC_LPI_Control_Status Register is read.

44.3.18.11.3 Gating Off the CSR Clock in the Tx LPI Mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode. To gate off the CSR Clock in the Tx LPI mode, the Transmit LPI Entry interrupt is set and the interrupt pin is asserted.

Before you begin

The following operations are performed when Bit 16 (LPIEN) of MAC_LPI_Control_Status is set:

Steps

1. The Transmit LPI Entry interrupt (TLPIEN bit of MAC_LPI_Control_Status Register) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the MAC_LPI_Control_Status Register.

Example

After the sbd_intr_o interrupt is asserted and the MAC RX is also in the LPI mode, you can gate off the CSR clock. If the MAC RX is not in the LPI mode when you gate off the CSR clock, the events on the MAC receiver do not get reported or updated in the CSR.

For restoring the CSR clock, switch on the CSR clock when the MAC has to come out of the TX LPI mode. After the CSR clock is resumed, reset Bit 16 (LPIEN) of MAC_LPI_Control_Status Register to bring the MAC out of the LPI mode.

44.3.18.12Programming Guidelines for Flexible Pulse-Per-Second Output

There are several programming tasks related to the Flexible Pulse-Per-Second (PPS) Output: generating single pulse on PPS, generating pulse train on PPS, and generating an interrupt without affecting PPS.

After you enable the Flexible Pulse-Per-Second Output feature in the DWC_ether_qos controller, you can perform the following tasks:

- Generating Single Pulse on PPS
- Generating Next Pulse on PPS, see **Note on Page 155**
- Generating a Pulse Train on PPS

- Generating an Interrupt without Affecting the PPS

44.3.18.12.1 Generating Single Pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time elapses. You can also program the behavior of the next pulse in advance. To generate a single pulse on PPS, program the MAC_PPS_Control and Target Time registers, and also program the start time value and the width of the PPS signal output. Finally, set the MAC_PPS_Control register to generte a single pulse on the PPS signal output.

Complete the following steps to generate single pulse on PPS:

Steps

1. Program 11 or 10 (for interrupt) in Bits [6:5], TRGTMODSEL, of MAC_PPS_Control Register. This instructs the MAC to use the Target Time registers (register MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds) for start time of PPS signal output.
2. Program the start time value in the Target Time registers (register MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds).
3. Program the width of the PPS signal output in MAC_PPS(#i)_Width (for i = 0; i <= DWC_EQOS_PPS_OUT_NUM-1) Register.
4. Program Bits [3:0], PPSCMD, of MAC_PPS_Control to 0001. This instructs the MAC to generate single pulse on the PPS signal output at the time programmed in the Target Time registers.

44.3.18.12.2 Generating Next Pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time elapses. You can also program the behavior of the next pulse in advance.

To program the next pulse, following is the procedure:

Steps

1. Program the start time for the next pulse in the Target Time registers. This time should be more than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in MAC_PPS(#i)_Width (for i = 0; i <= DWC_EQOS_PPS_OUT_NUM-1).
3. Program Bits [3:0], PPSCMD, of MAC_PPS_Control to generate a single pulse after the time at which the previous pulse is de-asserted. This instructs the MAC to generate single pulse on the PPS signal output, at the time programmed in Target Time registers.
   If you give this command before the previous pulse becomes low, then the new command overwrites theprevious command and the QOS may generate only 1 extended pulse.

44.3.18.12.3 Generating a Pulse Train on PPS

The DWC_ether_qos provides the flexibility to program the start or stop time, width, and interval of the pulse generated on the ptp_pps_o output. To generate a pulse train on the pulse per second (PPS), program the MAC_PPS_Contro and Target Time l register, set the interval between the train of pulses on the PPS signal output. Further, program the width of the PPS signal output, set the MAC_PPS_Control register and program the stop value in the Target Time register. Finally, program the stop value in the Target Time registers.

Complete the following steps to generate a pulse train on PPS:

Steps

1. Program 11 or 10 (for interrupt) in Bits [6:5], TRGTMODSEL, of MAC_PPS_Control Register. This instructs the MAC to use the Target Time registers (register 736 and 737) for start time of the PPS signal output.
2. Program the start time value in the Target Time registers (register 736 and 737).
3. Program the interval value between the train of pulses on the PPS signal output inMAC_PPS(#i)_Interval (for i = 0; i <= DWC_EQOS_PPS_OUT_NUM-1) Register.
4. Program the width of the PPS signal output in MAC_PPS(#i)_Width (for i = 0; i <= DWC_EQOS_PPS_OUT_NUM-1) Register.
5. Program Bits[3:0], PPSCMD, of MAC_PPS_Control Register to 0010. This instructs the MAC to generate train of pulses on the PPS signal output with start time programmed in Target Time registers.
   By default, the PPS pulse train is free-running unless stopped by 'STOP Pulse train at time' or 'STOP Pulse Train immediately' commands.
6. Program the stop value in the Target Time registers. Ensure that Bit 31 (TSTRBUSY) of MAC_PPS(#i)_Target_Time_Nanoseconds (for i = 0; i <= DWC_EQOS_PPS_OUT_NUM-1) Register is reset before programming the Target Time registers again.
7. Program the PPSCMD field (bit 3:0) of MAC_PPS_Control to 0100. This stops the train of pulses on PPS signal output after the programmed stop time specified in Step 6 elapses.

Example

You can stop the pulse train at any time by programming 0101 in the PPSCMD field. Similarly, you can cancel the Stop Pulse train command (given in Step 7) by programming 0110 in the PPSCMD field before the time (programmed in Step 6) elapses. You can cancel the pulse train generation by programming 0011 in the PPSCMD field before the programmed start time (in Step 2) elapses.

44.3.18.12.4 Generating an Interrupt without Affecting the PPS

The MAC_PPS_Control register enable you to set the Target Time registers to generate interrupts, to generate interrupts and the ppps start and stop time, or to generate only the pps start and stop time. This topic discusses the procedure to generate only an interrupt event.

The Bits [6:5], TRGTMODSEL, of the MAC_PPS_Control Register enable you to program the Target Time registers (register 736 and 737) to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

Complete the following steps to program the Target Time registers to generate only interrupt event:

Steps

1. Program 00 (for interrupt) in Bits [6:5], TRGTMODSEL, of MAC_PPS_Control Register. This instructs the MAC to use the Target Time registers for target time interrupt.
2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.
   If Bits [6:5], TRGTMODSEL, are changed (for example, to control the PPS), then the interrupt generation is over-written with the new mode and new programmed Target Time register value.

44.3.18.13Programming Guidelines for Split Header on Receive

Split Header support is an optional feature that you can enable by selecting the Enable Split Header Feature option and by setting the SPH bit in the DMA_CH#_Control register. The DMA can process the header and payload of received packets separately.

When split header support is enabled, the buffer 1 (RDES0) of the descriptors is used only for the header. Therefore the DMA writes the header of the received packet in buffer 1 when FD bit of RDES3 is set. The payload of the received packet is directed to the buffer2 (RDES2).

For subsequent descriptors (FD is set to 0), buffer1 is not used and the payload is written only to buffer 2. When the host enables split header structure by setting the SPH Bit[24] of register DMA_CH[n]_Control_register, the host must ensure that Buffer1 is always valid in all descriptors by creating Buffer1 and indicating this by setting Bit[24] (BUF1V) of RDES3 for all descriptors.

44.3.18.14Programming Guidelines for VLAN filtering on Receive

In VLAN tag filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

Complete the following steps to program VLAN filtering on receive:

Steps

1. Program MAC_VLAN_Tag register for the following bit to select the filtering method:
   a) ETV: Enable 12-Bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   b) VTHM: VLAN Tag Hash Table Match Enable.
   c) ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should enabled by setting EDVLP)
   d) ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   e) DOVLTC: Ignores VLAN Type for Tag Match
   f) VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching
2. Program VL of MAC_VLAN_Tag Register for the 12-bit or 16-bit VLAN tag.
3. If Hash filtering of VLAN tag is enabled, program MAC_VLAN_Hash_Table Register. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table. To help you program the hash table, a sample C routine that generates a VLAN tag's 4-bit hash is included in resources/sample_codes/ directory of your workspace.

44.3.18.15Programming Guidelines for Extended VLAN Filtering and Routing on Receive

44.3.18.15.1 Programming Guidelines for Extended VLAN Filtering and Routing on Receive - Write

In the extended VLAN filtering, perfect filtering is done for the VLAN Tag Filters and for each VLAN Tag Filter, the VLAN Tag ID is compared. The combined result of all the enabled VLAN Tag Filters determines the overall VLAN Tag Filter result. The filter result is passed to the application as part of the status bits.

For the indirect access of the per VLAN Tag registers, follow these steps to write into the registers:

Steps

1. Write the required data into the MAC VLAN Tag Data register.
2. Program the VLAN Tag Control Register's OFS field with the required Filter Register's offset and command type to the CT field. For a write command, set this bit to 0.
3. Write 1 to the OB field and wait till the OB bit is reset to do the next write. This will guarantee that the appropriate VLAN Tag Filter Register has been programmed.

44.3.18.15.2 Programming Guidelines for Extended VLAN Filtering and Routing on Receive - Read

In the extended VLAN filtering, perfect filtering is done for the VLAN Tag Filters and for each VLAN Tag Filter, the VLAN Tag ID is compared. The combined result of all the enabled VLAN Tag Filters determines the overall VLAN Tag Filter result. The filter result is passed to the application as part of the status bits. To read from each VLAN Tag register, set the OFS field in the VLAN Tag Control Register and set the OB field.

For the indirect access of the per VLAN Tag registers, follow these steps to read from the registers:

Steps

1. Program the VLAN Tag Control Register's OFS Field with the required register's offset and command type to the CT field. For a read command, set this bit to 1.
2. Write 1 to the OB field and wait till the OB bit is reset. The appropriate VLAN Tag Filter register's value is available in the MAC VLAN Tag Data Register.

44.3.18.16Programming sequence for Queue/Channel Based VLAN Inclusion Register

To program the queue/channel-based VLAN inclusion register, complete the following steps:

Note: Indirect VLAN include registers are not accessed while setting the CBTI bit.

Steps

1. Set the CBTI bit of MAC_VLAN_Incl register to 1, to enable queue/channel based VLAN Tag insertion on all transmitted packets. This bit must be set before any indirect access to the queue/channel specific MAC_VLAN_Incl(#i) register.
2. Program the VLAN Tag and VLAN Type to be inserted in packets from a particular queue/channel in VLT and CSVL fields of MAC_VLAN_Incl register; corresponding offset address in ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on) must be set. Set the RDWR bit to 0 to indicate write access. The write to byte 0 (byte 3 in Big Endian mode) of MAC_VLAN_Incl register initiates access to indirect access MAC_VLAN_Incl(#i) register.
3. The BUSY bit of MAC_VLAN_Incl register is set by DWC_ether_qos to indicate the progress of access to indirect access MAC_VLAN_Incl(#i) register. On completion of the access, the BUSY bit is cleared. The application must not attempt subsequent access to MAC_VLAN_Incl(#i) register when the BUSY bit is 1.
4. Repeat step 2 and step 3 to program VLAN Tag and VLAN Type to be inserted in packets from the remaining queues/channels. The application must ensure that the required VLAN Tag and VLAN Type for all the queues/channels are programmed; otherwise unintended VLAN Tag and VLAN Type might be inserted.

44.3.19 Definition of the PHY Interfaces MII, RMII and RGMII

The module has up to 3 PHY interfaces. Some products may contain less.

Table 500 PHY Interfaces of Gigabit Ethernet MAC

IF	Signal Name in Standard	GETH Signal Name	I/O
MII	TX_EN	TXEN	O
	TXD[3:0]	TXD[3:0]	O
	TX_ER	TXER	O
	TX_CLK	TXCLK	I
	COL	COL	I
	CRS	CRS	I
	RX_DV	RXDV	I
	RXD[3:0]	RXD[3:0]	I
	RX_ER	RXER	I
	RX_CLK	RXCLK	I

Table 500 PHY Interfaces of Gigabit Ethernet MAC (cont'd)

IF	Signal Name in Standard	GETH Signal Name	I/O
RMII	TX_EN	TXEN	O
	TXD[1:0]	TXD[1:0]	O
	CRS_DV	CRSDV	I
	RXD[1:0]	RXD[1:0]	I
	REF_CLK	REFCLK	I
RGMII	TD[3:0]	TXD[3:0]	O
	TX_CTL	TCTL	O
	TXC	TXCLK	O
	n. a.	GREFCLK (Reference input)	I
	RD[3:0]	RXD[3:0]	I
	RX_CTL	RCTL	I
	RXC	RXCLK	I
MDIO	MDC	MDC	O
	MDIO	MDIO	I/O

44.4 Functional Specifications for Flexible Header Feature (FHE)

44.4.1 Overview of Feature

- Programmable control to enabled/disable Flexible Header Feature
- The Flexible Header position is fixed within the Ethernet packet
  - After 12 bytes from the start of packet data (After MAC SA field in the Ethernet packet)
  - This Header Length is fixed to 4 bytes .
- All the packets (from Application on Tx path and from Line on the Rx path) will have this Flexible Header
  - The Pause/PFC frames are sent/received without this Flexible Header as Pause/PFC is link to link communication generated and consumed at MAC Layer.
- All the existing offload features of the QOS IP like COE, IEEE 1588 PTP, VLAN tagging, Packet filtering and queueing based on L2/L3/L4 headers, etc. will be transparently supported when the FH is present.
- PTP Offload (Automatic generation of SYNC, Delay_Resp, Pdelay_Req, and Pdelay_Resp PTP packets in the Transmit path) features will not be supported when Flexible Header feature is enabled.

44.4.1.1 Trasmit Path Features

- The application/software will insert the Header inside the packet to be transmitted at the correct location
- The Hardware parser logic in TxCOE is modified to be aware of the presence of this additional header
- The VLAN/SA Insertion/Replacement/Deletion features are supported in the presence of Flexible Header
- The IEEE 1588 based One-step timestamping (OST) feature supported in the presence of Flexible Header
- The application/host software must ensure that the minimum and maximum Ethernet packet size rules are not violated after the insertion of the FH field. The MAC Transmitted will include every packet byte (including FH) in the updation of RMON counters
- The Ethernet CRC computation must include the FH field for all the frames that have the FH field after SA.

44.4.1.2 Receive Path Features

- In this mode, the MAC Receiver will always strip the 4 Bytes of Flexible Header present after the MAC SA field and provides the 4B Flexible Header as a packet status for all incoming Rx packets (except Pause/PFC frames)
- For Pause/PFC frames, DUT checks for the presence of MAC DA to be 01:80:C2:00:00:01 else it will be considered as non-pause/non-PFC frame for which Flexible header after MAC SA will always be stripped. This means that “Unicast” PAUSE/PFC will not be supported in this mode.
- The hardware parser logic in RxCOE (L2/VLAN/L3/L4 filter) is aware of the Flexible Header.
- Double VLAN tagging mode will be supported even when FH field is present. The 4B of FH will be provided in the Descriptor Status field by the Receive DMA after transferring the packet to the host memory.
- The RMON counters in the MAC Receiver will count the FH bytes as part of the Ethernet packet received.
- The MAC Receiver will continue to check for the same min/max packet size rules, irrespective of the presence of the FH bytes.
- The Ethernet CRC received on the line must include the FH field for all the frames that have FH field.

NOTE: VLAN Tag in Rx Status feature is not supported when Flexible Header feature is used, as the corresponding Rx Packet status field is used to give the 4B of FH instead of the VLAN Tags. VLAN stripping feature is supported, for use cases where the VLAN fields can be used for routing and subsequently stripped.

44.4.2 Functional Description of the FHE Functionality

The flexible header functionality is enabled in both Transmit and Receive directions when bit[31] (FHE) of the MAC_Ext_Configuration register is set to 1. When this is enabled, all the incoming packets (from Application on Tx path and from Line on the Rx path) must have this Flexible Header. Hence this bit must be set only once during initialization, before any Transmit DMA is started or MAC Receiver is enabled. Otherwise, a mix of packets with and without FH will be present in the data-paths.

44.4.2.1 Transmit Checksum Offload Engine (TxCOE)

When the Flexible Header functionality is enabled, the TxCOE engine will consider the 4B Flexible header as a part of Ethernet Header (6 DA + 6 SA + 4 FH = 16 bytes total) and look for Type/Length/VLAN fields from Byte 17 onwards in the Ethernet packet.

44.4.2.2 Transmit SA/VLAN Insertion/Replacment

The Flexible Header is assumed to be present after MAC SA (incase of SA Insertion is not enabled) and after the MAC DA (in case SA Insertion is enabled).

The VLAN Insertion/Replacement/Deletion point will be after the Flexible Header field.

44.4.2.3 PTP One_Step Processing

When the Flexible Header functionality is enabled , the PTP Type is decoded assuming the EtherType/Tag starts after the Flexible Header (Flexible header is assumed to be after MAC SA). So, determining the PTP Type (without Tag, with Single Tag or Double Tag) is done accordingly.

When PTP Over UDP/IP is enabled, the Correction field offset is determined by TxCOE engine which parses the Tx packet and offset determination is based on presence of the Flexible header.

44.4.2.4 Transmit Pad Insertion

When Flexible Header functionality is enabled then Padding is appended only when total packet size is less than 60B including Flexible header. Note that padding is done only when CPC[1:0] (Pad CRC Control is set to 2'b00).

44.4.2.5 Transmit Packet Length implication

When Flexible Header functionality is enabled then the following limits are determined by including Flexible header as part of packet data

- Jabber Limit
- Carrier Extension
- Collision Window

All the management (RMON/MMC) counters consider Flexible header as part of packet data.

There is no impact of the FH in any of the other Transmit functions like TX Scheduler (CBS, DWRR, WFQ algorithms). These functions use only the total length of Ethernet packet for their calculations for which the FH is taken as part of the total packet.

The Transmit PAUSE/PFC packet generator will operate as usual and will not insert any FH in the generated PAUSE/PFC packet.

44.4.2.6 Receive Status

When the Flexible Header feature is enabled, the VLAN status in the Rx Descriptor in DMA configurations and VLAN Status in MTL configurations will not be provided and instead the 4B stripped Flexible header of the received packets will be provided in those fields.

As VLAN Status in the Rx Status is not be supported when Flexible Header feature is enabled, the EIVLRXS and EVLRXS fields of MAC_VLAN _Tag_Ctrl will be ignored and treated as always "0".

44.4.2.7 MAC Receiver functions and offloads

When the Flexible Header functionality is enabled, the MAC Receiver will consider the 4B Flexible Header as a part of Ethernet Header. Hence it will look for the Ethernet Type/VLAN Tag from Byte 17 of the received packet instead of Byte 13.

Hence all offload functions like Type/VLAN-/L3-/L4-header based filtering and queueing, PTP/AVTP/OAM packet processing, will look for their corresponding header fields at an offset of 4 Bytes as compared to the normal mode (FH mode is disabled).

The only exception is when the DA field = 01:80:C2:00:00:01 (Special multicast address) in the received packet. Such packets will be parsed for PFC/PAUSE control fields in the usual position assuming that FH field is absent even when FH mode is enabled.

Note that the RMON Octet counters (MAC layer) will count the FH as part of the packet length. Similarly, the maximum packet size rule checks will consider FH as part of the Ethernet packet for consideration.

44.4.3 High Level Microarchitecture

44.4.3.1 Tx Side

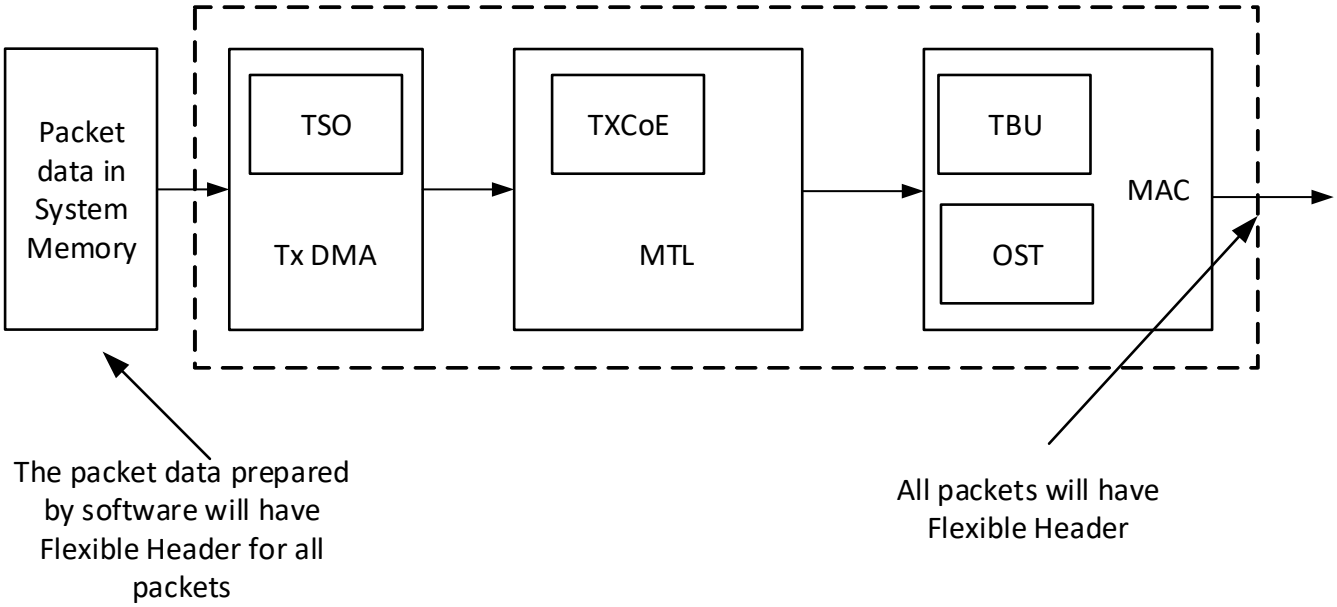


Figure 720 Tx Flow

On Tx side, all the packets initiated by the software will have Flexible Header when the FHE bit in MAC_Ext_Configuration register is set. The TxCOE engines assume 4B flexible header after MAC SA as a part of Ethernet header. The TBU block in the MAC takes are of SA/VLAN insertion/replacement in the presence of the Flexible Header. It also identifies the start of the PTP Header for PTPover Ethernet packet during One-step Timestamping.

Hence the RTL Design changes are limited to these blocks – TxCOE, OST & TBU.

44.4.3.2 Rx Side

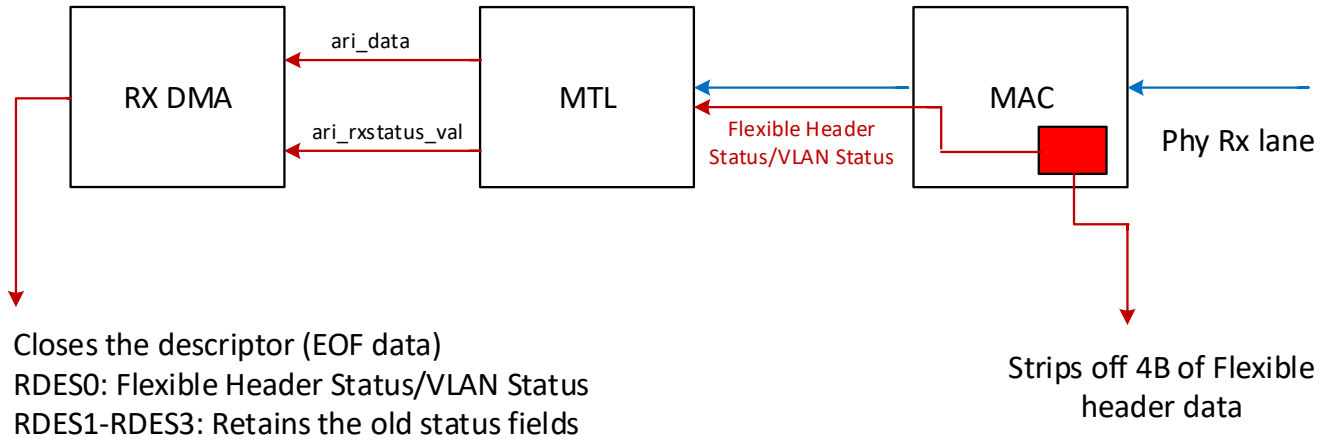


Figure 721 Rx Flow

For the incoming Rx packets, the 4B Flexible header after SA is stripped in the MAC if the FHE field of MAC_Ext_Configuration register is set. Pause/PFC packets are assumed not to have a Flexible header and hence no stripping is done. The stripped data is provided as one of the Status word to MTL.

Hence, the main RTL/Design changes are restricted to RPE and RRPE blocks.

- In the RRPE module the FH is stripped and the rest of packet given to all the other functions.
- In the RPE module the stripped/extracted FH is made part of the Rx Status.

44.4.4 DMA Descriptors / MTL Status

44.4.4.1 Receive Descriptor (DMA Configuration)

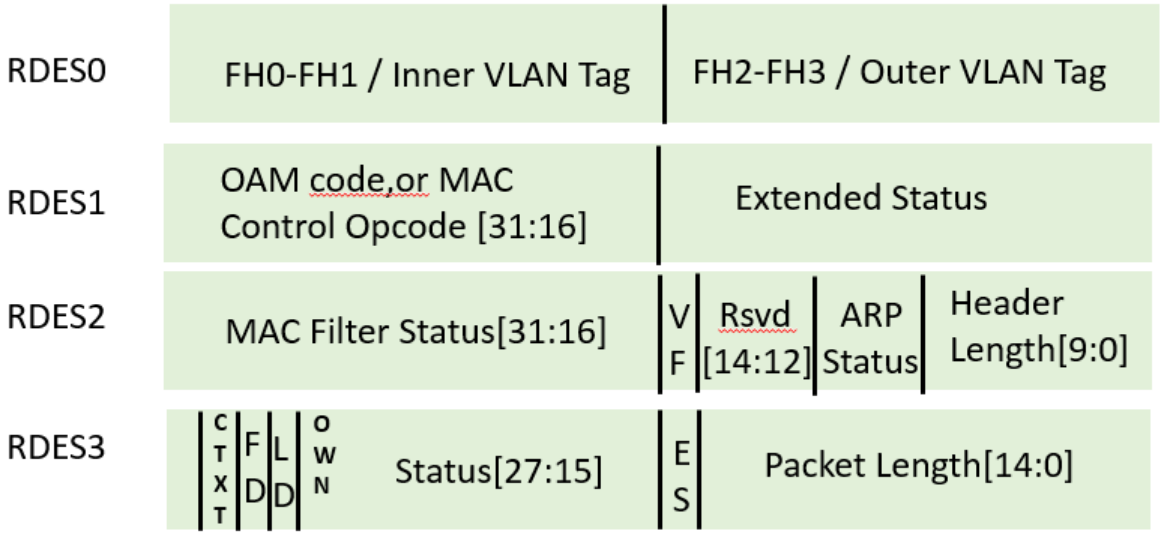


Figure 722 Receive Descriptor (DMA Configuration)

When FHE bit of MAC_Ext_Configuration register is set, then the stripped 4B Flexible header data (FH0,FH1,FH2,FH3 in the order the header bytes arrive at the Rx lane) will be provided in the RDES0 during Rx descriptor write back.

44.4.4.2 ARI Status (MTL Configuration)

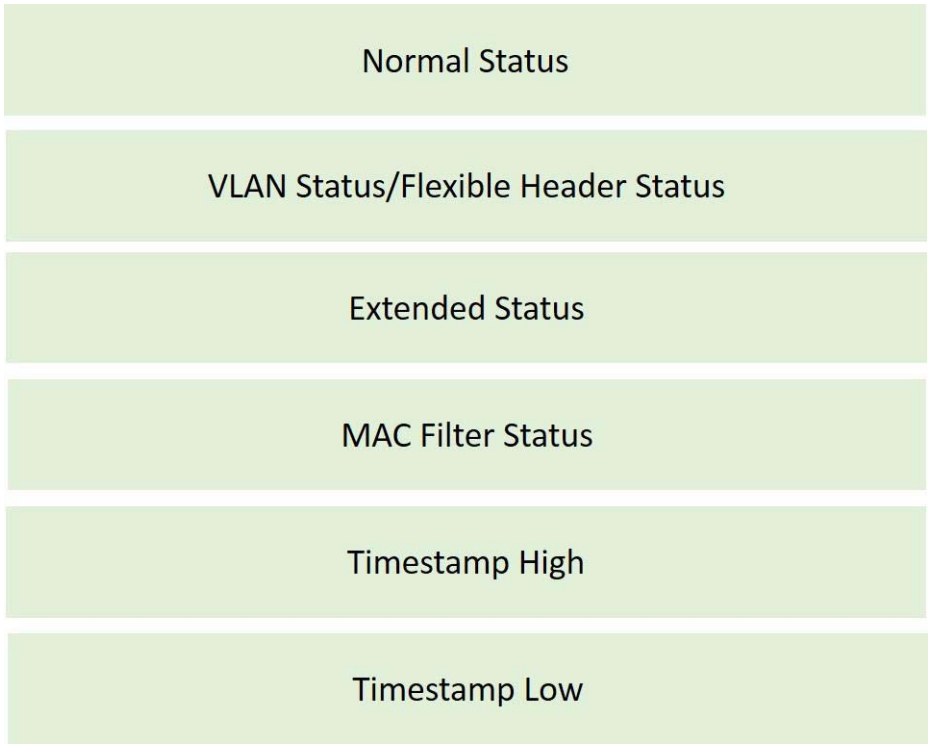


Figure 723 ARI Status (MTL Configuration)

When FHE bit is enabled in the configuration then the stripped 4B Flexible header data is provided in the second status word in the place of VLAN status.

Flexible Header Status:

If FHE bit of MAC_Ext_Configuration register is set then 4B of Flexible header after SA is stripped from the incoming Rx packets and provided in this status word.

Table 501

Field	Description
31:00	Flexible Header
	This field indicates the stripped Flexible Header data.
	Data presented as {FH0,FH1,FH2,FH3} in the same order the header bytes arrive on the line.

44.4.5 Software Initialization and Guidelines

The FHE bit in the MAC_Ext_Configuration should be enabled/disabled only when the MAC Transmitter and Receiver, all Transmit DMA are disabled and no packets are in the pipeline.

The MAC Tx and Rx can be disabled by setting 0 on bit[0] (RE) and bit[1] (TE) of the MAC_Configuration register.

The packets not being in the design pipeline can be checked by monitoring the

- MTL_TXQn_Debug register(s)
- MTL_RXQn_Debug register(s)
- DMA_Debug_Status0/1/2 registers

44.5 Registers

General notes on registers:

- All CSRs are implemented as 32-bit registers and can be accessed in 1 read/write cycle with a 32-bit MCI interface (or 32-bit AHB slave port). If you initiate a non-32-bit register access, the byte enable signals (mci_be_i[3:0]) qualify the corresponding register bytes. Only write operations are qualified with byte enables, while read operations are always 32-bit, unless the register is affected by a read operation. Byte enable signals qualify reads to such registers as, for example, the MMC counter registers.
- The transfer of particular register contents (e.g., MAC DA address register) to the Transmit or Receive clock domains are initiated when a particular byte is written. This is indicated in the register descriptions, where applicable.
- After a write operation, there should not be any further writes to the same location until the first write is updated. Otherwise, the second write operation will not get updated to the destination clock domain. Thus the delay between two writes to the same register location should be at least 6 cycles of the destination clock (PHY receive clock or PHY transmit clock).

44.5.1 Register Description

Accesses to reserved addresses (part of the address space but no register defined) will generate a bus error (BE).

All registers in the DMA and MAC Registers map (offset 0000_H - $1FFC_H$) are P = ACCEN protected. The DMA registers for additional Module Control are ACCEN protected according to separate table below.

All registers in the GETH address spaces are reset with the application reset (definition see SCU section “Reset Operation”).

Register Overview Tables of GETH

Table 502 Register Overview - GETH (ascending Offset Address)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
MAC_CONFIGURATION	MAC Configuration Register	0000_H	U,SV	U,SV,P	179
MAC_EXT_CONFIGURATION	MAC Extended Configuration Register	0004_H	U,SV	U,SV,P	184
MAC_PACKET_FILTER	MAC Packet Filter Register	0008_H	U,SV	U,SV,P	187
MAC_WATCHDOG_TIMEOUT	MAC Watchdog Timeout Register	$000C_H$	U,SV	U,SV,P	189
MAC_VLAN_TAG_CTRL	MAC VLAN Tag Control Register	0050_H	U,SV	U,SV,P	190
MAC_VLAN_TAG_DATA	MAC VLAN Tag Data Register	0054_H	U,SV	U,SV,P	193
MAC_VLAN_TAG_FILTER_i	MAC VLAN Tag Filter i Register	0054_H	U,SV	U,SV,P	194
MAC_VLAN_HASH_TABLE	MAC VLAN Hash Table Register	0058_H	U,SV	U,SV,P	196
MAC_VLAN_INCL	MAC VLAN Tag Inclusion or Replacement Register	0060_H	U,SV	U,SV,P	197

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
MAC_VLAN_INCL_Q_i	MAC VLAN Tag Inclusion or Replacement Register per Queue	0060_H	U,SV	U,SV,P	199
MAC_INNER_VLAN_INCL_i	MAC Inner VLAN Tag Inclusion or Replacement Register	0064_H	U,SV	U,SV,P	200
MAC_Q0_TX_FLOW_CTRL	MAC Queue 0 TX Flow Control Register	0070_H	U,SV	U,SV,P	202
MAC_RX_FLOW_CTRL	MAC Receive Flow Control Register	0090_H	U,SV	U,SV,P	204
MAC_RXQ_CTRL4	MAC Receive Queue Control 4 register	0094_H	U,SV	U,SV,P	205
MAC_RXQ_CTRL0	MAC Receive Queue Control 0 Register	$00A0_H$	U,SV	U,SV,P	207
MAC_RXQ_CTRL1	MAC Receive Queue Control 1 Register	$00A4_H$	U,SV	U,SV,P	208
MAC_RXQ_CTRL2	MAC Receive Queue Control 2 Register	$00A8_H$	U,SV	U,SV,P	210
MAC_INTERRUPT_STATUS	MAC Interrupt Status Register	$00B0_H$	U,SV	U,SV,P	211
MAC_INTERRUPT_ENABLE	MAC Interrupt Enable Register	$00B4_H$	U,SV	U,SV,P	214
MAC_RX_TX_STATUS	MAC Receive Transmit Status Register	$00B8_H$	U,SV	U,SV,P	216
MAC_PMT_CONTROL_STATUS	MAC PMT Control and Status Register	$00C0_H$	U,SV	U,SV,P	218
MAC_RWK_PACKET_FILTER	MAC Wake-up Packet Filter Register	$00C4_H$	U,SV	U,SV,P	220
RWK_FILTER_COMMAND_0	MAC Wake-up Filter Command 0 Register	$00C4_H$	U,SV	U,SV,P	222
RWK_FILTER_OFFSET_0	MAC Wake-up Filter Offset 0 Register	$00C4_H$	U,SV	U,SV,P	224
RWK_FILTER_CRC_i	MAC Wake-up Filter CRC i Register	$00C4_H$	U,SV	U,SV,P	226
RWK_FILTER_BYTE_MASK_i	MAC Wake-up i Filter Byte Mask register	$00C4_H$	U,SV	U,SV,P	226
MAC_LPI_CONTROL_STATUS	MAC LPI Control and Status Register	$00D0_H$	U,SV	U,SV,P	227
MAC_LPI_TIMERS_CONTROL	MAC LPI Timers Control Register	$00D4_H$	U,SV	U,SV,P	230
MAC_LPI_ENTRY_TIMER	MAC LPI Entry Timer Register	$00D8_H$	U,SV	U,SV,P	231

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
MAC_1US_TIC_COUNTER	MAC One Microsecond Tic Counter Register	$00DC_H$	U,SV	U,SV,P	231
MAC_PHYIF_CONTROL_STATUS	MAC PHY Interface Control and Status Register	$00F8_H$	U,SV	U,SV,P	232
MAC_VERSION	MAC Version Register	0110_H	U,SV	U,SV,P	233
MAC_DEBUG	MAC Debug Register	0114_H	U,SV	U,SV,P	234
MAC_HW_FEATURE0	MAC Hardware Feature Register 0	$011C_H$	U,SV	U,SV,P	235
MAC_HW_FEATURE1	MAC Hardware Feature Register 1	0120_H	U,SV	U,SV,P	238
MAC_HW_FEATURE2	MAC Hardware Feature Register 2	0124_H	U,SV	U,SV,P	240
MAC_HW_FEATURE3	MAC Hardware Feature Register 3	0128_H	U,SV	U,SV,P	242
MAC_MDIO_ADDRESS	MAC MDIO Address Register	0200_H	U,SV	U,SV,P	243
MAC_MDIO_DATA	MAC MDIO Data Register	0204_H	U,SV	U,SV,P	247
MAC_CSR_SW_CTRL	MAC CSR Software Controls Register	0230_H	U,SV	U,SV,P	248
MAC_EXT_CFG1	MAC Extended Configuration Register 1	0238_H	U,SV	U,SV,P	248
MAC_ADDRESS0_HIGH	MAC Address 0 High Register	0300_H	U,SV	U,SV,P	249
MAC_ADDRESS0_LOW	MAC Address 0 Low Register	0304_H	U,SV	U,SV,P	250
MAC_ADDRESSi_HIGH	MAC Address i High Register	0308_H+(i-1)*8	U,SV	U,SV,P	251
MAC_ADDRESSi_LOW	MAC Address i Low Register	$030C_H$+(i-1)*8	U,SV	U,SV,P	252
MMC_CONTROL	MMC Control Register	0700_H	U,SV	U,SV,P	253
MMC_RX_INTERRUPT	MMC Receive Interrupts Register	0704_H	U,SV	U,SV,P	254
MMC_TX_INTERRUPT	MMC Transmit Interrupts Register	0708_H	U,SV	U,SV,P	258
MMC_RX_INTERRUPT_MASK	MMC Receive Interrupts Mask Register	$070C_H$	U,SV	U,SV,P	262
MMC_TX_INTERRUPT_MASK	MMC Transmit Interrupts Mask Register	0710_H	U,SV	U,SV,P	266
TX_OCTET_COUNT_GOOD_BAD	Good And Bad Transmitted Octet Count Register	0714_H	U,SV	U,SV,P	269

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
TX_PACKET_COUNT_GOOD_BAD	Good And Bad Transmitted Packets Count Register	0718_H	U,SV	U,SV,P	269
TX_BROADCAST_PACKETS_GOOD	Good Transmitted Broadcast Packets Count Register	$071C_H$	U,SV	U,SV,P	270
TX_MULTICAST_PACKETS_GOOD	Good Transmitted Multicast Packets Count Register	0720_H	U,SV	U,SV,P	270
TX_64OCTETS_PACKETS_GOOD_BAD	Good And Bad 64 Octets Packets Transmitted Count Register	0724_H	U,SV	U,SV,P	271
TX_65TO127OCTETS_PACKETS_GOOD_BAD	Good And Bad 65to127 Octets Packets Transmitted Count Register	0728_H	U,SV	U,SV,P	271
TX_128TO255OCTETS_PACKETS_GOOD_BAD	Good And Bad 128to255 Octets Packets Transmitted Count Register	$072C_H$	U,SV	U,SV,P	272
TX_256TO511OCTETS_PACKETS_GOOD_BAD	Good And Bad 256to511 Octets Packets Transmitted Count Register	0730_H	U,SV	U,SV,P	273
TX_512TO1023OCTETS_PACKETS_GOOD_BAD	Good And Bad 512to1023 Octets Packets Transmitted Count Register	0734_H	U,SV	U,SV,P	273
TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD	Good And Bad 1024toMax Octets Packets Transmitted Count Register	0738_H	U,SV	U,SV,P	274
TX_UNICAST_PACKETS_GOOD_BAD	Good Transmitted Unicat Packets Count Register	$073C_H$	U,SV	U,SV,P	274
TX_MULTICAST_PACKETS_GOOD_BAD	Good And Bad Transmitted Multicast Packets Count Register	0740_H	U,SV	U,SV,P	275
TX_BROADCAST_PACKETS_GOOD_BAD	Good And Bad Transmitted Broadcast Packets Count Register	0744_H	U,SV	U,SV,P	275
TX_UNDERFLOW_ERROR_PACKETS	Transmitted Underflow Error Packets Count Register	0748_H	U,SV	U,SV,P	276
TX_SINGLE_COLLISION_GOOD_PACKETS	Good Transmitted Single Collision Count Register	$074C_H$	U,SV	U,SV,P	276

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
TX_MULTIPLE_COLLISION_GOOD_PACKETS	Transmitted Multiple Collision Count Register	0750_H	U,SV	U,SV,P	277
TX_DEFERRED_PACKETS	Transmitted Deferred Packets Count Register	0754_H	U,SV	U,SV,P	277
TX_LATE_COLLISION_PACKETS	Transmitted Late Collision Packets Count Register	0758_H	U,SV	U,SV,P	278
TX_EXCESSIVE_COLLISION_PACKETS	Transmitted Excessive Collision Packets Count Register	$075C_H$	U,SV	U,SV,P	278
TX_CARRIER_ERROR_PACKETS	Transmitted Carrier Error Packets Count Register	0760_H	U,SV	U,SV,P	279
TX_OCTET_COUNT_GOOD	Good Transmitted Octet Count Register	0764_H	U,SV	U,SV,P	279
TX_PACKET_COUNT_GOOD	Good Transmitted Packet Count Register	0768_H	U,SV	U,SV,P	280
TX_EXCESSIVE_DEFERRAL_ERROR	Transmitted Excessive Deferral Error Count Register	$076C_H$	U,SV	U,SV,P	280
TX_PAUSE_PACKETS	Transmitted Pause Packets Count Register	0770_H	U,SV	U,SV,P	281
TX_VLAN_PACKETS_GOOD	Good Transmitted VLAN Packets Count Register	0774_H	U,SV	U,SV,P	281
TX_OSIZE_PACKETS_GOOD	Good Transmitted Osize Packets Count Register	0778_H	U,SV	U,SV,P	282
RX_PACKETS_COUNT_GOOD_BAD	Good And Bad Received Packets Count Register	0780_H	U,SV	U,SV,P	283
RX_OCTET_COUNT_GOOD_BAD	Good And Bad Received Octet Count Register	0784_H	U,SV	U,SV,P	283
RX_OCTET_COUNT_GOOD	Good Received Octet Count Register	0788_H	U,SV	U,SV,P	283
RX_BROADCAST_PACKETS_GOOD	Good Received Broadcast Packets Count Register	$078C_H$	U,SV	U,SV,P	284
RX_MULTICAST_PACKETS_GOOD	Good Received Multicast Packets Count Register	0790_H	U,SV	U,SV,P	284
RX_CRC_ERROR_PACKETS	Received CRC Error Packets Count Register	0794_H	U,SV	U,SV,P	285

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
RX_ALIGNMENT_ERROR_PACKETS	Received Alignment Error Count Register	0798_H	U,SV	U,SV,P	285
RX_RUNT_ERROR_PACKETS	Received Runtime Error Count Register	$079C_H$	U,SV	U,SV,P	286
RX_JABBER_ERROR_PACKETS	Received Jabber Error Count Register	$07A0_H$	U,SV	U,SV,P	287
RX_UNDERSIZE_PACKETS_GOOD	Good Received Undersized Packets Count Register	$07A4_H$	U,SV	U,SV,P	287
RX_OVERSIZE_PACKETS_GOOD	Good Received Oversized Packets Count Register	$07A8_H$	U,SV	U,SV,P	288
RX_64OCTETS_PACKETS_GOOD_BAD	Good And Bad 64 Octets Packets Received Count Register	$07AC_H$	U,SV	U,SV,P	288
RX_65TO127OCTETS_PACKETS_GOOD_BAD	Good And Bad 65to127 Octets Packets Received Count Register	$07B0_H$	U,SV	U,SV,P	289
RX_128TO255OCTETS_PACKETS_GOOD_BAD	Good And Bad 128to255 Octets Packets Received Count Register	$07B4_H$	U,SV	U,SV,P	290
RX_256TO511OCTETS_PACKETS_GOOD_BAD	Good And Bad 256to511 Octets Packets Received Count Register	$07B8_H$	U,SV	U,SV,P	290
RX_512TO1023OCTETS_PACKETS_GOOD_BAD	Good And Bad 512to1023 Octets Packets Received Count Register	$07BC_H$	U,SV	U,SV,P	291
RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD	Good And Bad 1024toMax Octets Packets Received Count Register	$07C0_H$	U,SV	U,SV,P	291
RX_UNICAST_PACKETS_GOOD	Good Received Unicat Packets Count Register	$07C4_H$	U,SV	U,SV,P	292
RX_LENGTH_ERROR_PACKETS	Received Length Error Packets Count Register	$07C8_H$	U,SV	U,SV,P	292
RX_OUT_OF_RANGE_TYPE_PACKETS	Received Out Of Range Type Count Register	$07CC_H$	U,SV	U,SV,P	293
RX_PAUSE_PACKETS	Received Pause Packets Count Register	$07D0_H$	U,SV	U,SV,P	293
RX_FIFO_OVERFLOW_PACKETS	Received FIFO Overflow Count Register	$07D4_H$	U,SV	U,SV,P	294

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
RX_VLAN_PACKETS_GOOD_BAD	Good And Bad Received VLAN Packets Count Registerv	$07D8_H$	U,SV	U,SV,P	294
RX_WATCHDOG_ERROR_PACKETS	Received Watchdog Error Count Register	$07DC_H$	U,SV	U,SV,P	295
RX_RECEIVE_ERROR_PACKETS	Received Receive Error Count Register	$07E0_H$	U,SV	U,SV,P	295
RX_CONTROL_PACKETS_GOOD	Good Received Control Packets Count Register	$07E4_H$	U,SV	U,SV,P	296
TX_LPI_USEC_CNTR	Transmitted LPI Microseconds Count Register	$07EC_H$	U,SV	U,SV,P	296
TX_LPI_TRAN_CNTR	Transmitted LPI Transition Count Register	$07F0_H$	U,SV	U,SV,P	297
RX_LPI_USEC_CNTR	Received Microseconds LPI Count Register	$07F4_H$	U,SV	U,SV,P	297
RX_LPI_TRAN_CNTR	Received LPI Transition Count Register	$07F8_H$	U,SV	U,SV,P	298
MMC_IPC_RX_INTERRUPT_MASK	MMC IPC Receive Interrupts Mask Register	0800_H	U,SV	U,SV,P	298
MMC_IPC_RX_INTERRUPT	MMC IPC Receive Interrupts Register	0808_H	U,SV	U,SV,P	302
RXIPV4_GOOD_PACKETS	Good Received RxIPv4 Packets Count Register	0810_H	U,SV	U,SV,P	306
RXIPV4_HEADER_ERROR_PACKETS	Received IPv4 Header Error Packets Count Register	0814_H	U,SV	U,SV,P	306
RXIPV4_NO_PAYLOAD_PACKETS	Received IPv4 No Payload Packets Count Register	0818_H	U,SV	U,SV,P	307
RXIPV4_FRAGMENTED_PACKETS	Received IPv4 Fragmented Packets Count Register	$081C_H$	U,SV	U,SV,P	307
RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS	Received IPv4 UPD Checksum Disabled Packets Count Register	0820_H	U,SV	U,SV,P	308
RXIPV6_GOOD_PACKETS	Good Received RxIPv6 Packets Count Register	0824_H	U,SV	U,SV,P	308

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
RXIPV6_HEADER_ERROR_PACKETS	Received IPv6 Header Error Packets Count Register	0828_H	U,SV	U,SV,P	309
RXIPV6_NO_PAYLOAD_PACKETS	Received IPv6 No Payload Packets Count Register	$082C_H$	U,SV	U,SV,P	309
RXUDP_GOOD_PACKETS	Good Received UDP Packets Count Register	0830_H	U,SV	U,SV,P	310
RXUDP_ERROR_PACKETS	Received UDP Error Packets Count Register	0834_H	U,SV	U,SV,P	310
RXTCP_GOOD_PACKETS	Good Received TCP Packets Count Register	0838_H	U,SV	U,SV,P	311
RXTCP_ERROR_PACKETS	Received TCP Error Packets Count Register	$083C_H$	U,SV	U,SV,P	311
RXICMP_GOOD_PACKETS	Good Received ICMP Packets Count Register	0840_H	U,SV	U,SV,P	312
RXICMP_ERROR_PACKETS	Received ICMP Error Packets Count Register	0844_H	U,SV	U,SV,P	312
RXIPV4_GOOD_OCTETS	Good Received IPV4 Octets Count Register	0850_H	U,SV	U,SV,P	313
RXIPV4_HEADER_ERROR_OCTETS	Received IPV4 Header Error Octets Count Register	0854_H	U,SV	U,SV,P	313
RXIPV4_NO_PAYLOAD_OCTETS	Received IPV4 No Payload Octets Count Register	0858_H	U,SV	U,SV,P	314
RXIPV4_FRAGMENTED_OCTETS	Received IPV4 Fragmented Octets Count Register	$085C_H$	U,SV	U,SV,P	315
RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS	Received IPV4 UPD Checksum Disabled Octets Count Register	0860_H	U,SV	U,SV,P	315
RXIPV6_GOOD_OCTETS	Good Received IPV6 Octets Count Register	0864_H	U,SV	U,SV,P	316
RXIPV6_HEADER_ERROR_OCTETS	Received IPV6 Header Error Octets Count Register	0868_H	U,SV	U,SV,P	316
RXIPV6_NO_PAYLOAD_OCTETS	Received IPV6 No Payload Octets Count Register	$086C_H$	U,SV	U,SV,P	317
RXUDP_GOOD_OCTETS	Good Received UDP Octets Count Register	0870_H	U,SV	U,SV,P	317
RXUDP_ERROR_OCTETS	Received UDP Error Octets Count Register	0874_H	U,SV	U,SV,P	318

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
RXTCP_GOOD_OCTETS	Good Received TCP Octets Count Register	0878_H	U,SV	U,SV,P	318
RXTCP_ERROR_OCTETS	Received TCP Error Octets Count Register	$087C_H$	U,SV	U,SV,P	319
RXICMP_GOOD_OCTETS	Good Received ICMP Octets Count Register	0880_H	U,SV	U,SV,P	319
RXICMP_ERROR_OCTETS	Received ICMP Error Octets Count Register	0884_H	U,SV	U,SV,P	320
MAC_TIMESTAMP_CONTROL	MAC Timestamp Control Register	$0B00_H$	U,SV	U,SV,P	320
MAC_SUB_SECOND_INCREMENT	MAC Sub-Second Increment Register	$0B04_H$	U,SV	U,SV,P	324
MAC_SYSTEM_TIME_SECONDS	MAC System Time Seconds Register	$0B08_H$	U,SV	U,SV,P	325
MAC_SYSTEM_TIME_NANOSECONDS	MAC System Time Nanoseconds Register	$0B0C_H$	U,SV	U,SV,P	325
MAC_SYSTEM_TIME_SECONDS_UPDATE	MAC System Time Seconds Update Register	$0B10_H$	U,SV	U,SV,P	326
MAC_SYSTEM_TIME_NANOSECONDS_UPDATE	MAC System Time Nanoseconds Update Register	$0B14_H$	U,SV	U,SV,P	327
MAC_TIMESTAMP_ADDEND	MAC Timestamp Addend Register	$0B18_H$	U,SV	U,SV,P	328
MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS	MAC System Time Higher Word Seconds Register	$0B1C_H$	U,SV	U,SV,P	329
MAC_TIMESTAMP_STATUS	MAC Timestamp Status Register	$0B20_H$	U,SV	U,SV,P	329
MAC_TX_TIMESTAMP_STATUS_NANOSECONDS	MAC Transmit Timestamp Nanoseconds Status Register	$0B30_H$	U,SV	U,SV,P	332
MAC_TX_TIMESTAMP_STATUS_SECONDS	MAC Transmit Timestamp Seconds Status Register	$0B34_H$	U,SV	U,SV,P	333
MAC_TIMESTAMP_INGRESS_ASYM_CORR	MAC Timestamp Ingress Asymmetry Correction Register	$0B50_H$	U,SV	U,SV,P	333

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
MAC_TIMESTAMP_EGRESS_ASYM_CORR	MAC Timestamp Egress Asymmetry Correction Register	$0B54_H$	U,SV	U,SV,P	334
MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND	MAC Timestamp Ingress Correction Nanoseconds Register	$0B58_H$	U,SV	U,SV,P	335
MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND	MAC Timestamp Egress Correction Nanoseconds Register	$0B5C_H$	U,SV	U,SV,P	335
MAC_TIMESTAMP_INGRESS_CORR_SUBNANOSEC	MAC Timestamp Ingress Correction Subnanoseconds Register	$0B60_H$	U,SV	U,SV,P	336
MAC_TIMESTAMP_EGRESS_CORR_SUBNANOSEC	MAC Timestamp Egress Correction Subnanoseconds Register	$0B64_H$	U,SV	U,SV,P	336
MAC_PPS_CONTROL	MAC PPS Control Register	$0B70_H$	U,SV	U,SV,P	337
MAC_PPS0_TARGET_TIME_SECONDS	MAC PPS 0 Target Time Seconds Register	$0B80_H$	U,SV	U,SV,P	340
MAC_PPS0_TARGET_TIME_NANOSECONDS	MAC PPS 0 Target Time Nanoeconds Register	$0B84_H$	U,SV	U,SV,P	340
MAC_PPS0_INTERVAL	MAC PPS 0 Interval Register	$0B88_H$	U,SV	U,SV,P	341
MAC_PPS0_WIDTH	MAC PPS 0 Width Register	$0B8C_H$	U,SV	U,SV,P	342
MTL_OPERATION_MODE	MTL Operation Mode Register	$0C00_H$	U,SV	U,SV,P	343
MTL_INTERRUPT_STATUS	MTL Interrupt Status Register	$0C20_H$	U,SV	U,SV,P	344
MTL_RXQ_DMA_MAP0	MTL Receive Queue and DMA Channel Mapping 0 Register	$0C30_H$	U,SV	U,SV,P	345
MTL_TXQ0_OPERATION_MODE	MTL Queue 0 Transmit Operation Mode Register	$0D00_H$	U,SV	U,SV,P	348
MTL_TXQ0_UNDERFLOW	MTL Queue 0 Transmit Underflow Counter Register	$0D04_H$	U,SV	U,SV,P	350

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
MTL_TXQ0_DEBUG	MTL Queue 0 Transmit Debug Register	$0D08_H$	U,SV	U,SV,P	350
MTL_TXQ0_ETS_STATUS	MTL Queue 0 Transmit Status Register	$0D14_H$	U,SV	U,SV,P	352
MTL_TXQ0_QUANTUM_WEIGHT	MTL Queue 0 Transmit Quantum or Weights Register	$0D18_H$	U,SV	U,SV,P	352
MTL_Q0_INTERRUPT_CONTROL_STATUS	MTL Queue 0 Interrupt Control Status Register	$0D2C_H$	U,SV	U,SV,P	353
MTL_RXQ0_OPERATION_MODE	MTL Queue 0 Receive Operation Mode Register	$0D30_H$	U,SV	U,SV,P	355
MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT	MTL Queue 0 Receive Missed Packet and Overflow Counter Register	$0D34_H$	U,SV	U,SV,P	357
MTL_RXQ0_DEBUG	MTL Queue 0 Receive Debug Register	$0D38_H$	U,SV	U,SV,P	359
MTL_RXQ0_CONTROL	MTL Queue 0 Receive Control Register	$0D3C_H$	U,SV	U,SV,P	360
MTL_TXQi_OPERATION_MODE	MTL Queue i Transmit Operation Mode Register	$0D40_H$+(i -1)*40_H	U,SV	U,SV,P	360
MTL_TXQi_UNDERFLOW	MTL Queue i Transmit Underflow Counter Register	$0D44_H$+(i -1)*40_H	U,SV	U,SV,P	362
MTL_TXQi_DEBUG	MTL Queue i Transmit Debug Register	$0D48_H$+(i -1)*40_H	U,SV	U,SV,P	363
MTL_TXQi_ETS_CONTROL	MTL Queue i Transmit ETS Control Register	$0D50_H$+(i -1)*40_H	U,SV	U,SV,P	364
MTL_TXQi_ETS_STATUS	MTL Queue i Transmit ETS Status Register	$0D54_H$+(i -1)*40_H	U,SV	U,SV,P	366
MTL_TXQi_QUANTUM_WEIGHT	MTL Queue i Transmit Quantum or Weights Register	$0D58_H$+(i -1)*40_H	U,SV	U,SV,P	366
MTL_TXQi_SENDSLOPECREDIT	MTL Queue i Transmit SendSlopeCredit Register	$0D5C_H$+(i -1)*40_H	U,SV	U,SV,P	368
MTL_TXQi_HICREDIT	MTL Queue i Transmit HiCredit Register	$0D60_H$+(i -1)*40_H	U,SV	U,SV,P	368
MTL_TXQi_LOCREDIT	MTL Queue i Transmit LoCredit Register	$0D64_H$+(i -1)*40_H	U,SV	U,SV,P	369
MTL_Qi_INTERRUPT_CONTROL_STATUS	MTL Queue i Interrupt Status Register	$0D6C_H$+(i -1)*40_H	U,SV	U,SV,P	369

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
MTL_RXQi_OPERATION_MODE	MTL Queue i Receive Operation Mode Register	$0D70_H$+(i-1)*40_H	U,SV	U,SV,P	371
MTL_RXQi_MISSED_PACKET_OVERFLOW_CNT	MTL Queue i Receive Missed Packet and Overflow Counter Register	$0D74_H$+(i-1)*40_H	U,SV	U,SV,P	374
MTL_RXQi_DEBUG	MTL Queue i Receive Debug Register	$0D78_H$+(i-1)*40_H	U,SV	U,SV,P	375
MTL_RXQi_CONTROL	MTL Queue i Receive Control Register	$0D7C_H$+(i-1)*40_H	U,SV	U,SV,P	376
DMA_MODE	DMA Bus Mode Register	1000_H	U,SV	U,SV,P	377
DMA_SYSBUS_MODE	DMA System Bus Mode Register	1004_H	U,SV	U,SV,P	380
DMA_INTERRUPT_STATUS	DMA Interrupt Status Register	1008_H	U,SV	U,SV,P	381
DMA_DEBUG_STATUS0	DMA Debug Status 0 Register	$100C_H$	U,SV	U,SV,P	383
DMA_DEBUG_STATUS1	DMA Debug Status 1 Register	1010_H	U,SV	U,SV,P	384
DMA_CHi_CONTROL	DMA Channel i Control Register	1100_H+i*80_H	U,SV	U,SV,P	385
DMA_CHi_TX_CONTROL	DMA Channel i Transmit Control Register	1104_H+i*80_H	U,SV	U,SV,P	386
DMA_CHi_RX_CONTROL	DMA Channel i Receive Control Register	1108_H+i*80_H	U,SV	U,SV,P	388
DMA_CHi_TXDESC_LIST_ADDRESS	DMA Channel i Transmit Descriptor List Address Register	1114_H+i*80_H	U,SV	U,SV,P	390
DMA_CHi_RXDESC_LIST_ADDRESS	DMA Channel i Receive Descriptor List Address Register	$111C_H$+i*80_H	U,SV	U,SV,P	391
DMA_CHi_TXDESC_TAIL_POINTER	DMA Channel i Transmit Descriptor Tail Pointer Register	1120_H+i*80_H	U,SV	U,SV,P	392
DMA_CHi_RXDESC_TAIL_POINTER	DMA Channel i Recieve Descriptor Tail Pointer Register	1128_H+i*80_H	U,SV	U,SV,P	393
DMA_CHi_TXDESC_RING_LENGTH	DMA Channel i Transmit Descriptor Ring Length Register	$112C_H$+i*80_H	U,SV	U,SV,P	393
DMA_CHi_RXDESC_RING_LENGTH	DMA Channel i Recieve Descriptor Ring Length Register	1130_H+i*80_H	U,SV	U,SV,P	394

Table 502 Register Overview - GETH (ascending Offset Address) (cont'd)

Short Name	Description	Offset Address	Access Mode		Page Number
			Read	Write	
DMA_CHi_INTERRUPT_ENABLE	DMA Channel i Interrupt Enable Register	1134_H+i*80_H	U,SV	U,SV,P	394
DMA_CHi_RX_INTERRUPT_WATCHDOG_TIMER	DMA Channel i Recieve Interrupt Watchdog Timer Register	1138_H+i*80_H	U,SV	U,SV,P	397
DMA_CHi_SLOT_FUNCTION_CONTROL_STATUS	DMA Channel i Slot Function Control and Status Register	$113C_H$+i*80_H	U,SV	U,SV,P	398
DMA_CHi_CURRENT_APP_TXDESC	DMA Channel i Current Application Transmit Descriptor Register	1144_H+i*80_H	U,SV	U,SV,P	399
DMA_CHi_CURRENT_APP_RXDESC	DMA Channel i Current Application Receive Descriptor Register	$114C_H$+i*80_H	U,SV	U,SV,P	399
DMA_CHi_CURRENT_APP_TXBUFFER	DMA Channel i Current Application Transmit Buffer Address Register	1154_H+i*80_H	U,SV	U,SV,P	400
DMA_CHi_CURRENT_APP_RXBUFFER	DMA Channel i Current Application Receive Buffer Address Register	$115C_H$+i*80_H	U,SV	U,SV,P	400
DMA_CHi_STATUS	DMA Channel i Status Register	1160_H+i*80_H	U,SV	U,SV,P	401
DMA_CHi_MISS_FRAME_CNT	DMA Channel i Missed Frames Count Register	1164_H+i*80_H	U,SV	U,SV,P	405
CLC	Clock Control Register	2000_H	SV,U	SV,E,P	405
ID	Module Identification Register	2004_H	SV,U	BE	406
GPCTL	General Purpose Control Register	2008_H	SV,U	SV,P	407
ACCEN0	Access Enable Register 0	$200C_H$	U,SV	SV,SE	408
ACCEN1	Access Enable Register 1	2010_H	U,SV	SV,SE	409
KRST0	Kernel Reset Register 0	2014_H	U,SV	SV,E,P	409
KRST1	Kernel Reset Register 1	2018_H	U,SV	SV,E,P	410
KRSTCLR	Kernel Reset Status Clear Register	$201C_H$	U,SV	SV,E,P	411
ACCEN0Dx	Access Enable Register 0 for DMAx	2020_H+x*8	U,SV	SV,SE	411
ACCEN1Dx	Access Enable Register 1 for DMAx	2024_H+x*8	U,SV	SV,SE	412
SKEWCTL	Skew Control Register	2040_H	SV,U	SV,P	413

44.5.1.1 GMAC Registers

MAC Configuration Register

The MAC Configuration Register establishes the operating mode of the MAC.

MAC_CONFIGURATION
MAC Configuration Register **(0000_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31		SARC		IPC		IPG		GPSLCE	S2KP	CST	ACS	WD	BE	JD	JE
r		rw		rw		rw		rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS	FES	DM	LM	ECRSFD	DO	DCRS	DR	RES_7	BL		DC	PRELEN		TE	RE
rw	rw	rw	rw	rw	rw	rw	rw	r	rw		rw	rw		rw	rw

Field	Bits	Type	Description
RE	0	rw	**Receiver Enable** When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the GMII or MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the GMII or MII interface. Value After Reset: 0x0
TE	1	rw	**Transmitter Enable** When this bit is set, the Tx state machine of the MAC is enabled for transmission on the GMII or MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets. Value After Reset: 0x0
PRELEN	3:2	rw	**Preamble Length for Transmit packets** These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the full-duplex mode. Value After Reset: 0x0 00_B 7 bytes of preamble 01_B 5 bytes of preamble 10_B 3 bytes of preamble 11_B Reserved

Field	Bits	Type	Description
DC	4	rw	**Deferral Check** When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode. If the MAC is configured for 1000/2500 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on GMII or MII. The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. Value After Reset: 0x0
BL	6:5	rw	**Back-Off Limit** The back-off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000/2500 Mbps; 512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. where n = retransmission attempt The random integer r takes the value in the range 0 <= r < 2^k Value After Reset: 0x0 00_B 01_B k = min (n, 8) 10_B k = min (n, 4) 11_B k = min (n, 1)
RES_7	7	r	**Reserved** Value After Reset: 0x0
DR	8	rw	**Disable Retry** When this bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status. When this bit is reset, the MAC retries based on the settings of the BL field. Value After Reset: 0x0

Field	Bits	Type	Description
DCRS	9	rw	**Disable Carrier Sense During Transmission** When this bit is set, the MAC transmitter ignores the (G)MII CRS signal during packet transmission in the half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission. When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission. Value After Reset: 0x0
DO	10	rw	**Disable Receive Own** When this bit is set, the MAC disables the reception of packets when the gmii_txen_o is asserted in the half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY. Value After Reset: 0x0
ECRSFD	11	rw	**Enable Carrier Sense Before Transmission in Full-Duplex Mode** When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the full-duplex mode. The MAC starts the transmission only when the CRS signal is low. When this bit is reset, the MAC transmitter ignores the status of the CRS signal. Value After Reset: 0x0
LM	12	rw	**Loopback Mode** When this bit is set, the MAC operates in the loopback mode at GMII or MII. The (G)MII Rx clock input (clk_rx_i) is required for the loopback to work properly. This is because the Tx clock is not internally looped back. Value After Reset: 0x0
DM	13	rw	**Duplex Mode** When this bit is set, the MAC operates in the full-duplex mode in which it can transmit and receive simultaneously. Value After Reset: 0x0
FES	14	rw	**Speed** This bit selects the speed mode: The mac_speed_o[0] signal reflects the value of this bit. Value After Reset: 0x0 0_B 10 Mbps when PS bit is 1 and 1 Gbps when PS bit is 0 1_B 100 Mbps when PS bit is 1 and 2.5 Gbps when PS bit is 0
PS	15	rw	**Port Select** This bit selects the Ethernet line speed. This bit, along with Bit 14, selects the exact line speed. In the 10/100 Mbps-only (always 1) or 1000 Mbps-only (always 0) configurations, this bit is read-only (RO) with appropriate value. In default 10/100/1000 Mbps configurations, this bit is read-write (R/W). The mac_speed_o[1] signal reflects the value of this bit. Value After Reset: 0x0 0_B For 1000 Mbps operations or 2500 Mbps operations$_B$ 1_B For 10 or 100 Mbps operations $_B$

Field	Bits	Type	Description
JE	16	rw	**Jumbo Packet Enable** When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status. Value After Reset: 0x0
JD	17	rw	**Jabber Disable** When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes. When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet. Value After Reset: 0x0
BE	18	rw	**Packet Burst Enable** When this bit is set, the MAC allows packet bursting during transmission in the GMII half-duplex mode. Value After Reset: 0x0
WD	19	rw	**Watchdog Disable** When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes. When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes. Value After Reset: 0x0
ACS	20	rw	**Automatic Pad or CRC Stripping** When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When this bit is reset, the MAC passes all incoming packets to the application, without any modification. **Note**: For information about how the settings of CST bit and this bit impact the packet length, see the Table, Packet Length based on the CST and ACS Bit . Value After Reset: 0x0
CST	21	rw	**CRC stripping for Type packets** When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. **Note**: For information about how the settings of the ACS bit and this bit impact the packet length, see the Table, Packet Length based on the CST and ACS Bits. Value After Reset: 0x0

Field	Bits	Type	Description
S2KP	22	rw	**IEEE 802.3as Support for 2K Packets** When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets. When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see the table, Giant Packet Status based on S2KP and JE Bits. **Note**: When the JE bit is set, setting this bit has no effect on the giant packet status. Value After Reset: 0x0
GPSLCE	23	rw	**Giant Packet Size Limit Control Enable** When this bit is set, the MAC considers the value in GPSL field in MAC_Ext_Configuration register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit. When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet). The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status. Value After Reset: 0x0
IPG	26:24	rw	**Inter-Packet Gap** These bits control the minimum IPG between packets during transmission. LOST SEQUENCE DEFINITION This range of minimum IPG is valid in full-duplex mode. In the half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered. When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG. The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in MAC_Ext_Configuration register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in MAC_Ext_Configuration register. Value After Reset: 0x0 000_B 96 bit times 001_B 88 bit times 010_B 80 bit times 111_B 40 bit times

Field	Bits	Type	Description
IPC	27	rw	**Checksum Offload** When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled. The Layer 3 and Layer 4 Packet Filter and Enable Split Header features automatically selects the IPC Full Checksum Offload Engine on the Receive side. When any of these features are enabled, you must set the IPC bit. Value After Reset: 0x0
SARC	30:28	rw	**Source Address Insertion or Replacement Control** This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]: **Note**: • Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value. Value After Reset: 0x0 000_B The mti_sa_ctrl_i and ati_sa_ctrl_i input signals control the SA field generation. 001_B The mti_sa_ctrl_i and ati_sa_ctrl_i input signals control the SA field generation. 010_B If Bit 30 is set to 0, the MAC inserts the content of the MAC Address 0 registers (MAC registers 192 and 193) in the SA field of all transmitted packets.If Bit 30 is set to 1 and the Enable MAC Address Register 1 option is selected while configuring the core, the MAC inserts the content of the MAC Address 1 registers (MAC registers 194 and 195) in the SA field of all transmitted packets. 011_B If Bit 30 is set to 0, the MAC replaces the content of the MAC Address 0 registers (MAC registers 192 and 193) in the SA field of all transmitted packets.If Bit 30 is set to 1 and the MAC Address Register 1 is enabled, the MAC replaces the content of the MAC Address 1 registers (MAC registers 194 and 195) in the SA field of all transmitted packets.
RES_31	31	r	**Reserved for future use.**

MAC Extended Configuration Register

The MAC Extended Configuration Register establishes the operating mode of the MAC.

MAC_EXT_CONFIGURATION

MAC Extended Configuration Register **(0004_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FHE	RES_30	EIPG					EIPGEN	RES_23	HDSMS			RES_19	USP	SPEN	DCRCC
rw	r	rw					rw	r	rw			r	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_14		GPSL													
r		rw													

Field	Bits	Type	Description
GPSL	13:0	rw	**Giant Packet Size Limit** If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes. For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. When the Enable Double VLAN Processing option is selected, the MAC adds 8 bytes to the programmed value for double VLAN tagged packets. The value in this field is applicable when the GPSLCE bit is set in MAC_Configuration register. Value After Reset: 0x0
RES_15_14	15:14	r	**Reserved** Value After Reset: 0x0
DCRCC	16	rw	**Disable CRC Checking for Received Packets** When this bit is set, the MAC receiver does not check the CRC field in the received packets. When this bit is reset, the MAC receiver always checks the CRC field in the received packets. Value After Reset: 0x0
SPEN	17	rw	**Slow Protocol Detection Enable** When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid sub-types. When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets. Value After Reset: 0x0

Field	Bits	Type	Description
USP	18	rw	**Unicast Slow Protocol Packet Detect** When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the MAC_Address0_High and MAC_Address0_Low registers. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02). When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5. Value After Reset: 0x0
RES_19	19	r	**Reserved** Value After Reset: 0x0
HDSMS	22:20	rw	**Maximum Size for Splitting the Header Data** These bits indicate the maximum header size allowed for splitting the header data in the received packet: 101_B-111_B Reserved If the Enable Split Header Structure option is not selected, these bits are reserved and read-only (RO). Value After Reset: 0x0 000_B 64 bytes 001_B 128 bytes 010_B 256 bytes 011_B 512 bytes 100_B 1024 bytes
RES_23	23	r	**Reserved** Value After Reset: 0x0
EIPGEN	24	rw	**Extended Inter-Packet Gap Enable** When this bit is set, the MAC interprets EIPG field and IPG field in MAC_Configuration register together as minimum IPG greater than 96 bit times in steps of 8 bit times. When this bit is reset, the MAC ignores EIPG field and interprets IPG field in MAC_Configuration register as minimum IPG less than or equal to 96 bit times in steps of 8 bit times. Note: The extended Inter-Packet Gap feature must be enabled when operating in Full-Duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-Duplex mode. Value After Reset: 0x0

Field	Bits	Type	Description
EIPG	29:25	rw	**Extended Inter-Packet Gap** The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits), along with IPG field in MAC_Configuration register, gives the minimum IPG greater than 96 bit times in steps of 8 bit times: {EIPG, IPG} 8'h00 - 104 bit times 8'h01 - 112 bit times 8'h02 - 120 bit times ---------- 8'hFF - 2144 bit times Value After Reset: 0x0
RES_30	30	r	**Reserved** Value After Reset: 0x0
FHE	31	rw	**Flexible Header Enable** When this is set then it is expected that all the incoming packets from the Line to have 4B Flexible Header in the Rx path (except for the pause and PFC packets). Similarly, in the Tx path all the packets from application are expected to have 4B Flexible header. The position of the Flexible Header is always fixed at 12Bytes from the beginning of the packet (after MAC DA/SA).

MAC Packet Filter Register

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

MAC_PACKET_FILTER
MAC Packet Filter Register **(0008_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA	**RES_30_22**									**RES_21_20**		**RES_19_17**			**VTFE**
rw	r									r		r			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_11					**RES_10**	**SAF**	**SAIF**	**PCF**		**DBF**	**PM**	**DAIF**	**RES_2_1**		**PR**
r					r	rw	rw	rw		rw	rw	rw	r		rw

Field	Bits	Type	Description
PR	0	rw	**Promiscuous Mode** When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set. Value After Reset: 0x0

Field	Bits	Type	Description
RES_2_1	2:1	r	**Reserved**
DAIF	3	rw	**DA Inverse Filtering** When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed. Value After Reset: 0x0
PM	4	rw	**Pass All Multicast** When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. Value After Reset: 0x0
DBF	5	rw	**Disable Broadcast Packets** When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings. When this bit is reset, the AFM module passes all received broadcast packets. Value After Reset: 0x0
PCF	7:6	rw	**Pass Control Packets** These bits control the forwarding of all control packets (including unicast and multicast Pause packets). Value After Reset: 0x0 00_B The MAC filters all control packets from reaching the application. 01_B The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter. 10_B The MAC forwards all control packets to the application even if they fail the Address filter. 11_B The MAC forwards the control packets that pass the Address filter.
SAIF	8	rw	**SA Inverse Filtering** When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA Address filter. When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA Address filter. Value After Reset: 0x0

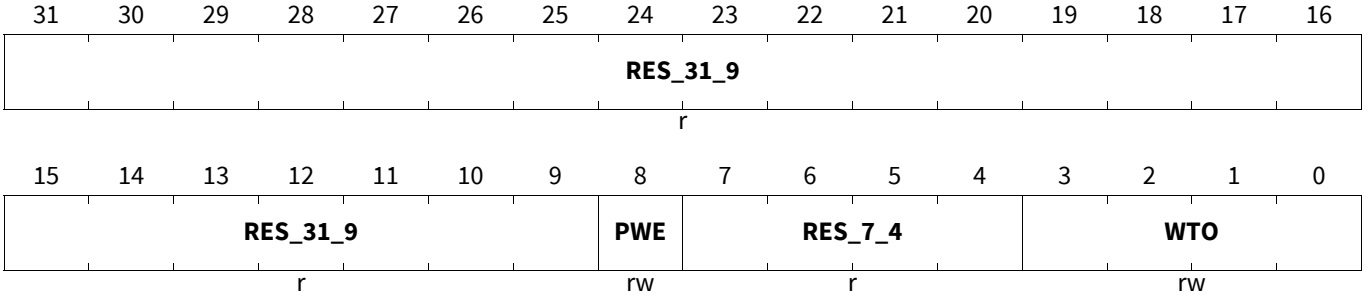
Field	Bits	Type	Description
SAF	9	rw	**Source Address Filter Enable** When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet. When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison. **Note**: According to the IEEE specification, Bit 47 of the SA is reserved. However, in DWC_ether_qos, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA. Value After Reset: 0x0
RES_10	10	r	**Reserved**
RES_15_11	15:11	r	**Reserved** Value After Reset: 0x0
VTFE	16	rw	**VLAN Tag Filter Enable** When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag. Value After Reset: 0x0
RES_19_17	19:17	r	**Reserved** Value After Reset: 0x0
RES_21_20	21:20	r	**Reserved**
RES_30_22	30:22	r	**Reserved** Value After Reset: 0x0
RA	31	rw	**Receive All** When this bit is set, the MAC Receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word. When this bit is reset, the Receiver module passes only those packets to the application that pass the SA or DA address filter. Value After Reset: 0x0

MAC Watchdog Timeout Register

The Watchdog Timeout register controls the watchdog timeout for received packets.

MAC_WATCHDOG_TIMEOUT

MAC Watchdog Timeout Register (000C$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
WTO	3:0	rw	**Watchdog Timeout** When the PWE bit is set and the WD bit of the MAC_Configuration register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet. Encoding is as follows: LOST SEQUENCE DEFINITION **Note**: When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped. Value After Reset: 0x0 0_H 2 KB 1_H 3 KB 2_H 4 KB 3_H 5 KB C_H 14 KB D_H 15 KB E_H 16383 Bytes F_H Reserved
RES_7_4	7:4	r	**Reserved** Value After Reset: 0x0
PWE	8	rw	**Programmable Watchdog Enable** When this bit is set and the WD bit of the MAC_Configuration register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in MAC_Configuration register. Value After Reset: 0x0
RES_31_9	31:9	r	**Reserved** Value After Reset: 0x0

MAC VLAN Tag Control Register

This register is the redefined format of the MAC VLAN Tag Register. It is used for indirect addressing. It contains the address offset, command type and Busy Bit for CSR access of the Per VLAN Tag registers.

MAC_VLAN_TAG_CTRL

MAC VLAN Tag Control Register **(0050_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIVLRXS	RES_30	EIVLS		RES_27	EDVLP	VTHM	EVLRXS	RES_23	EVLS		RES_19_20		ESVL	VTIM	RES_7_16
rw	r	rw		rw	rw	rw	rw	r	rw		r		rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_7_16									OFS					CT	OB
r									rw					rw	rw

Field	Bits	Type	Description
OB	0	rw	**Operation Busy** This bit is set along with a read or write command for initiating the indirect access to per VLAN Tag Filter register. This bit is reset when the read or write command to per VLAN Tag Filter indirect access register is complete.The next indirect register access can be initiated only after this bit is reset. During a write operation, the bit is reset only after the data has been written into the Per VLAN Tag register. During a read operation, the data should be read from the MAC_VLAN_Tag_Data register only after this bit is reset. Value After Reset: 0x0
CT	1	rw	**Command Type** This bit indicates if the current register access is a read or a write. When set, it indicate a read operation. When reset, it indicates a write operation. Value After Reset: 0x0
OFS	6:2	rw	**Offset** This field holds the address offset of the MAC VLAN Tag Filter Register which the application is trying to access. The width of the field depends on the number of MAC VLAN Tag Registers enabled. Value After Reset: 0x0
RES_7_16	16:7	r	**Reserved** Value After Reset: 0x0
VTIM	17	rw	**VLAN Tag Inverse Match Enable** When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched. Value After Reset: 0x0
ESVL	18	rw	**Enable S-VLAN** When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets. Value After Reset: 0x0

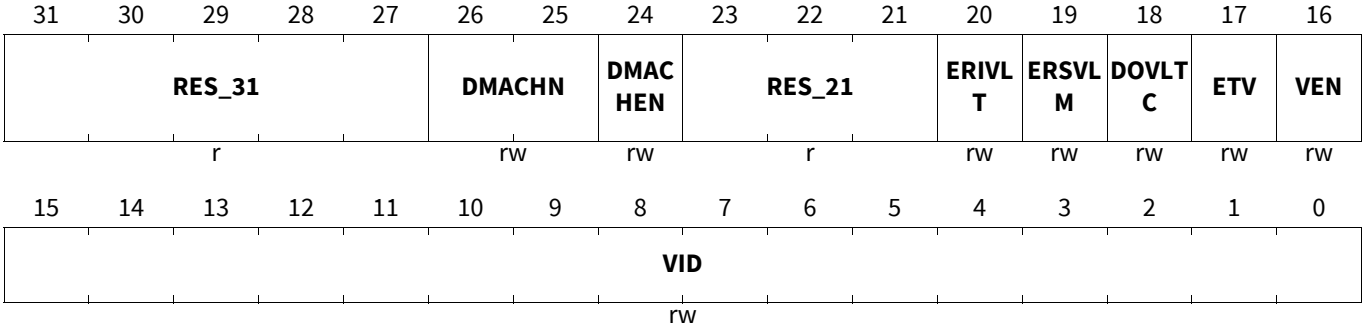
Field	Bits	Type	Description
RES_19_20	20:19	r	**Reserved** Value After Reset: 0x0
EVLS	22:21	rw	**Enable VLAN Tag Stripping on Receive** This field indicates the stripping operation on the outer VLAN Tag in received packet: Value After Reset: 0x0 00_B Do not strip 01_B Strip if VLAN filter passes 10_B Strip if VLAN filter fails 11_B Always strip
RES_23	23	r	**Reserved** Value After Reset: 0x0
EVLRXS	24	rw	**Enable VLAN Tag in Rx status** When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status. Value After Reset: 0x0
VTHM	25	rw	**VLAN Tag Hash Table Match Enable** When this bit is set, the most significant four bits of CRC of VLAN Tag (ones-complement of most significant four bits of CRC of VLAN Tag when ETV bit is reset) are used to index the content of the MAC_VLAN_Hash_Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN hash table. When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison. When the ETV bit is reset, the ones-complement of the CRC of the 16-bit VLAN tag is used for comparison. When this bit is reset, the VLAN Hash Match operation is not performed. Value After Reset: 0x0
EDVLP	26	rw	**Enable Double VLAN Processing** When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present). Value After Reset: 0x0
RES_27	27	rw	**Reserved** Value After Reset: 0x0
EIVLS	29:28	rw	**Enable Inner VLAN Tag Stripping on Receive** This field indicates the stripping operation on inner VLAN Tag in received packet: Value After Reset: 0x0 00_B Do not strip 01_B Strip if VLAN filter passes 10_B Strip if VLAN filter fails 11_B Always strip
RES_30	30	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
EIVLRXS	31	rw	**Enable Inner VLAN Tag in Rx Status** When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status. Value After Reset: 0x0

MAC VLAN Tag Data Register

This register holds the read/write data for Indirect Access of the Per VLAN Tag registers. During the read access, this field contains valid read data only after the OB bit is reset. During the write access, this field should be valid prior to setting the OB bit in the MAC_VLAN_Tag_Ctrl Register.

MAC_VLAN_TAG_DATA
MAC VLAN Tag Data Register **(0054_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
VID	15:0	rw	**VLAN Tag ID** This field holds the VLAN Tag value which is used by the MAC for perfect comparison. It is valid when VLAN Tag Enable is set. Value After Reset: 0x0
VEN	16	rw	**VLAN Tag Enable** This bit is used to enable or disable the VLAN Tag. When this bit is set, the MAC compares the VLAN Tag of received packet with the VLAN Tag ID. When this bit is reset, no comparison is performed irrespective of the programming of the other fields. Value After Reset: 0x0
ETV	17	rw	**12bits or 16bits VLAN comparison** This bit is valid only when VEN of the Filter is set. When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Value After Reset: 0x0

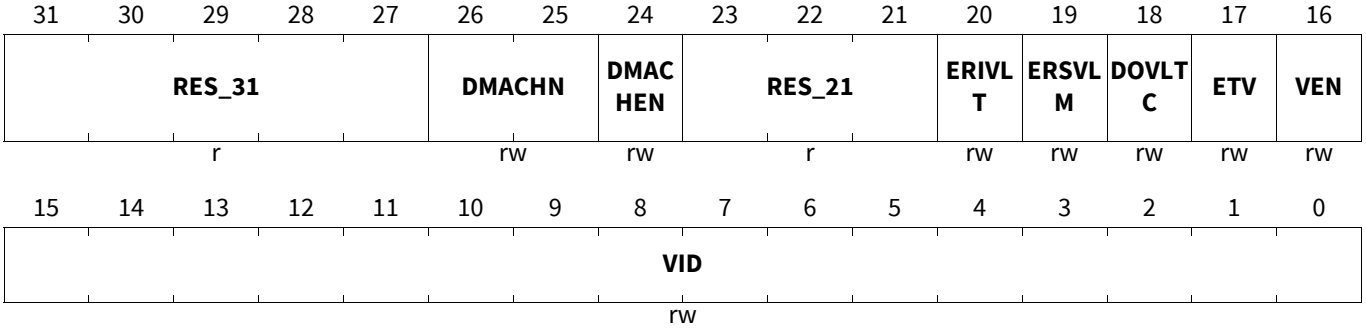
Field	Bits	Type	Description
DOVLTC	18	rw	**Disable VLAN Type Comparison** This bit is valid only when VLAN Tag Enable of the Filter is set. When this bit is set, the MAC does not check whether the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit is of type S-VLAN or C-VLAN. When this bit is reset, the MAC filters or matches the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit only when VLAN Tag type is similar to the one specified by the Enable S-VLAN Match for received Frames bit. Value After Reset: 0x0
ERSVLM	19	rw	**Enable S-VLAN Match for received Frames** This bit is valid only when VLAN Tag Enable of the Filter is set. When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets. Value After Reset: 0x0
ERIVLT	20	rw	**Enable Inner VLAN Tag Comparison** This bit is valid only when VLAN Tag Enable of the Filter is set. When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). Value After Reset: 0x0
RES_21	23:21	r	**Reserved** Value After Reset: 0x0
DMACHEN	24	rw	**DMA Channel Number Enable** This bit is the Enable for the DMA Channel Number value programmed in the field DMACH. When this bit is reset, the Routing does not occur based on VLAN Filter result. The frame is routed based on DA Based DMA Channel Routing. Value After Reset: 0x0
DMACHN	26:25	rw	**DMA Channel Number** The DMA Channel number to which the VLAN Tagged Frame is to be routed if it passes this VLAN Tag Filter is programmed in this field. If the Routing based on VLAN Tag Filter is not necessary, this field need not be programmed. Value After Reset: 0x0
RES_31	31:27	r	**Reserved** Value After Reset: 0x0

MAC VLAN Tag Filter i Register

This register contains VLAN Tag filter control information.

MAC_VLAN_TAG_FILTER_i (i=0-7)

MAC VLAN Tag Filter i Register **(0054_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
VID	15:0	rw	**VLAN Tag ID** This field holds the VLAN Tag value which is used by the MAC for perfect comparison. It is valid when VLAN Tag Enable is set. Value After Reset: 0x0
VEN	16	rw	**VLAN Tag Enable** This bit is used to enable or disable the VLAN Tag. When this bit is set, the MAC compares the VLAN Tag of received packet with the VLAN Tag ID. When this bit is reset, no comparison is performed irrespective of the programming of the other fields. Value After Reset: 0x0
ETV	17	rw	**12bits or 16bits VLAN comparison** This bit is valid only when VEN of the Filter is set. When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Value After Reset: 0x0
DOVLTC	18	rw	**Disable VLAN Type Comparison** This bit is valid only when VLAN Tag Enable of the Filter is set. When this bit is set, the MAC does not check whether the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit is of type S-VLAN or C-VLAN. When this bit is reset, the MAC filters or matches the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit only when VLAN Tag type is similar to the one specified by the Enable S-VLAN Match for received Frames bit. Value After Reset: 0x0
ERSVLM	19	rw	**Enable S-VLAN Match for received Frames** This bit is valid only when VLAN Tag Enable of the Filter is set. When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets. Value After Reset: 0x0

Field	Bits	Type	Description
ERIVLT	20	rw	**Enable Inner VLAN Tag Comparison** This bit is valid only when VLAN Tag Enable of the Filter is set. When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). Value After Reset: 0x0
RES_21	23:21	r	**Reserved** Value After Reset: 0x0
DMACHEN	24	rw	**DMA Channel Number Enable** This bit is the Enable for the DMA Channel Number value programmed in the field DMACH. When this bit is reset, the Routing does not occur based on VLAN Filter result. The frame is routed based on DA Based DMA Channel Routing. Value After Reset: 0x0
DMACHN	26:25	rw	**DMA Channel Number** The DMA Channel number to which the VLAN Tagged Frame is to be routed if it passes this VLAN Tag Filter is programmed in this field. If the Routing based on VLAN Tag Filter is not necessary, this field need not be programmed. Value After Reset: 0x0
RES_31	31:27	r	**Reserved** Value After Reset: 0x0

MAC VLAN Hash Table Register

When the VTHM bit of the MAC_VLAN_Tag register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of MAC_VLAN_Tag Register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a hash value of 4b'1000 selects Bit 8 of the VLAN Hash table.

The hash value of the destination address is calculated in the following way:

Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).

Perform bitwise reversal for the value obtained in step 1.

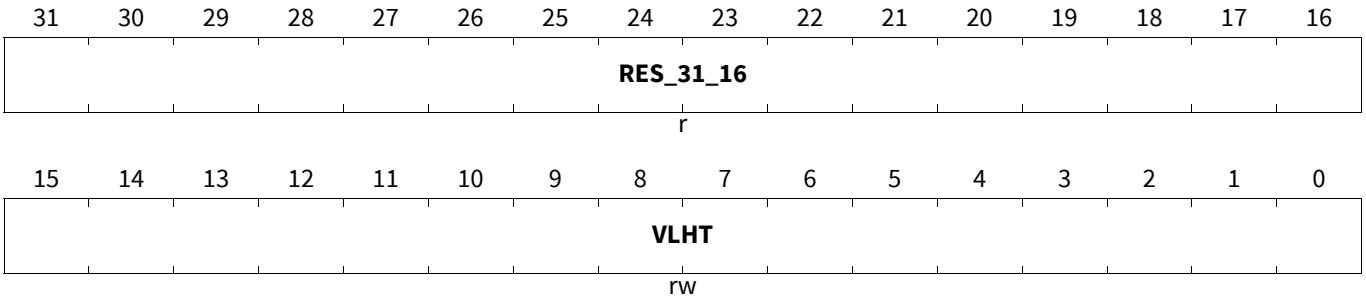
Take the upper four bits from the value obtained in step 2.

If the VLAN hash Table register is configured to be double-synchronized to the (G)MII clock domain, the synchronization is triggered only when Bits[15:8] (in little-endian mode) or Bits[7:0] (in big-endian mode) of this register are written.

If double-synchronization is enabled, consecutive writes to this register should be performed after at least four clock cycles in the destination clock domain.

MAC_VLAN_HASH_TABLE

MAC VLAN Hash Table Register (0058$_H$) Application Reset Value: 0000 0000$_H$



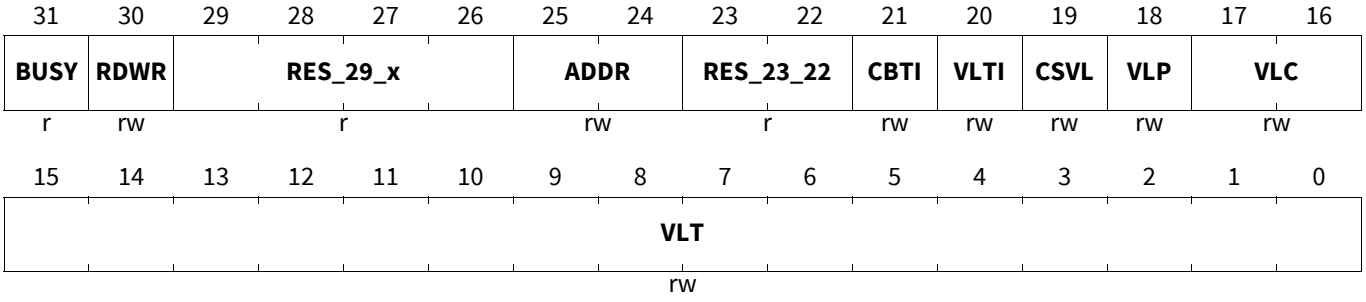
Field	Bits	Type	Description
VLHT	15:0	rw	**VLAN Hash Table** This field contains the 16-bit VLAN Hash Table. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

MAC VLAN Tag Inclusion or Replacement Register

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

MAC_VLAN_INCL

MAC VLAN Tag Inclusion or Replacement Register(0060$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
VLT	15:0	rw	**VLAN Tag for Transmit Packets** This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and Bits[11:0] are the VID field in the VLAN tag. The following list describes the bits of this field: • Bits[15:13]: User Priority • Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) • Bits[11:0]: VLAN Identifier (VID) field of VLAN tag Value After Reset: 0x0
VLC	17:16	rw	**VLAN Tag Control in Transmit Packets** **Note**: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value. Value After Reset: 0x0 00_B No VLAN tag deletion, insertion, or replacement 01_B VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags. 10_B VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. 11_B VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8).
VLP	18	rw	**VLAN Priority Control** When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, the mti_vlan_ctrl_i control input is used and bits[17:16] are ignored. Value After Reset: 0x0
CSVL	19	rw	**C-VLAN or S-VLAN** When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets. Value After Reset: 0x0
VLTI	20	rw	**VLAN Tag Input** When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from: • The Tx descriptor Value After Reset: 0x0

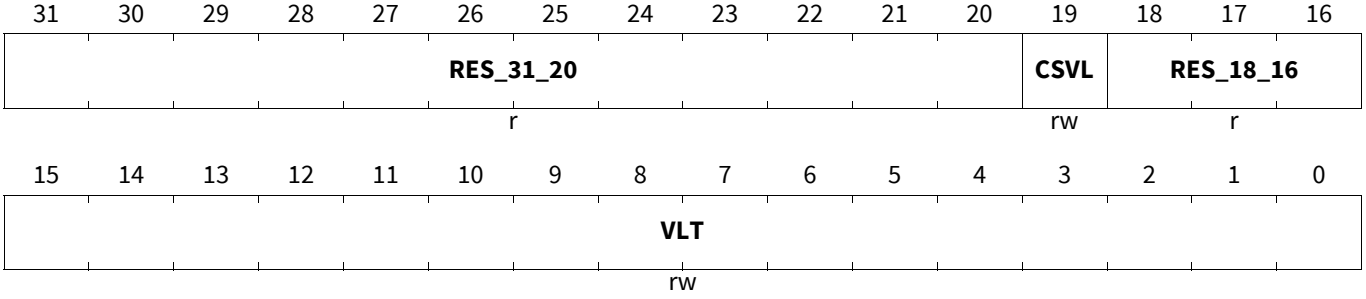
Field	Bits	Type	Description
CBTI	21	rw	**Channel based tag insertion** When this bit is set, outer VLAN tag is inserted for every packets transmitted by the MAC. The tag value is taken from the queue/channel specific VLAN tag register. The VLTI, VLP, VLC and VLT fields of this register are ignored when this bit is set. When this bit is set, a write operation to byte 3 of this register initiates the read/write access to the indirect register. When reset outer VLAN operation is based on the setting of VLTI, VLP, VLC and VLT fields of this register Value After Reset: 0x0
RES_23_22	23:22	r	**Reserved** Value After Reset: 0x0
ADDR	25:24	rw	**Address** This field selects one of the queue/channel specific VLAN Inclusion register for read/write access This doesn't have any effect when CBTI is reset Value After Reset: 0x0
RES_29_x	29:26	r	**Reserved** Value After Reset: 0x0
RDWR	30	rw	**Read write control** This bit controls the read or write operation for indirectly accessing the queue/channel specific VLAN Inclusion register. When set indicates write operation and when reset indicates read operation This doesn't have any effect when CBTI is reset Value After Reset: 0x0
BUSY	31	r	**Busy** Value After Reset: 0x0

MAC VLAN Tag Inclusion or Replacement Register per Queue

The Tx Queue x VLAN Tag Inclusion register contains the VLAN tag for insertion in the Transmit packets from Tx Queue x. It also contains the VLAN tag insertion controls.

MAC_VLAN_INCL_Q_i (i=0-3)

MAC VLAN Tag Inclusion or Replacement Register per Queue(0060_H) Application Reset Value: $0000\ 0000_H$



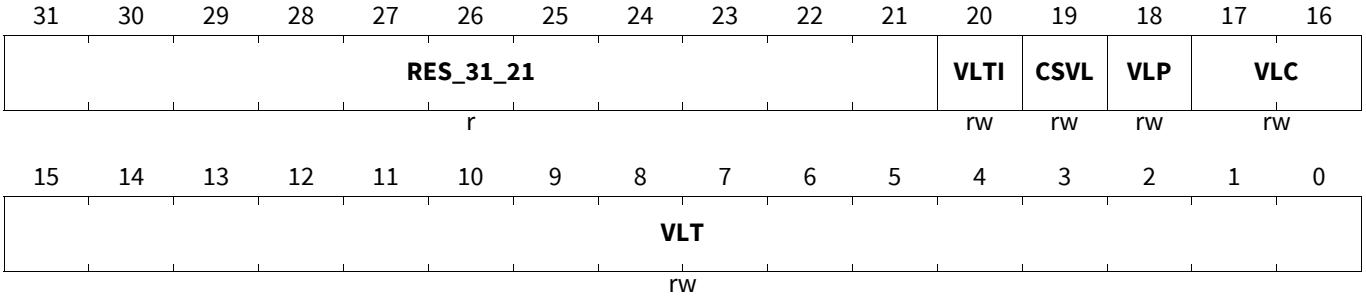
Field	Bits	Type	Description
VLT	15:0	rw	**VLAN Tag for Transmit Packets** This field contains the value of the VLAN tag to be inserted. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and Bits[11:0] are the VID field in the VLAN tag. The following list describes the bits of this field: • Bits[15:13]: User Priority • Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) • Bits[11:0]: VLAN Identifier (VID) field of VLAN tag Value After Reset: 0x0
RES_18_16	18:16	r	**Reserved** Value After Reset: 0x0
CSVL	19	rw	**C-VLAN or S-VLAN** When this bit is set, S-VLAN type (0x88A8) is inserted in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted in the 13th and 14th bytes of transmitted packets. Value After Reset: 0x0
RES_31_20	31:20	r	**Reserved** Value After Reset: 0x0

MAC Inner VLAN Tag Inclusion or Replacement Register

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the Transmit packet. It also contains the inner VLAN tag insertion controls.

MAC_INNER_VLAN_INCL_i (i=0-3)

MAC Inner VLAN Tag Inclusion or Replacement Register(0064_H) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
VLT	15:0	rw	**VLAN Tag for Transmit Packets** This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits[15:13] are the User Priority field, Bit 12 is the CFI/DEI field, and Bits[11:0] are the VID field in the VLAN tag. The following list describes the bits of this field: • Bits[15:13]: User Priority • Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) • Bits[11:0]: VLAN Identifier (VID) field of VLAN tag Value After Reset: 0x0
VLC	17:16	rw	**VLAN Tag Control in Transmit Packets** The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags. The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag. The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8). **Note**: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value. Value After Reset: 0x0 00_B No VLAN tag deletion, insertion, or replacement 01_B VLAN tag deletion 10_B VLAN tag insertion 11_B VLAN tag replacement
VLP	18	rw	**VLAN Priority Control** When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the mti_vlan_ctrl_i control input is used and the VLC field is ignored. Value After Reset: 0x0
CSVL	19	rw	**C-VLAN or S-VLAN** When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets. Value After Reset: 0x0

Field	Bits	Type	Description
VLTI	20	rw	**VLAN Tag Input** When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from: • The Tx descriptor Value After Reset: 0x0
RES_31_21	31:21	r	**Reserved** Value After Reset: 0x0

MAC Queue 0 TX Flow Control Register

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

When the PFCE bit in the MAC_Rx_Flow_Ctrl register is enabled, this register controls the generation of Priority Flow Control (PFC) frames with priorities mapped according to PSRQ0 in the MAC_RxQ_Ctrl2 register.

MAC_Q0_TX_FLOW_CTRL

MAC Queue 0 TX Flow Control Register (0070_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PT															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_8								DZPQ	PLT			RES_3_2		TFE	FCB_BPA
r								rw	rw			r		rw	rw

Field	Bits	Type	Description
FCB_BPA	0	rw	**Flow Control Busy or Backpressure Activate** This bit initiates a Pause packet in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set. **Full-Duplex Mode:** In the full-duplex mode, this bit should be read as 1'b0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1'b1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 1'b0. You should not write to this register until this bit is cleared. **Half--Duplex Mode:** When this bit is set (and TFE bit is set) in the half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically ORed with the mti_flowctrl_i input signal for the backpressure function. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
TFE	1	rw	**Transmit Flow Control Enable** **Full-Duplex Mode:** In the full-duplex mode, when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets. **Half--Duplex Mode:** In the half-duplex mode, when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled. Value After Reset: 0x0
RES_3_2	3:2	r	**Reserved** Value After Reset: 0x0

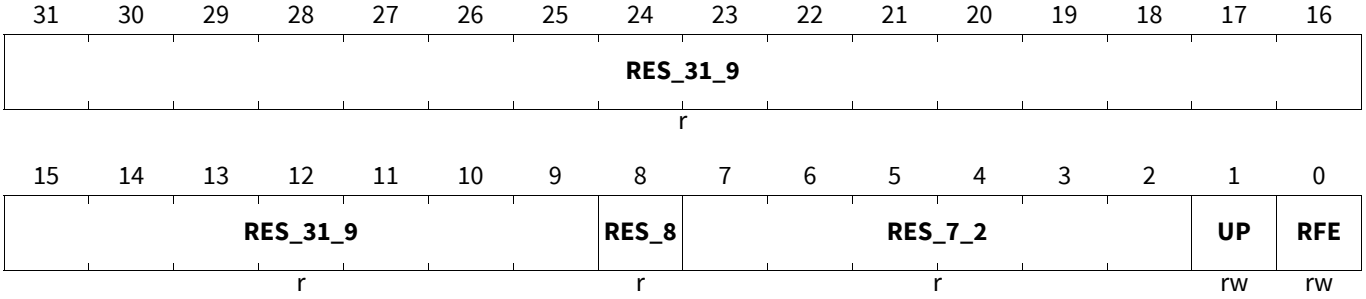
Field	Bits	Type	Description
PLT	6:4	rw	**Pause Low Threshold** This field configures the threshold of the Pause timer at which the input flow control signal mti_flowctrl_i (or sbd_flowctrl_i) is checked for automatic retransmission of the Pause packet. The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256-28) slot times after the first Pause packet is transmitted. The following list provides the threshold values for different values: 110_B-111_B Reserved The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times. Value After Reset: 0x0 000_B Pause Time minus 4 Slot Times (PT -4 slot times) 001_B Pause Time minus 28 Slot Times (PT -28 slot times) 010_B Pause Time minus 36 Slot Times (PT -36 slot times) 011_B Pause Time minus 144 Slot Times (PT -144 slot times) 100_B Pause Time minus 256 Slot Times (PT -256 slot times) 101_B Pause Time minus 512 Slot Times (PT -512 slot times)
DZPQ	7	rw	**Disable Zero-Quanta Pause** When this bit is set, it disables the automatic generation of the zero-quanta Pause packets on de-assertion of the flow-control signal from the FIFO layer (MTL or external sideband flow control signal sbd_flowctrl_i or mti_flowctrl_i). When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled. Value After Reset: 0x0
RES_15_8	15:8	r	**Reserved** Value After Reset: 0x0
PT	31:16	rw	**Pause Time** This field holds the value to be used in the Pause Time field in the Tx control packet. If the Pause Time bits are configured to be double-synchronized to the (G)MII clock domain, consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain. Value After Reset: 0x0

MAC Receive Flow Control Register

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

MAC_RX_FLOW_CTRL

MAC Receive Flow Control Register (0090$_H$) Application Reset Value: 0000 0000$_H$



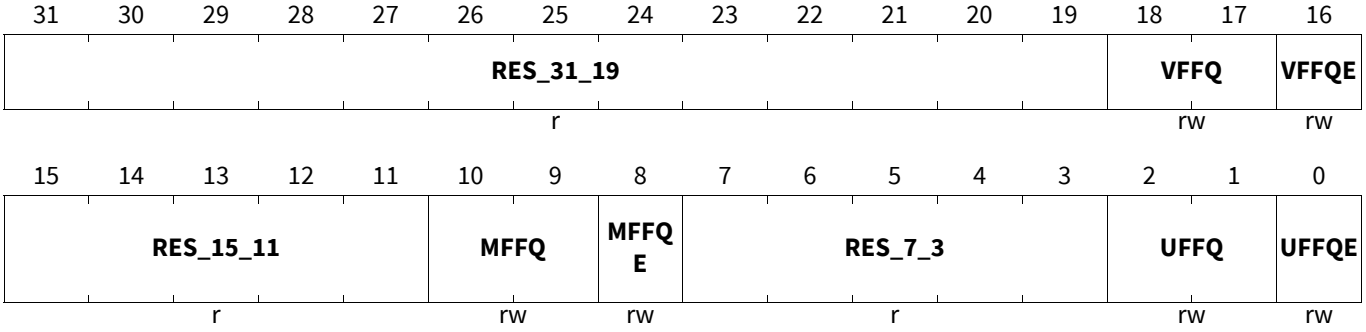
Field	Bits	Type	Description
RFE	0	rw	**Receive Flow Control Enable** When this bit is set and the MAC is operating in full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in half-duplex mode, the decode function of the Pause packet is disabled. When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time. Value After Reset: 0x0
UP	1	rw	**Unicast Pause Packet Detect** A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in MAC_Address0_High and MAC_Address0_Low. When this bit is reset, the MAC only detects Pause packets with unique multicast address. **Note**: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01_80_C2_00_00_01) is as specified in IEEE 802.1 Qbb-2011. Value After Reset: 0x0
RES_7_2	7:2	r	**Reserved** Value After Reset: 0x0
RES_8	8	r	**Reserved for future use.**
RES_31_9	31:9	r	**Reserved** Value After Reset: 0x0

MAC Receive Queue Control 4 register

The Receive Queue Control 4 register controls the routing of unicast and multicast packets that fail the Destination or Source address filter to the Rx queues.

MAC_RXQ_CTRL4

MAC Receive Queue Control 4 register (0094$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
UFFQE	0	rw	**Unicast Address Filter Fail Packets Queuing Enable** When this bit is set, the Unicast packets which fail the Destination or Source address filter is routed to the Rx Queue Number programmed in the UFFQ. When this bit is reset, the Unicast packets which fail the Destination or Source address filter is routed based on other routing options. This bit is valid only when the RA bit of the MAC_Packet_Filter register is set. Value After Reset: 0x0
UFFQ	2:1	rw	**Unicast Address Filter Fail Packets Queue** This field holds the Rx queue number to which the Unicast packets failing the Destination or Source Address filter are routed to. This field is valid only when the UFFQE bit is set. Value After Reset: 0x0
RES_7_3	7:3	r	**Reserved** Value After Reset: 0x0
MFFQE	8	rw	**Multicast Address Filter Fail Packets Queuing Enable** When this bit is set, the Multicast packets which fail the Destination or Source address filter is routed to the Rx Queue Number programmed in the MFFQ. When this bit is reset, the Multicast packets which fail the Destination or Source address filter is routed based on other routing options. This bit is valid only when the RA bit of the MAC_Packet_Filter register is set. Value After Reset: 0x0
MFFQ	10:9	rw	**Multicast Address Filter Fail Packets Queue** This field holds the Rx queue number to which the Multicast packets failing the Destination or Source Address filter are routed to. This field is valid only when the MFFQE bit is set. Value After Reset: 0x0
RES_15_11	15:11	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
VFFQE	16	rw	**VLAN Tag Filter Fail Packets Queuing Enable** When this bit is set, the tagged packets which fail the Destination or Source address filter or fail the VLAN tag filter, are routed to the Rx Queue Number programmed in the VFFQ. When this bit is reset, the tagged packets which fail the Destination or Source address filter or fail the VLAN tag filter are routed based on other routing options. This bit is valid only when the RA bit of the MAC_Packet_Filter register is set.
VFFQ	18:17	rw	**VLAN Tag Filter Fail Packets Queue** This field holds the Rx queue number to which the tagged packets failing the Destination or Source Address filter (and UFFQE/MFFQE not enabled) or failing the VLAN tag filter must be routed to. This field is valid only when the VFFQE bit is set. Value After Reset: 0x0
RES_31_19	31:19	r	**Reserved** Value After Reset: 0x0

MAC Receive Queue Control 0 Register

The Receive Queue Control 0 register controls the queue management in the MAC Receiver.

In multiple Rx queues configuration, all the queues are disabled by default. Enable the Rx queue by programming the corresponding field in this register.

MAC_RXQ_CTRL0

MAC Receive Queue Control 0 Register **(00A0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_8								RXQ3EN		RXQ2EN		RXQ1EN		RXQ0EN	
r								rw		rw		rw		rw	

Field	Bits	Type	Description
RXQ0EN	1:0	rw	**Receive Queue 0 Enable** This field indicates whether Rx Queue 0 is enabled for AV or DCB. Value After Reset: 0x0 00_B Not enabled 01_B Queue 0 enabled for AV 10_B Queue 0 enabled for DCB/Generic 11_B Reserved
RXQ1EN	3:2	rw	**Receive Queue 1 Enable** This field is similar to the RXQ0EN field. Value After Reset: 0x0

Field	Bits	Type	Description
RXQ2EN	5:4	rw	**Receive Queue 2 Enable** This field is similar to the RXQ0EN field. Value After Reset: 0x0
RXQ3EN	7:6	rw	**Receive Queue 3 Enable** This field is similar to the RXQ0EN field. Value After Reset: 0x0
RES_15_8	15:8	r	**Reserved for future use.**
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

MAC Receive Queue Control 1 Register

The Receive Queue Control 1 register controls the routing of multicast, broadcast, AV, DCB, and untagged packets to the Rx queues. This register is present only when you select multiple queues in the Receive path.

MAC_RXQ_CTRL1

MAC Receive Queue Control 1 Register (00A4$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_24								TPQC		TACPQE	MCBCQEN	RES_19	MCBCQ		
r								rw		rw	rw	r	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15	UPQ			RES_11	RES_10_8			RES_7	PTPQ			RES_3	AVCPQ		
r	rw			r	r			r	rw			r	rw		

Field	Bits	Type	Description
AVCPQ	2:0	rw	**AV Untagged Control Packets Queue** This field specifies the Receive queue on which the received AV tagged and untagged control packets are routed: The AV tagged (when TACPQE bit is set) and untagged control packets are routed to Receive queue specified by this field. Value After Reset: 0x0 000_B Receive Queue 0 001_B Receive Queue 1 010_B Receive Queue 2 011_B Receive Queue 3 100_B Receive Queue 4 101_B Receive Queue 5 110_B Receive Queue 6 111_B Receive Queue 7
RES_3	3	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
PTPQ	6:4	rw	**PTP Packets Queue** This field specifies the Rx queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed. When the AV8021ASMEN bit of MAC_Timestamp_Control register is set, only untagged PTP over Ethernet packets are routed on an Rx Queue. If the bit is not set, then based on programming of TPQC field, both tagged and untagged PTPoE packets can be routed to this Rx Queue. Value After Reset: 0x0 000_B Rx Queue 0 001_B Rx Queue 1 010_B Rx Queue 2 011_B Rx Queue 3 100_B Rx Queue 4 101_B Rx Queue 5 110_B Rx Queue 6 111_B Rx Queue 7
RES_7	7	r	**Reserved** Value After Reset: 0x0
RES_10_8	10:8	r	**Reserved for future use.**
RES_11	11	r	**Reserved** Value After Reset: 0x0
UPQ	14:12	rw	**Untagged Packet Queue** This field indicates the Rx Queue to which Untagged Packets are to be routed. Any Rx Queue enabled for Generic/DCB/AV traffic can be used to route the Untagged Packets. LOST SEQUENCE DEFINITION Value After Reset: 0x0 000_B Rx Queue 0 001_B Rx Queue 1 111_B Rx Queue 7
RES_15	15	r	**Reserved** Value After Reset: 0x0
MCBCQ	18:16	rw	**Multicast and Broadcast Queue** This field specifies the Rx Queue onto which Multicast or Broadcast Packets are routed. Any Rx Queue enabled for Generic/DCB/AV traffic can be used to route the Multicast or Broadcast Packets. LOST SEQUENCE DEFINITION Value After Reset: 0x0 000_B Rx Queue 0 001_B Rx Queue 1 111_B Rx Queue 7
RES_19	19	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
MCBCQEN	20	rw	**Multicast and Broadcast Queue Enable** This bit specifies that Multicast or Broadcast packets routing to the Rx Queue is enabled and the Multicast or Broadcast packets must be routed to Rx Queue specified in MCBCQ field. Value After Reset: 0x0
TACPQE	21	rw	**Tagged AV Control Packets Queuing Enable.** When set, the MAC routes the received Tagged AV Control packets to the Rx queue specified by AVCPQ field. When reset, the MAC routes the received Tagged AV Control packets based on the tag priority matching the PSRQ fields in MAC_RxQ_Ctrl2 register. Value After Reset: 0x0
TPQC	23:22	rw	**Tagged PTP over Ethernet Packets Queuing Control - MFFQE** This field controls the routing of the VLAN Tagged PTPoE packets. The following programmable options are allowed. • 2'b00: VLAN Tagged PTPoE packets are routed as generic VLAN Tagged packet (based on PSRQ for only non-AV enabled Rx Queues). • 2'b01: VLAN Tagged PTPoE packets are routed to Rx Queue specified by PTPQ field (That Rx Queue can be enabled for AV or non AV traffic). • 2'b10: VLAN Tagged PTPoE packets are routed to only AV enabled Rx Queues based on PSRQ. • 2'b11: Reserved Value After Reset: 0x0
RES_31_24	31:24	r	**Reserved** Value After Reset: 0x0

MAC Receive Queue Control 2 Register

This register controls the routing of tagged packets based on the USP (user Priority) field of the received packets to the Rx Queues 0 to 3. This register is present when multiple Rx Queues are selected while configuring the core.

MAC_RXQ_CTRL2

MAC Receive Queue Control 2 Register **($00A8_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSRQ3								**PSRQ2**							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSRQ1								**PSRQ0**							
rw								rw							

Field	Bits	Type	Description
PSRQ0	7:0	rw	**Priorities Selected in the Receive Queue 0** This field decides the priorities assigned to Rx Queue 0. All packets with priorities that match the values set in this field are routed to Rx Queue 0. For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx Queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. Value After Reset: 0x0
PSRQ1	15:8	rw	**Priorities Selected in the Receive Queue 1** This field decides the priorities assigned to Rx Queue 1. All packets with priorities that match the values set in this field are routed to Rx Queue 1. For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx Queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. Value After Reset: 0x0
PSRQ2	23:16	rw	**Priorities Selected in the Receive Queue 2** This field decides the priorities assigned to Rx Queue 2. All packets with priorities that match the values set in this field are routed to Rx Queue 2. For example, if PSRQ2[1, 0] are set, packets with USP field equal to 1 or 0 are routed to Rx Queue 2. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. Value After Reset: 0x0
PSRQ3	31:24	rw	**Priorities Selected in the Receive Queue 3** This field decides the priorities assigned to Rx Queue 3. All packets with priorities that match the values set in this field are routed to Rx Queue 3. For example, if PSRQ3[6, 3] are set, packets with USP field equal to 3 or 6 are routed to Rx Queue 3. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues. Value After Reset: 0x0

MAC Interrupt Status Register

The Interrupt Status register contains the status of interrupts.

MAC_INTERRUPT_STATUS

MAC Interrupt Status Register **(00B0$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_19													MDIOIS	RES_17_15	
r													r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_17_15	RXSTSIS	TXSTSIS	TSIS	MMCRXIPIS	MMCTXIS	MMCRXIS	MMCIS	RES_7_6		LPIIS	PMTIS	PHYIS	RES_2_1		RGSMIIIS
r	r	r	r	r	r	r	r	r		r	r	r	r		r

Field	Bits	Type	Description
RGSMIIIS	0	r	**RGMII or SMII Interrupt Status** This bit is set because of any change in value of the Link Status of RGMII or SMII interface (LNKSTS bit in MAC_PHYIF_Control_Status register). This bit is cleared when the MAC_PHYIF_Control_Status register is read (or LNKSTS bit of MAC_PHYIF_Control_Status register is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set). Value After Reset: 0x0
RES_2_1	2:1	r	**Reserved for future use.**
PHYIS	3	r	**PHY Interrupt** This bit is set when rising edge is detected on the phy_intr_i input. This bit is cleared when this register is read (or this bit is writtento 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set).Value After Reset: 0x0
PMTIS	4	r	**PMT Interrupt Status** This bit is set when a Magic packet or Wake-on-LAN packet is received in the powerdown mode (RWKPRCVD and MGKPRCVD bits in MAC_PMT_Control_Status register). This bit is cleared when corresponding interrupt source bit are cleared because of a Read operation to the MAC_PMT_Control_Status register (or corresponding interrupt source bit of MAC_PMT_Control_Status register is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set). Value After Reset: 0x0
LPIIS	5	r	**LPI Interrupt Status** When the Energy Efficient Ethernet feature is enabled, this bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared when the corresponding interrupt source bit of MAC_LPI_Control_Status register is read (or corresponding interrupt source bit of MAC_LPI_Control_Status register is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set). Value After Reset: 0x0
RES_7_6	7:6	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
MMCIS	8	r	**MMC Interrupt Status** This bit is set high when Bit 11, Bit 10, or Bit 9 is set high. This bit is cleared only when all these bits are low. Value After Reset: 0x0
MMCRXIS	9	r	**MMC Receive Interrupt Status** This bit is set high when an interrupt is generated in the MMC Receive Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared. Value After Reset: 0x0
MMCTXIS	10	r	**MMC Transmit Interrupt Status** This bit is set high when an interrupt is generated in the MMC Transmit Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared. Value After Reset: 0x0
MMCRXIPIS	11	r	**MMC Receive Checksum Offload Interrupt Status** This bit is set high when an interrupt is generated in the MMC Receive Checksum Offload Interrupt Register. This bit is cleared when all bits in this interrupt register are cleared. Value After Reset: 0x0
TSIS	12	r	**Timestamp Interrupt Status** If the Timestamp feature is enabled, this bit is set when any of the following conditions is true: • The system time value is equal to or exceeds the value specified in the Target Time High and Low registers. • There is an overflow in the Seconds register. • The Target Time Error occurred, that is, programmed target time already elapsed. In configurations other than EQOS_CORE, when drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the MAC_Tx_Timestamp_Status_Nanoseconds and Mac_TxTimestamp_Status_Seconds registers. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set) in the MAC_Timestamp_Status register. Value After Reset: 0x0

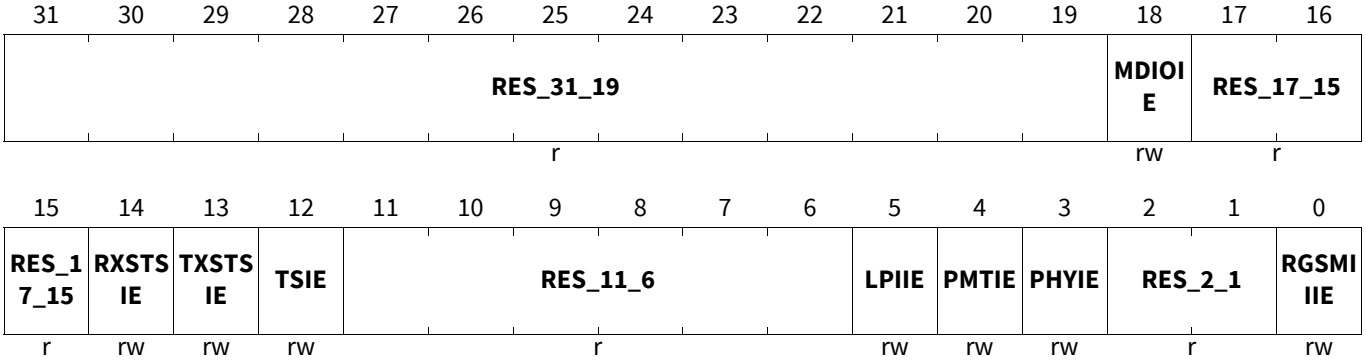
Field	Bits	Type	Description
TXSTSIS	13	r	**Transmit Status Interrupt** This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the MAC_Rx_Tx_Status register: • Excessive Collision (EXCOL) • Late Collision (LCOL) • Excessive Deferral (EXDEF) • Loss of Carrier (LCARR) • No Carrier (NCARR) • Jabber Timeout (TJT) This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set) in the MAC_Rx_Tx_Status register. Value After Reset: 0x0
RXSTSIS	14	r	**Receive Status Interrupt** This bit indicates the status of received packets. This bit is set when the RWT bit is set in the MAC_Rx_Tx_Status register. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of MAC_CSR_SW_Ctrl register is set) in the MAC_Rx_Tx_Status register. Value After Reset: 0x0
RES_17_15	17:15	r	**Reserved for future use.**
MDIOIS	18	r	**MDIO Interrupt Status** This bit indicates an interrupt event after the completion of MDIO operation. To reset this bit, the application has to read this bit/Write 1 to this bit when RCWE bit of MAC_CSR_SW_Ctrl register is set.Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
RES_31_19	31:19	r	**Reserved** Value After Reset: 0x0

MAC Interrupt Enable Register

The Interrupt Enable register contains the masks for generating the interrupts.

MAC_INTERRUPT_ENABLE

MAC Interrupt Enable Register **($00B4_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RGSMIIIE	0	rw	**RGMII or SMII Interrupt Enable** When this bit is set, it enables the assertion of the interrupt signal because of the setting of RGSMIIIS bit in MAC_Interrupt_Status register. Value After Reset: 0x0
RES_2_1	2:1	r	**Reserved for future use.**
PHYIE	3	rw	**PHY Interrupt Enable** When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in MAC_Interrupt_Status register. Value After Reset: 0x0
PMTIE	4	rw	**PMT Interrupt Enable** When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in MAC_Interrupt_Status register. Value After Reset: 0x0
LPIIE	5	rw	**LPI Interrupt Enable** When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in MAC_Interrupt_Status register. Value After Reset: 0x0
RES_11_6	11:6	r	**Reserved** Value After Reset: 0x0
TSIE	12	rw	**Timestamp Interrupt Enable** When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in MAC_Interrupt_Status register. Value After Reset: 0x0
TXSTSIE	13	rw	**Transmit Status Interrupt Enable** When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the MAC_Interrupt_Status register. Value After Reset: 0x0

Field	Bits	Type	Description
RXSTSIE	14	rw	**Receive Status Interrupt Enable** When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the MAC_Interrupt_Status register. Value After Reset: 0x0
RES_17_15	17:15	r	**Reserved for future use.**
MDIOIE	18	rw	**MDIO Interrupt Enable** MDIO Interrupt Enable: When this bit is set, it enables the assertion of the interrupt when MDIOIS field is set in the MAC_Interrupt_Status register. Value After Reset: 0x0
RES_31_19	31:19	r	**Reserved** Value After Reset: 0x0

MAC Receive Transmit Status Register

The Receive Transmit Status register contains the Receive and Transmit Error status.

MAC_RX_TX_STATUS

MAC Receive Transmit Status Register (00B8$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_9															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_31_9							RWT	RES_7_6		EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
r							r	r		r	r	r	r	r	r

Field	Bits	Type	Description
TJT	0	r	**Transmit Jabber Timeout** This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the MAC_Configuration register. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the MAC_Configuration register. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
NCARR	1	r	**No Carrier** When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
LCARR	2	r	**Loss of Carrier** When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the loss of carrier occurred during packet transmission, that is, the phy_crs_i signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
EXDEF	3	r	**Excessive Deferral** When the DTXSTS bit is set in the MTL_Operation_Mode register and the DC bit is set in the MAC_Configuration register, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 in 1000/2500 Mbps mode or when Jumbo packet is enabled). Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
LCOL	4	r	**Late Collision** When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode; 512 bytes including Preamble and Carrier Extension in GMII mode). This bit is not valid if the Underflow error occurs. This bit is reserved in the EQOS-CORE configurations and also in configurations with full-duplex mode. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
EXCOL	5	r	**Excessive Collisions** When the DTXSTS bit is set in the MTL_Operation_Mode register, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the MAC_Configuration register, this bit is set after the first collision and the packet transmission is aborted. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
RES_7_6	7:6	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
RWT	8	r	**Receive Watchdog Timeout** This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the MAC_Configuration register. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the MAC_Configuration register. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
RES_31_9	31:9	r	**Reserved** Value After Reset: 0x0

MAC PMT Control and Status Register

The PMT Control and Status Register.

MAC_PMT_CONTROL_STATUS

MAC PMT Control and Status Register (00C0$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWKFILTRST	RES_30_29		RWKPTR					RES_23_11							
rw	r		r					r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_23_11					RWKPFE	GLBLUCAST	RES_8_7		RWKPRCVD	MGKPRCVD	RES_4_3		RWKPKTEN	MGKPKTEN	PWRDWN
r					rw	rw	r		r	r	r		rw	rw	rw

Field	Bits	Type	Description
PWRDWN	0	rw	**Power Down** When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote Wake-Up Packet Enable bit is set high. **Note**: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0

Field	Bits	Type	Description
MGKPKTEN	1	rw	**Magic Packet Enable** When this bit is set, a power management event is generated when the MAC receives a magic packet. Value After Reset: 0x0
RWKPKTEN	2	rw	**Remote Wake-Up Packet Enable** When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet. Value After Reset: 0x0
RES_4_3	4:3	r	**Reserved** Value After Reset: 0x0
MGKPRCVD	5	r	**Magic Packet Received** When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
RWKPRCVD	6	r	**Remote Wake-Up Packet Received** When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read. Access restriction applies. Clears on read (or write of 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
RES_8_7	8:7	r	**Reserved** Value After Reset: 0x0
GLBLUCAST	9	rw	**Global Unicast** When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet. Value After Reset: 0x0

Field	Bits	Type	Description
RWKPFE	10	rw	**Remote Wake-up Packet Forwarding Enable** When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected Wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet. The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high. Note: If Magic Packet Enable and Wake-Up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
RES_23_11	23:11	r	**Reserved** Value After Reset: 0x0
RWKPTR	28:24	r	**Remote Wake-up FIFO Pointer** This field gives the current value (0 to 7) of the Remote Wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote Wake-up Packet Filter Register are transferred to the clk_rx_i domain when a Write occurs to that register. Value After Reset: 0x0
RES_30_29	30:29	r	**Reserved** Value After Reset: 0x0
RWKFILTRST	31	rw	**Remote Wake-Up Packet Filter Register Pointer Reset** When this bit is set, the remote wake-up packet filter register pointer is reset to 3'b000. It is automatically cleared after 1 clock cycle. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0

MAC Wake-up Packet Filter Register

The wkuppktfilter_reg register at address 0C4H loads the Wake-up Packet Filter register.

To load values in a Wake-up Packet Filter register, the entire register (wkuppktfilter_reg) must be written. The wkuppktfilter_reg register is loaded by sequentially loading the eight, sixteen or thirty two register values in address (0C4H) for wkuppktfilter_reg0, wkuppktfilter_reg1, - wkuppktfilter_reg31, respectively. The wkuppktfilter_reg register is read in a similar way. The DWC_ether_qos updates the wkuppktfilter_reg register current pointer value in Bits[26:24] of MAC_PMT_Control_Status register.

Filter i Byte Mask: The filter i byte mask register defines the bytes of the packet that are examined by filter i (0, 1, 2, 3, …,15) to determine whether or not a packet is a wake-up packet.

The MSB (31st bit) must be zero.

Bit j[30:0] is the byte mask.

If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter i Offset + j of the incoming packet; otherwise Filter i Offset + j is ignored.

Filter i Command: The 4-bit filter i command controls the filter i operation.

Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet.

Bit 2 (Inverse Mode), when set, reverses the logic of the CRC16 hash function signal, to reject a packet with matching CRC_16 value.

Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as “Pattern 1 AND NOT Pattern 2".

Bit 1 (And_Previous) implements the Boolean logic. When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set.

Bit 0 is the enable for filter i. If Bit 0 is not set, filter i is disabled.

Filter i Offset: This filter i offset register defines the offset (within the packet) from which the filter i examines the packets.

This 8-bit pattern-offset is the offset for the filter i first byte to be examined.

The minimum allowed offset is 12, which refers to the 13th byte of the packet.

The offset value 0 refers to the first byte of the packet.

Filter i CRC-16: This filter i CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programmed to the wake-up filter register block.

The 16-bit CRC calculation uses the following polynomial:

G(x) = x^16 + x^15 + x^2 + 1

Each mask, used in the hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following:

32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation.

8-bit Offset Pointer: Specifies the byte to start the CRC16 computation.

The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.

Note: If you are accessing these registers in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when CPU accesses Lane 3 (or Lane 0 in big-endian mode).

Note: After a write operation, there should not be any further writes to the same location until the first write is updated. Otherwise, the second write operation will not get updated to the destination clock domain. Thus the delay between two writes to the same register location should be at least 6 cycles of the destination clock (PHY receive clock or PHY transmit clock).

Notes on And_Previous bit setting

The And_Previous bit setting is applicable within a set of 4 filters.

Setting of And_Previous bit of filter that is not enabled has no effect. In other words, setting And_Previous bit of lowest number filter in the set of 4 filters has no effect. For example, setting of And_Previous bit of Filter 0 has no effect.

If And_Previous bit is set for filter to form AND chained filter, the AND chain breaks at the point any filter is not enabled. For example:

If Filter 2 And_Previous bit is set (bit 1 of Filter 2 command is set) but Filter 1 is not enabled (bit 0 of in Filter 1 command is reset), then only Filter 2 result is considered. If Filter 2 And_Previous bit is set (bit 1 of Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 of Filter 3 command is set), but Filter 1 is not enabled (bit 0 of in Filter 1 command is reset), then only Filter 2 result ANDed with Filter 3 result is considered.

If Filter 2 And_Previous bit is set (bit 1 of Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 of Filter 3 command is set), but Filter 2 is not enabled (bit 0 of in Filter 2 command is reset), then since setting of Filter 2 And_Previous bit has no effect only Filter 1 result ORed with Filter 3 result is considered. - If filters chained by And_Previous bit setting have complementary programming, then a frame may never pass the AND chained filter. For example, if Filter 2 And_Previous bit is set (bit 1 of Filter 2 command is set), Filter 1 Address_Type bit is set (bit 3 of Filter 1 command is set) indicating multicast detection and Filter 2 Address_Type bit is reset (bit 3 of Filter 2 command is reset) indicating unicast detection or vice versa, a remote wakeup frame does not pass the AND chained filter as a remote wakeup frame cannot be of both unicast and multicast address type.

MAC_RWK_PACKET_FILTER

MAC Wake-up Packet Filter Register **($00C4_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUPFRMFTR															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WKUPFRMFTR															
rw															

Field	Bits	Type	Description
WKUPFRMFTR	31:0	rw	**RWK Packet Filter** This field contains the various controls of RWK Packet filter. Value After Reset: 0x0

MAC Wake-up Filter Command 0 Register

This register controls the filter operation.

RWK_FILTER_COMMAND_0

MAC Wake-up Filter Command 0 Register **($00C4_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD3				Filter3_Command				RSVD2				Filter2_Command			
r				rw				r				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD1				Filter1_Command				RSVD0				Filter0_Command			
r				rw				r				rw			

Field	Bits	Type	Description
Filter0_Command	3:0	rw	**Filter0_Command** The 4-bit filter command controls the filter operation. • Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. • Bit 2 (Inverse Mode), when set, reverses the logic of the CRC16 hash function signal, to reject a packet with matching CRC_16 value. • Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as “Pattern 1 AND NOT Pattern 2". • Bit 1 (And_Previous) implements the Boolean logic. When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. • Bit 0 is the enable for filter. If Bit 0 is not set, filter is disabled. Value After Reset: 0x0
RSVD0	7:4	r	**Reserved** Value After Reset: 0x0
Filter1_Command	11:8	rw	**Filter1_Command** The 4-bit filter command controls the filter operation. • Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. • Bit 2 (Inverse Mode), when set, reverses the logic of the CRC16 hash function signal, to reject a packet with matching CRC_16 value. • Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as “Pattern 1 AND NOT Pattern 2". • Bit 1 (And_Previous) implements the Boolean logic. When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. • Bit 0 is the enable for filter. If Bit 0 is not set, filter is disabled. Value After Reset: 0x0
RSVD1	15:12	r	**Reserved** Value After Reset: 0x0

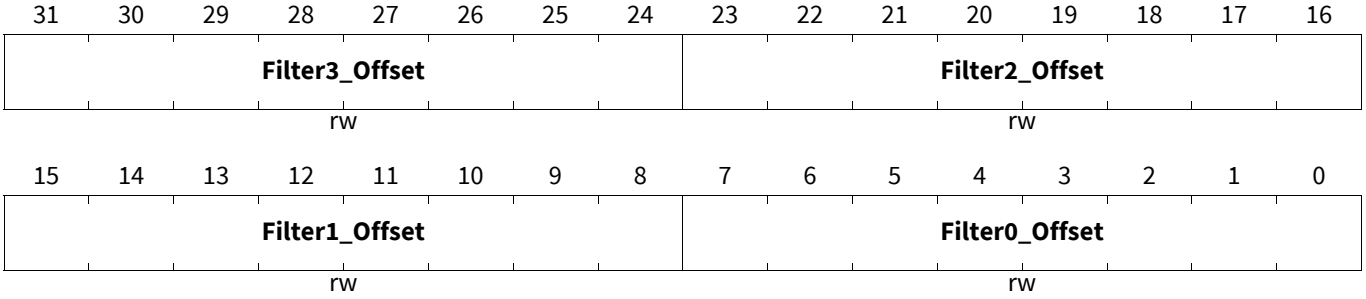
Field	Bits	Type	Description
Filter2_Command	19:16	rw	**Filter2_Command** The 4-bit filter command controls the filter operation. • Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. • Bit 2 (Inverse Mode), when set, reverses the logic of the CRC16 hash function signal, to reject a packet with matching CRC_16 value. • Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as “Pattern 1 AND NOT Pattern 2". • Bit 1 (And_Previous) implements the Boolean logic. When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. • Bit 0 is the enable for filter. If Bit 0 is not set, filter is disabled. Value After Reset: 0x0
RSVD2	23:20	r	**Reserved** Value After Reset: 0x0
Filter3_Command	27:24	rw	**Filter3_Command** The 4-bit filter command controls the filter operation. • Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. • Bit 2 (Inverse Mode), when set, reverses the logic of the CRC16 hash function signal, to reject a packet with matching CRC_16 value. • Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as “Pattern 1 AND NOT Pattern 2". • Bit 1 (And_Previous) implements the Boolean logic. When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. • Bit 0 is the enable for filter. If Bit 0 is not set, filter is disabled. Value After Reset: 0x0
RSVD3	31:28	r	**Reserved** Value After Reset: 0x0

MAC Wake-up Filter Offset 0 Register

This register contains the Remote Wakeup Filter Offset Values.

RWK_FILTER_OFFSET_0

MAC Wake-up Filter Offset 0 Register (00C4$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
Filter0_Offset	7:0	rw	**Filter0_Offset** This filter offset defines the offset (within the packet) from which the filter examines the packets. • This 8-bit pattern-offset is the offset for the filter first byte to be examined. • The minimum allowed offset is 12, which refers to the 13th byte of the packet. • The offset value 0 refers to the first byte of the packet. Value After Reset: 0x0
Filter1_Offset	15:8	rw	**Filter1_Offset** This filter offset defines the offset (within the packet) from which the filter examines the packets. • This 8-bit pattern-offset is the offset for the filter first byte to be examined. • The minimum allowed offset is 12, which refers to the 13th byte of the packet. • The offset value 0 refers to the first byte of the packet. Value After Reset: 0x0
Filter2_Offset	23:16	rw	**Filter2_Offset** This filter offset defines the offset (within the packet) from which the filter examines the packets. • This 8-bit pattern-offset is the offset for the filter first byte to be examined. • The minimum allowed offset is 12, which refers to the 13th byte of the packet. • The offset value 0 refers to the first byte of the packet. Value After Reset: 0x0

Field	Bits	Type	Description
Filter3_Offset	31:24	rw	**Filter3_Offset** This filter offset defines the offset (within the packet) from which the filter examines the packets. • This 8-bit pattern-offset is the offset for the filter first byte to be examined. • The minimum allowed offset is 12, which refers to the 13th byte of the packet. • The offset value 0 refers to the first byte of the packet. Value After Reset: 0x0

MAC Wake-up Filter CRC i Register

This register contains the CRC-16 to be matched.

RWK_FILTER_CRC_i (i=0-1)

MAC Wake-up Filter CRC i Register **($00C4_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Filter1_CRC															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Filter0_CRC															
rw															

Field	Bits	Type	Description
Filter0_CRC	15:0	rw	**Filter0_CRC** This filter CRC-16 contains the CRC_16 value of the pattern. • The 16-bit CRC calculation uses the following polynomial: G(x) = x^16 + x^15 + x^2 + 1 Value After Reset: 0x0
Filter1_CRC	31:16	rw	**Filter1_CRC** This filter CRC-16 contains the CRC_16 value of the pattern. • The 16-bit CRC calculation uses the following polynomial: G(x) = x^16 + x^15 + x^2 + 1 Value After Reset: 0x0

MAC Wake-up i Filter Byte Mask register

This register contains the Remote Wakeup Filter Byte Mask.

RWK_FILTER_BYTE_MASK_i (i=0-3)

MAC Wake-up i Filter Byte Mask register (00C4$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Filteri_Byte_Mask															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Filteri_Byte_Mask															
rw															

Field	Bits	Type	Description
Filteri_Byte_Mask	31:0	rw	**Filteri 32-bit Mask** Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation. Value After Reset: 0x0

MAC LPI Control and Status Register

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

MAC_LPI_CONTROL_STATUS

MAC LPI Control and Status Register (00D0$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_22										LPITCSE	LPIATE	LPITXA	PLSEN	PLS	LPIEN
r										rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_10						RLPIST	TLPIST	RES_7_4				RLPIEX	RLPIEN	TLPIEX	TLPIEN
r						r	r	r				r	r	r	r

Field	Bits	Type	Description
TLPIEN	0	r	**Transmit LPI Entry** When this bit is set, it indicates that the MAC Transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Value After Reset: 0x0

Field	Bits	Type	Description
TLPIEX	1	r	**Transmit LPI Exit** When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Value After Reset: 0x0
RLPIEN	2	r	**Receive LPI Entry** When this bit is set, it indicates that the MAC Receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). **Note**: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock. Value After Reset: 0x0
RLPIEX	3	r	**Receive LPI Exit** When this bit is set, it indicates that the MAC Receiver has stopped receiving the LPI pattern on the GMII or MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). **Note**: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock. Value After Reset: 0x0
RES_7_4	7:4	r	**Reserved** Value After Reset: 0x0
TLPIST	8	r	**Transmit LPI State** When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the GMII or MII interface. Value After Reset: 0x0
RLPIST	9	r	**Receive LPI State** When this bit is set, it indicates that the MAC is receiving the LPI pattern on the GMII or MII interface. Value After Reset: 0x0
RES_15_10	15:10	r	**Reserved** Value After Reset: 0x0
LPIEN	16	rw	**LPI Enable** When this bit is set, it instructs the MAC Transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission. This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission. Value After Reset: 0x0

Field	Bits	Type	Description
PLS	17	rw	**PHY Link Status** This bit indicates the link status of the PHY. The MAC Transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER. When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down. Value After Reset: 0x0
PLSEN	18	rw	**PHY Link Status Enable** This bit enables the link status received on the RGMII, SGMII, or SMII Receive paths to be used for activating the LPI LS TIMER. When this bit is set, the MAC uses the link-status bits of the MAC_PHYIF_Control_Status register and the PLS bit for the LPI LS Timer trigger. When this bit is reset, the MAC ignores the link-status bits of the MAC_PHYIF_Control_Status register and takes only the PLS bit. Value After Reset: 0x0
LPITXA	19	rw	**LPI Tx Automate** This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side. This bit is not functional in the EQOS-CORE configurations in which the Tx clock gating is done during the LPI mode. If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of MTL_TxQ0_Operation_Mode register, when the MAC is in the LPI mode, it exits the LPI mode. When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode. Value After Reset: 0x0
LPIATE	20	rw	**LPI Timer Enable** This bit controls the automatic entry of the MAC Transmitter into and exit out of the LPI state. When LPIATE, LPITXA and LPIEN bits are set, the MAC Transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the MAC_LPI_Entry_Timer register. After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the Transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again. When LPIATE is 0, the LPI Auto timer is disabled and MAC Transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions. Value After Reset: 0x0

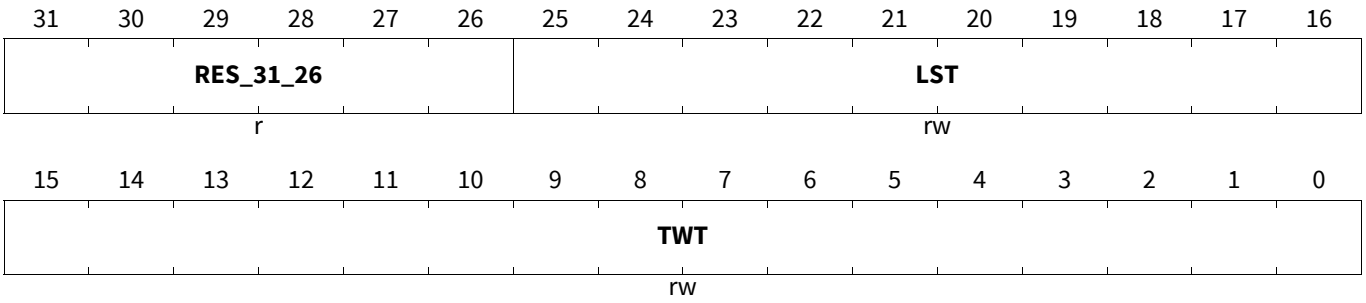
Field	Bits	Type	Description
LPITCSE	21	rw	**LPI Tx Clock Stop Enable** When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode. If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed. Value After Reset: 0x0
RES_31_22	31:22	r	**Reserved** Value After Reset: 0x0

MAC LPI Timers Control Register

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

MAC_LPI_TIMERS_CONTROL

MAC LPI Timers Control Register **(00D4$_H$)** **Application Reset Value: 03E8 0000$_H$**

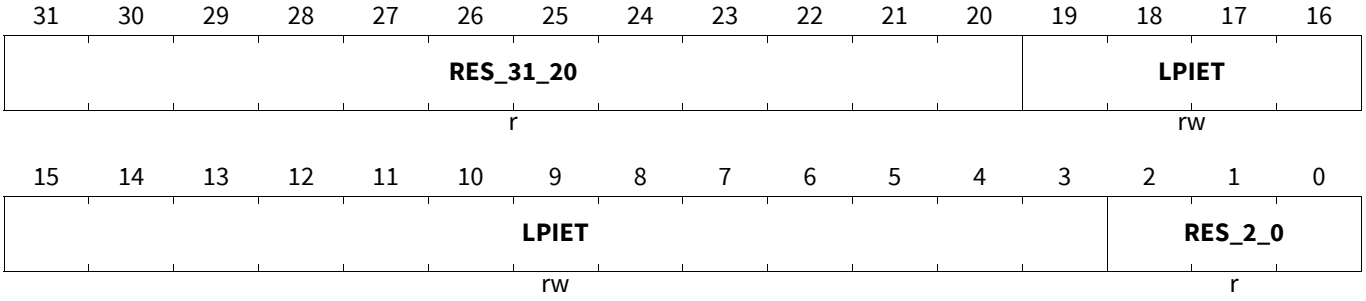


Field	Bits	Type	Description
TWT	15:0	rw	**LPI TW Timer** This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer. Value After Reset: 0x0
LST	25:16	rw	**LPI LS Timer** This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard. Value After Reset 0x3e8
RES_31_26	31:26	r	**Reserved** Value After Reset: 0x0

MAC LPI Entry Timer Register

This register controls the Tx LPI entry timer. This counter is enabled only when bit[20](LPITE) bit of MAC_LPI_Control_Status is set to 1. This register is present only when you select the Energy Efficient Ethernet feature in coreConsultant.

MAC_LPI_ENTRY_TIMER
MAC LPI Entry Timer Register **(00D8$_H$)** **Application Reset Value: 0000 0000$_H$**

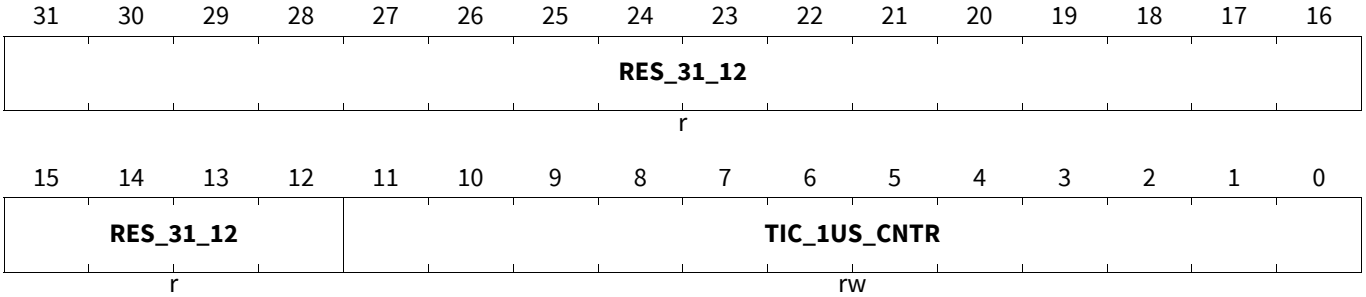


Field	Bits	Type	Description
RES_2_0	2:0	r	**Reserved** Value After Reset: 0x0
LPIET	19:3	rw	**LPI Entry Timer** This field specifies the time in microseconds the MAC waits to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1. Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 micro-seconds. Value After Reset: 0x0
RES_31_20	31:20	r	**Reserved** Value After Reset: 0x0

MAC One Microsecond Tic Counter Register

This register controls the generation of the Reference time (1 microsecond tic) for all the LPI timers. This timer has to be programmed by the software initially.

MAC_1US_TIC_COUNTER
MAC One Microsecond Tic Counter Register **(00DC$_H$)** **Application Reset Value: 0000 0063$_H$**



Field	Bits	Type	Description
TIC_1US_CNTR	11:0	rw	**1US TIC Counter** The application must program this counter so that the number of clock cycles of CSR clock is 1us. (Subtract 1 from the value before programming). For example if the CSR clock is 100MHz then this field needs to be programmed to value 100 - 1 = 99 (which is 0x63). This is required to generate the 1US events that are used to update some of the EEE related counters. Value After Reset: 0x63
RES_31_12	31:12	r	**Reserved** Value After Reset: 0x0

MAC PHY Interface Control and Status Register

The PHY Interface Control and Status register indicates the status signals received by the SGMII, RGMII, or SMII interface (selected at reset) from the PHY. This register is optional. It is present only when the MAC is configured for the SGMII, RGMII, or SMII PHY interface.

MAC_PHYIF_CONTROL_STATUS

MAC PHY Interface Control and Status Register (00F8$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		RES_31_20										LNKSTS	LNKSPEED		LNKMOD
		r										r	r		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_5											RES_4	RES_3	RES_2	LUD	TC
r											r	r	r	rw	rw

Field	Bits	Type	Description
TC	0	rw	**Transmit Configuration in RGMII, SGMII, or SMII** When set, this bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in the RGMII, SMII, or SGMII port. When this bit is reset, no such information is driven to the PHY. The details of this feature are provided in the following sections: • “Reduced Gigabit Media Independent Interface” • “Serial Media Independent Interface” • “Serial Gigabit Media Independent Interface” Value After Reset: 0x0

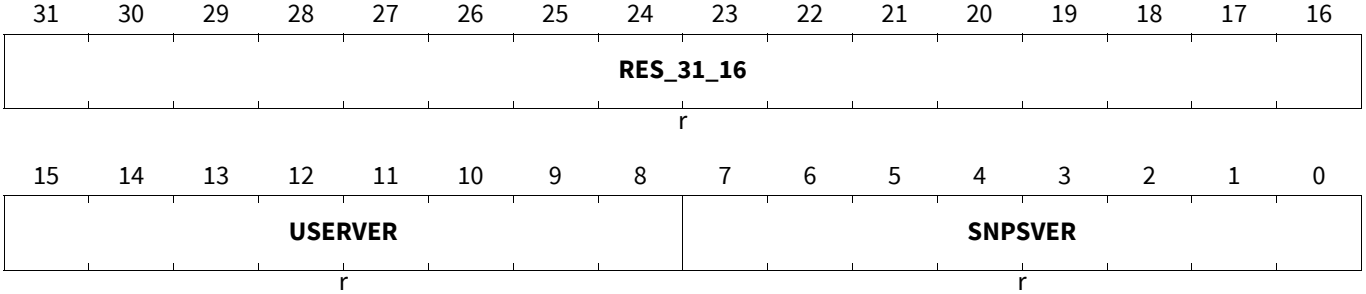
Field	Bits	Type	Description
LUD	1	rw	**Link Up or Down** This bit indicates whether the link is up or down during transmission of configuration in the RGMII, SGMII, or SMII interface: Value After Reset: 0x0 0_B Link Down 1_B Link Up
RES_2	2	r	**Reserved for future use.**
RES_3	3	r	**Reserved** Value After Reset: 0x0
RES_4	4	r	**Reserved for future use.**
RES_15_5	15:5	r	**Reserved** Value After Reset: 0x0
LNKMOD	16	r	**Link Mode** This bit indicates the current mode of operation of the link: Value After Reset: 0x0 0_B Half-duplex mode 1_B Full-duplex mode
LNKSPEED	18:17	r	**Link Speed** This bit indicates the current speed of the link: Value After Reset: 0x0 00_B 2.5 MHz 01_B 25 MHz 10_B 125 MHz
LNKSTS	19	r	**Link Status** This bit indicates whether the link is up (1'b1) or down (1'b0). Value After Reset: 0x0
RES_31_20	29:20	r	**Reserved** Value After Reset: 0x0

MAC Version Register

The version register identifies the version of the DWC_ether_qos. This register contains two bytes: one that Synopsys uses to identify the core release number, and the other that you set while configuring the core.

MAC_VERSION

MAC Version Register **(0110_H)** **Application Reset Value: $0000\ 1050_H$**

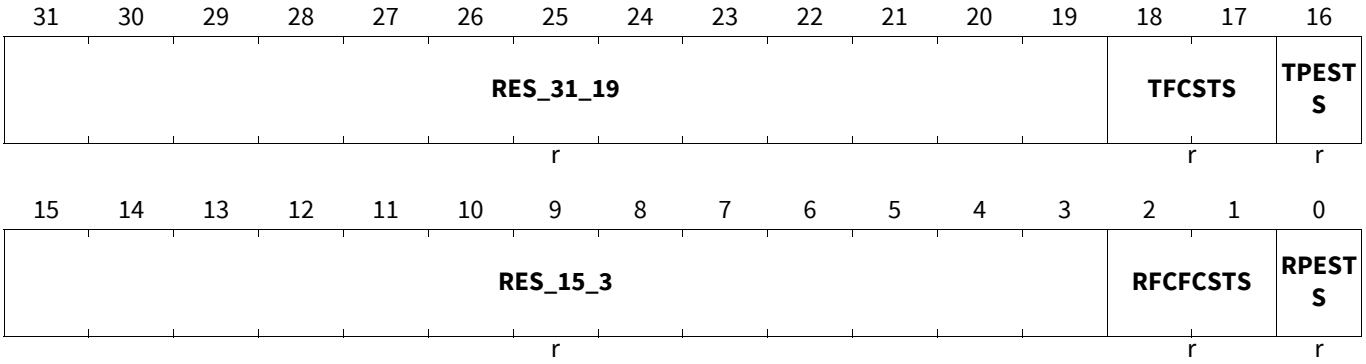


Field	Bits	Type	Description
SNPSVER	7:0	r	**Synopsys-defined Version** IP Version
USERVER	15:8	r	**User-defined Version (configured with coreConsultant)** Value After Reset: 0x10
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

MAC Debug Register

The Debug register provides the debug status of various MAC blocks.

MAC_DEBUG
MAC Debug Register **(0114_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RPESTS	0	r	**MAC GMII or MII Receive Protocol Engine Status** When this bit is set, it indicates that the MAC GMII or MII receive protocol engine is actively receiving data, and it is not in the Idle state. Value After Reset: 0x0
RFCFCSTS	2:1	r	**MAC Receive Packet Controller FIFO Status** When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module. Value After Reset: 0x0
RES_15_3	15:3	r	**Reserved** Value After Reset: 0x0
TPESTS	16	r	**MAC GMII or MII Transmit Protocol Engine Status** When this bit is set, it indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data, and it is not in the Idle state. Value After Reset: 0x0

Field	Bits	Type	Description
TFCSTS	18:17	r	**MAC Transmit Packet Controller Status** This field indicates the state of the MAC Transmit Packet Controller module: Status of the previous packet OR IPG or backoff period to be over Value After Reset: 0x0 00_B Idle state 01_B Waiting for one of the following: 10_B Generating and transmitting a Pause control packet (in full-duplex mode) 11_B Transferring input packet for transmission
RES_31_19	31:19	r	**Reserved** Value After Reset: 0x0

MAC Hardware Feature Register 0

This register indicates the presence of first set of the optional features or functions of the DWC_ether_qos. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

All bits are set or reset according to the features selected while configuring the core in coreConsultant.

MAC_HW_FEATURE0

MAC Hardware Feature Register 0 **($011C_H$)** **Application Reset Value: $0A7D\ 71F7_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31	ACTPHYSEL			SAVLANINS	TSSTSSEL		MACADR64SEL	MACADR32SEL	ADDMACADRSEL					RES_17	RXCOESEL
r	r			r	r		r	r	r					r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15	TXCOESEL	EEESEL	TSSEL	RES_11_10		ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
MIISEL	0	r	**10 or 100 Mbps Support** This bit is set to 1 when 10/100 Mbps is selected as the Mode of Operation Value After Reset: 0x1
GMIISEL	1	r	**1000 Mbps Support** This bit is set to 1 when 1000 Mbps is selected as the Mode of Operation Value After Reset: 0x1
HDSEL	2	r	**Half-duplex Support** This bit is set to 1 when the half-duplex mode is selected Value After Reset: 0x1

Field	Bits	Type	Description
PCSSEL	3	r	**PCS Registers (TBI, SGMII, or RTBI PHY interface)** This bit is set to 1 when the TBI, SGMII, or RTBI PHY interface option is selected Value After Reset: 0x0
VLHASH	4	r	**VLAN Hash Filter Selected** This bit is set to 1 when the Enable VLAN Hash Table Based Filtering option is selected Value After Reset: 0x1
SMASEL	5	r	**SMA (MDIO) Interface** This bit is set to 1 when the Enable Station Management (MDIO Interface) option is selected Value After Reset: 0x1
RWKSEL	6	r	**PMT Remote Wake-up Packet Enable** This bit is set to 1 when the Enable Remote Wake-Up Packet Detection option is selected Value After Reset: 0x1
MGKSEL	7	r	**PMT Magic Packet Enable** This bit is set to 1 when the Enable Magic Packet Detection option is selected Value After Reset: 0x1
MMCSEL	8	r	**RMON Module Enable** This bit is set to 1 when the Enable MAC Management Counters (MMC) option is selected Value After Reset: 0x1
ARPOFFSEL	9	r	**ARP Offload Enabled** This bit is set to 1 when the Enable IPv4 ARP Offload option is selected Value After Reset: 0x0
RES_11_10	11:10	r	**Reserved** Value After Reset: 0x0
TSSEL	12	r	**IEEE 1588-2008 Timestamp Enabled** This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected Value After Reset: 0x1
EEESEL	13	r	**Energy Efficient Ethernet Enabled** This bit is set to 1 when the Enable Energy Efficient Ethernet (EEE) option is selected Value After Reset: 0x1
TXCOESEL	14	r	**Transmit Checksum Offload Enabled** This bit is set to 1 when the Enable Transmit TCP/IP Checksum Insertion option is selected Value After Reset: 0x1
RES_15	15	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
RXCOESEL	16	r	**Receive Checksum Offload Enabled** This bit is set to 1 when the Enable Receive TCP/IP Checksum Check option is selected Value After Reset: 0x1
RES_17	17	r	**Reserved** Value After Reset: 0x0
ADDMACADRSEL	22:18	r	**MAC Addresses 1-31 Selected** This bit is set to 1 when the Enable Additional 1-31 MAC Address Registers option is selected Value After Reset: 0x1f
MACADR32SEL	23	r	**MAC Addresses 32-63 Selected** This bit is set to 1 when the Enable Additional 32 MAC Address Registers (32-63) option is selected Value After Reset: 0x0
MACADR64SEL	24	r	**MAC Addresses 64-127 Selected** This bit is set to 1 when the Enable Additional 64 MAC Address Registers (64-127) option is selected Value After Reset: 0x0
TSSTSSEL	26:25	r	**Timestamp System Time Source** This bit indicates the source of the Timestamp system time: This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected Value After Reset: 0x1 00_B Reserved 01_B Internal 10_B External 11_B Both
SAVLANINS	27	r	**Source Address or VLAN Insertion Enable** This bit is set to 1 when the Enable SA and VLAN Insertion on Tx option is selected Value After Reset: 0x1
ACTPHYSEL	30:28	r	**Active PHY Selected** When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion: All Others: Reserved Value After Reset: 0x0 000_B GMII or MII 001_B RGMII 010_B SGMII 011_B TBI 100_B RMII 101_B RTBI 110_B SMII 111_B RevMII
RES_31	31	r	**Reserved** Value After Reset: 0x0

MAC Hardware Feature Register 1

This register indicates the presence of second set of the optional features or functions of the DWC_ether_qos. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

All bits are set or reset according to the features selected while configuring the core in coreConsultant.

MAC_HW_FEATURE1

MAC Hardware Feature Register 1 **(0120_H)** **Application Reset Value: $0092\ 2966_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31	L3L4FNUM				RES_26	HASHTBLSZ		POUOST	RES_22	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN
r	r				r	r		r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR64		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE					SPRAM	RXFIFOSIZE				
r		r	r	r	r					r	r				

Field	Bits	Type	Description
RXFIFOSIZE	4:0	r	**MTL Receive FIFO Size** This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7: 01100_B-11111_B Reserved Value After Reset: 0x6 00_H 128 bytes 01_H 256 bytes 02_H 512 bytes 03_H 1,024 bytes 04_H 2,048 bytes 05_H 4,096 bytes 06_H 8,192 bytes 07_H 16,384 bytes 08_H 32,767 bytes 09_H 64 KB $0A_H$ 128 KB $0B_H$ 256 KB
SPRAM	5	r	**Single Port RAM Enable** This bit is set to 1 when the Use single port RAM Feature is selected. Value After Reset: 0x1

Field	Bits	Type	Description
TXFIFOSIZE	10:6	r	**MTL Transmit FIFO Size** This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7: 01011_B-11111_B Reserved Value After Reset: 0x5 00_H 128 bytes 01_H 256 bytes 02_H 512 bytes 03_H 1,024 bytes 04_H 2,048 bytes 05_H 4,096 bytes 06_H 8,192 bytes 07_H 16,384 bytes 08_H 32 KB 09_H 64 KB $0A_H$ 128 KB
OSTEN	11	r	**One-Step Timestamping Enable** This bit is set to 1 when the Enable One-Step Timestamp Feature is selected. Value After Reset: 0x1
PTOEN	12	r	**PTP Offload Enable** This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected. Value After Reset: 0x0
ADVTHWORD	13	r	**IEEE 1588 High Word Register Enable** This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected Value After Reset: 0x1
ADDR64	15:14	r	**Address Width.** This field indicates the configured address width: Value After Reset: 0x0 00_B 32 01_B 40 10_B 48 11_B Reserved
DCBEN	16	r	**DCB Feature Enable** This bit is set to 1 when the Enable Data Center Bridging option is selected Value After Reset: 0x0
SPHEN	17	r	**Split Header Feature Enable** This bit is set to 1 when the Enable Split Header Structure option is selected Value After Reset: 0x1

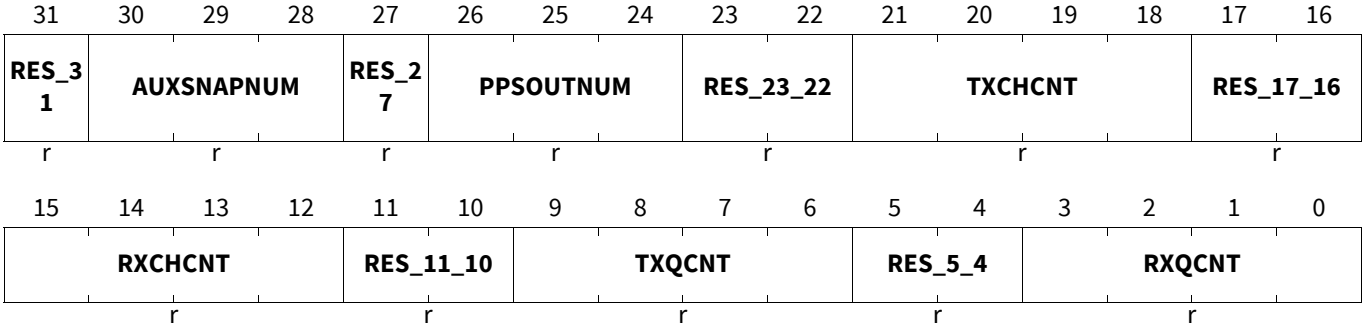
Field	Bits	Type	Description
TSOEN	18	r	**TCP Segmentation Offload Enable** This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected Value After Reset: 0x0
DBGMEMA	19	r	**DMA Debug Registers Enable** This bit is set to 1 when the Debug Mode Enable option is selected Value After Reset: 0x0
AVSEL	20	r	**AV Feature Enable** This bit is set to 1 when the Enable Audio Video Bridging option is selected. Value After Reset: 0x1
RAVSEL	21	r	**Rx Side Only AV Feature Enable** This bit is set to 1 when the Enable Audio Video Bridging option on Rx Side Only is selected. Value After Reset: 0x0
RES_22	22	r	**Reserved** Value After Reset: 0x0
POUOST	23	r	**One Step for PTP over UDP/IP Feature Enable** This bit is set to 1 when the Enable One step timestamp for PTP over UDP/IP feature is selected. Value After Reset: 0x1
HASHTBLSZ	25:24	r	**Hash Table Size** This field indicates the size of the hash table: Value After Reset: 0x0 00_B No hash table 01_B 64 10_B 128 11_B 256
RES_26	26	r	**Reserved** Value After Reset: 0x0
L3L4FNUM	30:27	r	**Total number of L3 or L4 Filters** This field indicates the total number of L3 or L4 filters: LOST SEQUENCE DEFINITION Value After Reset: 0x0 0_H No L3 or L4 Filter 1_H 1 L3 or L4 Filter 2_H 2 L3 or L4 Filters 8_H 8 L3 or L4
RES_31	31	r	**Reserved** Value After Reset: 0x0

MAC Hardware Feature Register 2

This register indicates the presence of third set of the optional features or functions of the DWC_ether_qos. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

MAC_HW_FEATURE2

MAC Hardware Feature Register 2 **(0124_H)** **Application Reset Value: $010C\ 30C3_H$**



Field	Bits	Type	Description
RXQCNT	3:0	r	**Number of MTL Receive Queues** This field indicates the number of MTL Receive queues: LOST SEQUENCE DEFINITION Value After Reset: 0x3 0_H 1 MTL Rx Queue 1_H 2 MTL Rx Queues 7_H 8 MTL Rx
RES_5_4	5:4	r	**Reserved** Value After Reset: 0x0
TXQCNT	9:6	r	**Number of MTL Transmit Queues** This field indicates the number of MTL Transmit queues: LOST SEQUENCE DEFINITION Value After Reset: 0x3 0_H 1 MTL Tx Queue 1_H 2 MTL Tx Queues 7_H 8 MTL Tx
RES_11_10	11:10	r	**Reserved** Value After Reset: 0x0
RXCHCNT	15:12	r	**Number of DMA Receive Channels** This field indicates the number of DMA Receive channels: LOST SEQUENCE DEFINITION Value After Reset: 0x3 0_H 1 DMA Rx Channel 1_H 2 DMA Rx Channels 7_H 8 DMA Rx
RES_17_16	17:16	r	**Reserved** Value After Reset: 0x0

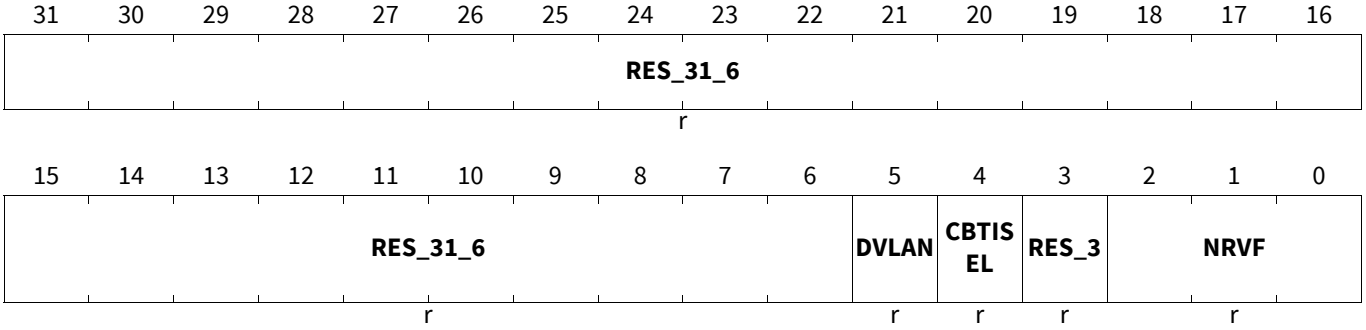
Field	Bits	Type	Description
TXCHCNT	21:18	r	**Number of DMA Transmit Channels** This field indicates the number of DMA Transmit channels: LOST SEQUENCE DEFINITION Value After Reset: 0x3 0_H 1 DMA Tx Channel 1_H 2 DMA Tx Channels 7_H 8 DMA Tx
RES_23_22	23:22	r	**Reserved** Value After Reset: 0x0
PPSOUTNUM	26:24	r	**Number of PPS Outputs** This field indicates the number of PPS outputs: 101_B-111_B Reserved Value After Reset: 0x1 000_B No PPS output 001_B 1 PPS output 010_B 2 PPS outputs 011_B 3 PPS outputs 100_B 4 PPS outputs
RES_27	27	r	**Reserved** Value After Reset: 0x0
AUXSNAPNUM	30:28	r	**Number of Auxiliary Snapshot Inputs** This field indicates the number of auxiliary snapshot inputs: 101_B-111_B Reserved Value After Reset: 0x0 000_B No auxiliary input 001_B 1 auxiliary input 010_B 2 auxiliary inputs 011_B 3 auxiliary inputs 100_B 4 auxiliary inputs
RES_31	31	r	**Reserved** Value After Reset: 0x0

MAC Hardware Feature Register 3

This register indicates the presence of fourth set the optional features or functions of the DWC_ether_qos. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

MAC_HW_FEATURE3

MAC Hardware Feature Register 3 **(0128_H)** **Application Reset Value: $0032\ 0032_H$**



Field	Bits	Type	Description
NRVF	2:0	r	**Number of Extended VLAN Tag Filters Enabled** This field indicates the Number of Extended VLAN Tag Filters selected: 110_B-111_B Reserved Value After Reset: 0x2 000_B No Extended Rx VLAN Filters 001_B 4 Extended Rx VLAN Filters 010_B 8 Extended Rx VLAN Filters 011_B 16 Extended Rx VLAN Filters 100_B 24 Extended Rx VLAN Filters 101_B 32 Extended Rx VLAN Filters
RES_3	3	r	**Reserved** Value After Reset: 0x0
CBTISEL	4	r	**Queue/Channel based VLAN tag insertion on Tx Enable** This bit is set to 1 when the Enable Queue/Channel based VLAN tag insertion on Tx Feature is selected. Value After Reset: 0x1
DVLAN	5	r	**DVLAN** Value After Reset: 0x1
RES_31_6	31:6	r	**Reserved** Value After Reset: 0x0

MAC MDIO Address Register

The MDIO Address register controls the management cycles to external PHY through a management interface.

MAC_MDIO_ADDRESS

MAC MDIO Address Register **(0200_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_28				PSE	BTB	PA					RDA				
r				rw	rw	rw					rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15	NTC			CR				RES_7_5			SKAP	GOC_1	GOC_0	C45E	GB
r	rw			rw				r			rw	rw	rw	rw	rw

Field	Bits	Type	Description
GB	0	rw	**GMII Busy** The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIO slave. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in MAC_MDIO_Address and MAC_MDIO_Data registers as long as this bit is set. For write transfers, the application must first write 16-bit data in the GDl field (and also RA field when C45E is set) in MAC_MDIO_Data register before setting this bit. When C45E is set, it should also write into the RA field of MAC_MDIO_Data register before initiating a read transfer. When a read transfer is completed (GB=0), the data read from the PHY register is valid in the GD field of the MAC_MDIO_Data register. **Note**: Even if the addressed PHY is not present, there is no change in the functionality of this bit. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
C45E	1	rw	**Clause 45 PHY Enable** When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO. Value After Reset: 0x0
GOC_0	2	rw	**GMII Operation Command 0** This is the lower bit of the operation command to the PHY or RevMII. When in SMA mode (MDIO master) this bit along with GOC_1 determines the operation to be performed to the PHY. When only RevMII is selected in configuration this bit is read-only and tied to 1. Value After Reset: 0x0

Field	Bits	Type	Description
GOC_1	3	rw	**GMII Operation Command 1** This bit is higher bit of the operation command to the PHY or RevMII, GOC_1 and GOC_O is encoded as follows: 00_BReserved 01_BWrite 10_BPost Read Increment Address for Clause 45 PHY 11_BRead When Clause 22 PHY or RevMII is enabled, only Write and Read commands are valid. Value After Reset: 0x0
SKAP	4	rw	**Skip Address Packet** When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set. Value After Reset: 0x0
RES_7_5	7:5	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
CR	11:8	rw	**CSR Clock Range** The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency used in your design: 0110_B, 0111_B Reserved The suggested range of CSR clock frequency applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range. When Bit 11 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits as 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks: These bits are not used for accessing RevMII. These bits are read-only if the RevMII interface is selected as single PHY interface. Value After Reset: 0x0 0_H CSR clock = 60-100 MHz; MDC clock = CSR clock/42 1_H CSR clock = 100-150 MHz; MDC clock = CSR clock/62 2_H CSR clock = 20-35 MHz; MDC clock = CSR clock/16 3_H CSR clock = 35-60 MHz; MDC clock = CSR clock/26 4_H CSR clock = 150-250 MHz; MDC clock = CSR clock/102 5_H CSR clock = 250-300 MHz; MDC clock = CSR clock/124 6_H CSR clock = 300-500 MHz; MDC clock = CSR clock/204 7_H CSR clock = 500-800 MHz; MDC clock = CSR clock/324 8_H CSR clock/4 9_H CSR clock/6 A_H CSR clock/8 B_H CSR clock/10 C_H CSR clock/12 D_H CSR clock/14 E_H CSR clock/16 F_H CSR clock/18
NTC	14:12	rw	**Number of Trailing Clocks** This field controls the number of trailing clock cycles generated on gmii_mdc_o (MDC) after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 3'h3 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer. Value After Reset: 0x0
RES_15	15	r	**Reserved** Value After Reset: 0x0
RDA	20:16	rw	**Register/Device Address** These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY. Value After Reset: 0x0

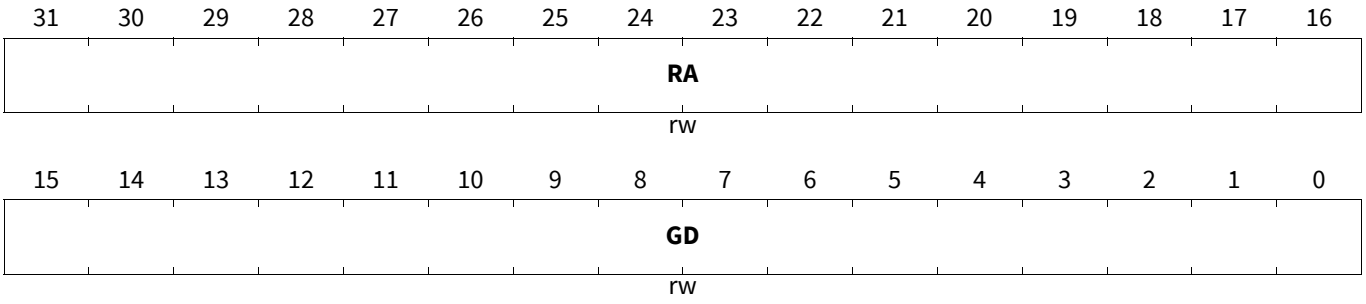
Field	Bits	Type	Description
PA	25:21	rw	**Physical Layer Address** This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing. This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing. Value After Reset: 0x0
BTB	26	rw	**Back to Back transactions** When this bit is set and the NTC has value greater than 0, then the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame. When this bit is reset, then the read/write command completion (GB is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame. This bit must not be set when NTC=0. Value After Reset: 0x0
PSE	27	rw	**Preamble Suppression Enable** When this bit is set, the SMA suppresses the 32-bit preamble and transmits MDIO frames with only 1 preamble bit. When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications. Value After Reset: 0x0
RES_31_28	31:28	r	**Reserved** Value After Reset: 0x0

MAC MDIO Data Register

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in MAC_MDIO_Address. This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

MAC_MDIO_DATA

MAC MDIO Data Register **(0204_H)** **Application Reset Value: $0000\ 0000_H$**



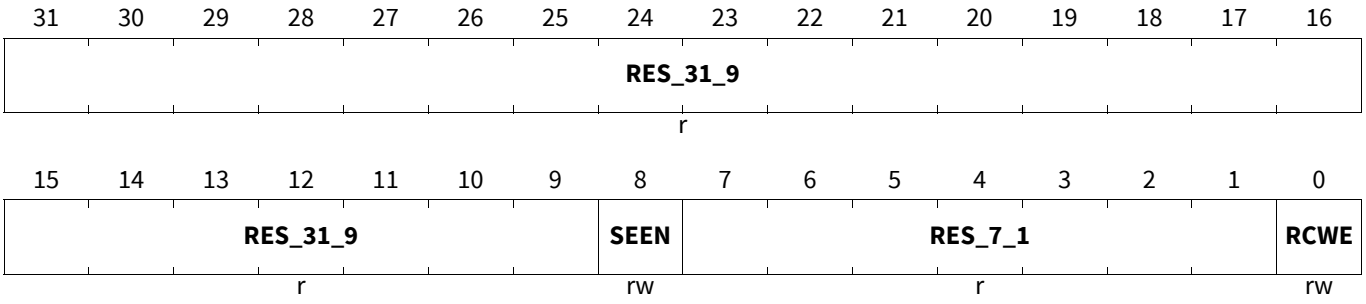
Field	Bits	Type	Description
GD	15:0	rw	**GMII Data** This field contains the 16-bit data value read from the PHY or RevMII after a Management Read operation or the 16-bit data value to be written to the PHY or RevMII before a Management Write operation. Value After Reset: 0x0
RA	31:16	rw	**Register Address** This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for. Value After Reset: 0x0

MAC CSR Software Controls Register

This register contains SW programmable controls for changing the CSR access response and status bits clearing.

MAC_CSR_SW_CTRL

MAC CSR Software Controls Register (0230$_H$) Application Reset Value: 0000 0000$_H$



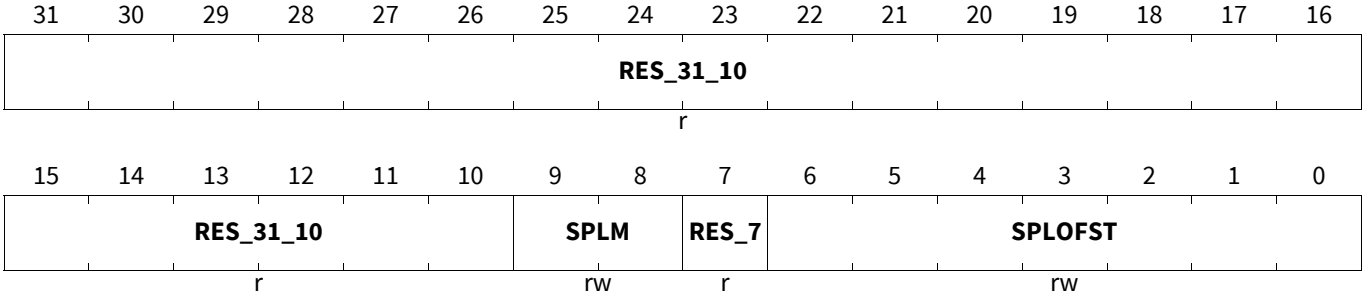
Field	Bits	Type	Description
RCWE	0	rw	**Register Clear on Write 1 Enable - RCWE** When this bit is set, the access mode of some register fields changes to Clear on Write 1, the application needs to set that respective bit to 1 to clear it. When this bit is reset, the access mode of these register fields remain as Clear on Read. Value After Reset: 0x0
RES_7_1	7:1	r	**Reserved** Value After Reset: 0x0
SEEN	8	rw	**Slave Error Response Enable** When this bit is set, the MAC responds with Slave Error for accesses to reserved registers in CSR space. When this bit is reset, the MAC responds with Okay response to any register accessed from CSR space. Value After Reset: 0x0
RES_31_9	31:9	r	**Reserved** Value After Reset: 0x0

MAC Extended Configuration Register 1

MAC Extended Configuration Register 1 contains Split mode control field and offset field for Split Header feature.

MAC_EXT_CFG1

MAC Extended Configuration Register 1 (0238_H) Application Reset Value: $0000\ 0002_H$



Field	Bits	Type	Description
SPLOFST	6:0	rw	**Split Offset** These bits indicate the value of offset from the beginning of Length/Type field at which header split should take place when the appropriate SPLM is selected. The reset value of this field is 2 bytes indicating a split at L2 header. Value is in terms ofbytes.
RES_7	7	r	**Reserved** Value After Reset: 0x0
SPLM	9:8	rw	**Split Mode** These bits indicate the mode of splitting the incoming Rx packets. They are 00_B Split at L3/L4 header 01_B Split at L2 header with an offset. Always Split atSPLOFST bytes from the beginning of Length/Type field ofthe Frame 10_B Combination mode: Split similar to SPLM=00 for IPpackets that are untagged or tagged and VLAN stripped 11_B Reserved
RES_31_10	31:10	r	**Reserved** Value After Reset: 0x0

MAC Address 0 High Register

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the (G)MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x112233445566 is received (0x11 in lane 0 of the first column) on the (G)MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x665544332211.

MAC_ADDRESS0_HIGH

MAC Address 0 High Register **(0300_H)** **Application Reset Value: 8000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	RES_30_x													DCS	
r	r													rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI															
rw															

Field	Bits	Type	Description
ADDRHI	15:0	rw	**MAC Address0[47:32]** This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets. Value After Reset: 0xffff
DCS	17:16	rw	**DMA Channel Select** If the PDC bit of MAC_Ext_Configuration register is not set: This field contains the binary representation of the DMA Channel number to which an Rx packet whose DA matches the MAC Address0 content is routed. If the PDC bit of MAC_Ext_Configuration register is set: This field contains the one-hot representation of one or more DMA Channel numbers to which an Rx packet whose DA matches the MAC Address0 content is routed.
RES_30_x	30:18	r	**Reserved** Value After Reset: 0x0
AE	31	r	**Address Enable** This bit is always set to 1. Value After Reset: 0x1

MAC Address 0 Low Register

The MAC Address0 Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

MAC_ADDRESS0_LOW

MAC Address 0 Low Register **(0304_H)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRLO															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO															
rw															

Field	Bits	Type	Description
ADDRLO	31:0	rw	**MAC Address0[31:0]** This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets. Value After Reset: 0xffffffff

MAC Address i High Register

The MAC Address1 High register holds the upper 16 bits of the second 6-byte MAC address of the station.

MAC_ADDRESSi_HIGH (i=1-31)
MAC Address i High Register **(0308_H+(i-1)*8)** **Application Reset Value: 0000 $FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	**SA**	**MBC**						**RES_23_x**						**DCS**	
rw	rw	rw						r						rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI															
rw															

Field	Bits	Type	Description
ADDRHI	15:0	rw	**MAC Address1 [47:32]** This field contains the upper 16 bits[47:32] of the second 6-byte MAC address. Value After Reset: 0xffff
DCS	17:16	rw	**DMA Channel Select** If the PDC bit of MAC_Ext_Configuration register is not set: This field contains the binary representation of the DMA Channel number to which an Rx packet whose DA matches the MAC Address0 content is routed. If the PDC bit of MAC_Ext_Configuration register is set: This field contains the one-hot representation of one or more DMA Channel numbers to which an Rx packet whose DA matches the MAC Address0 content is routed.
RES_23_x	23:18	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
MBC	29:24	rw	**Mask Byte Control** These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows: • Bit 29: Register 194[15:8] • Bit 28: Register 194[7:0] • Bit 27: Register 195[31:24] • … • Bit 24: Register 195[7:0] You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address. Value After Reset: 0x0
SA	30	rw	**Source Address** When this bit is set, the MAC Address1[47:0] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address1[47:0] is used to compare with the DA fields of the received packet. Value After Reset: 0x0
AE	31	rw	**Address Enable** When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering. Value After Reset: 0x0

MAC Address i Low Register

The MAC Address1 Low register holds the lower 32 bits of the second 6-byte MAC address of the station.

MAC_ADDRESSi_LOW (i=1-31)
MAC Address i Low Register **($030C_H$+(i-1)*8)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRLO															
rw															

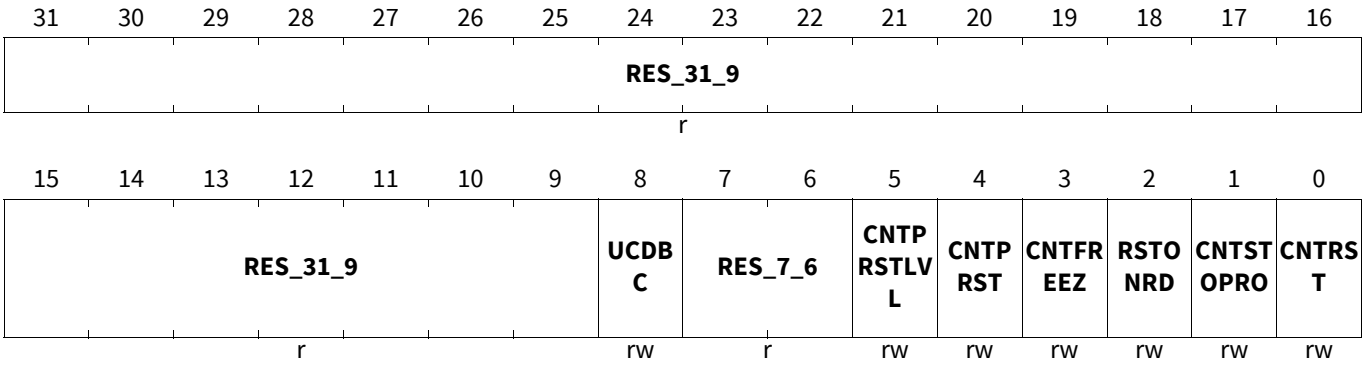
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO															
rw															

Field	Bits	Type	Description
ADDRLO	31:0	rw	**MAC Address1 [31:0]** This field contains the lower 32 bits of second 6-byte MAC address. The content of this field is undefined until loaded by the application after the initialization process. Value After Reset: 0xffffffff

MMC Control Register

This register establishes the operating mode of MMC.

MMC_CONTROL
MMC Control Register **(0700_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CNTRST	0	rw	**Counters Reset** When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets. Value After Reset: 0x0
CNTSTOPRO	1	rw	**Counter Stop Rollover** When this bit is set, the counter does not roll over to zero after reaching the maximum value. Value After Reset: 0x0
RSTONRD	2	rw	**Reset on Read** When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[7:0]) is read. Value After Reset: 0x0
CNTFREEZ	3	rw	**MMC Counter Freeze** When this bit is set, it freezes all MMC counters to their current value. Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode. Value After Reset: 0x0
CNTPRST	4	rw	**Counters Preset** When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle. This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full. Access restriction applies. Self-cleared. Setting 0 clears. Setting 1 sets. Value After Reset: 0x0

Field	Bits	Type	Description
CNTPRSTLVL	5	rw	**Full-Half Preset** When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF_F800 (Half 2KBytes) and all packet-counters gets preset to 0x7FFF_FFF0 (Half 16). When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF_F800 (Full 2KBytes) and all packet-counters gets preset to 0xFFFF_FFF0 (Full 16). For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0. Value After Reset: 0x0
RES_7_6	7:6	r	**Reserved** Value After Reset: 0x0
UCDBC	8	rw	**Update MMC Counters for Dropped Broadcast Packets** When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of MAC_Packet_Filter register. When reset, the MMC Counters are not updated for dropped Broadcast packets. Value After Reset: 0x0
RES_31_9	31:9	r	**Reserved** Value After Reset: 0x0

MMC Receive Interrupts Register

This register maintains the interrupts generated from all Receive statistics counters.

The MMC Receive Interrupt register maintains the interrupts that are generated when the following occur:

Receive statistic counters reach half of their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter).

Receive statistic counters cross their maximum values (0xFFFF_FFFF for 32 bit counter and 0xFFFF for 16 bit counter).

When the Counter Stop Rollover is set, interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

R_SS_RC means that this register bit is set internally, and it is cleared when the Counter register is read.

MMC_RX_INTERRUPT

MMC Receive Interrupts Register **(0704_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_28				RXLPITRCIS	RXLPIUSCIS	RXCTRLPIS	RXRCVERRPIS	RXWDOGPIS	RXVLANGBPIS	RXFOVPIS	RXPAUSPIS	RXORANGEPIS	RXLENERPIS	RXUCGPIS	RX1024TMAXOCTGBPIS
r				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX512T1023OCTGBPIS	RX256T511OCTGBPIS	RX128T255OCTGBPIS	RX65T127OCTGBPIS	RX64OCTGBPIS	RXOSIZEGPIS	RXUSIZEGPIS	RXJABERPIS	RXRUNTPIS	RXALGNERPIS	RXCRCERPIS	RXMCGPIS	RXBCGPIS	RXGOCTIS	RXGBOCTIS	RXGBPKTIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
RXGBPKTIS	0	r	**MMC Receive Good Bad Packet Counter Interrupt Status** This bit is set when the rxpacketcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXGBOCTIS	1	r	**MMC Receive Good Bad Octet Counter Interrupt Status** This bit is set when the rxoctetcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXGOCTIS	2	r	**MMC Receive Good Octet Counter Interrupt Status** This bit is set when the rxoctetcount_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXBCGPIS	3	r	**MMC Receive Broadcast Good Packet Counter Interrupt Status** This bit is set when the rxbroadcastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXMCGPIS	4	r	**MMC Receive Multicast Good Packet Counter Interrupt Status** This bit is set when the rxmulticastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXCRCERPIS	5	r	**MMC Receive CRC Error Packet Counter Interrupt Status** This bit is set when the rxcrcerror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
RXALGNERPIS	6	r	**MMC Receive Alignment Error Packet Counter Interrupt Status** This bit is set when the rxalignmenterror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXRUNTPIS	7	r	**MMC Receive Runt Packet Counter Interrupt Status** This bit is set when the rxrunterror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXJABERPIS	8	r	**MMC Receive Jabber Error Packet Counter Interrupt Status** This bit is set when the rxjabbererror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXUSIZEGPIS	9	r	**MMC Receive Undersize Good Packet Counter Interrupt Status** This bit is set when the rxundersize_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXOSIZEGPIS	10	r	**MMC Receive Oversize Good Packet Counter Interrupt Status** This bit is set when the rxoversize_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RX64OCTGBPIS	11	r	**MMC Receive 64 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the rx64octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RX65T127OCTGBPIS	12	r	**MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RX128T255OCTGBPIS	13	r	**MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
RX256T511OCTGBPIS	14	r	**MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RX512T1023OCTGBPIS	15	r	**MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RX1024TMAXOCTGBPIS	16	r	**MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status** This bit is set when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXUCGPIS	17	r	**MMC Receive Unicast Good Packet Counter Interrupt Status** This bit is set when the rxunicastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXLENERPIS	18	r	**MMC Receive Length Error Packet Counter Interrupt Status** This bit is set when the rxlengtherror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXORANGEPIS	19	r	**MMC Receive Out Of Range Error Packet Counter Interrupt Status.** This bit is set when the rxoutofrangetype counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXPAUSPIS	20	r	**MMC Receive Pause Packet Counter Interrupt Status** This bit is set when the rxpausepackets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXFOVPIS	21	r	**MMC Receive FIFO Overflow Packet Counter Interrupt Status** This bit is set when the rxfifooverflow counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
RXVLANGBPIS	22	r	**MMC Receive VLAN Good Bad Packet Counter Interrupt Status** This bit is set when the rxvlanpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXWDOGPIS	23	r	**MMC Receive Watchdog Error Packet Counter Interrupt Status** This bit is set when the rxwatchdog error counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXRCVERRPIS	24	r	**MMC Receive Error Packet Counter Interrupt Status** This bit is set when the rxrcverror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXCTRLPIS	25	r	**MMC Receive Control Packet Counter Interrupt Status** This bit is set when the rxctrlpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXLPIUSCIS	26	r	**MMC Receive LPI microsecond counter interrupt status** This bit is set when the Rx_LPI_USEC_Cntr counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXLPITRCIS	27	r	**MMC Receive LPI transition counter interrupt status** This bit is set when the Rx_LPI_Tran_Cntr counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_31_28	31:28	r	**Reserved** Value After Reset: 0x0

MMC Transmit Interrupts Register

This register maintains the interrupts generated from all Transmit statistics counters.

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values

(0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones.

The MMC Transmit Interrupt register is a 32 bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

MMC_TX_INTERRUPT

MMC Transmit Interrupts Register **(0708_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_28				TXLPITRCIS	TXLPIUSCIS	TXOSIZEGPIS	TXVLANGPIS	TXPAUSPIS	TXEXDEFPIS	TXGPKTIS	TXGOCTIS	TXCARERPIS	TXEXCOLPIS	TXLATCOLPIS	TXDEFPIS
r				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIS	TXSCOLGPIS	TXUFLOWERPIS	TXBCGBPIS	TXMCGBPIS	TXUCGBPIS	TX1024TMAXOCTGBPIS	TX512T1023OCTGBPIS	TX256T511OCTGBPIS	TX128T255OCTGBPIS	TX65T127OCTGBPIS	TX64OCTGBPIS	TXMCGPIS	TXBCGPIS	TXGBPKTIS	TXGBOCTIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
TXGBOCTIS	0	r	**MMC Transmit Good Bad Octet Counter Interrupt Status** This bit is set when the txoctetcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXGBPKTIS	1	r	**MMC Transmit Good Bad Packet Counter Interrupt Status** This bit is set when the txpacketcount_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXBCGPIS	2	r	**MMC Transmit Broadcast Good Packet Counter Interrupt Status** This bit is set when the txbroadcastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXMCGPIS	3	r	**MMC Transmit Multicast Good Packet Counter Interrupt Status** This bit is set when the txmulticastpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TX64OCTGBPIS	4	r	**MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the tx64octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TX65T127OCTGBPIS	5	r	**MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the tx65to127octets_gb counter reaches half the maximum value, and also when it reaches the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
TX128T255OCTGBPIS	6	r	**MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TX256T511OCTGBPIS	7	r	**MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TX512T1023OCTGBPIS	8	r	**MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Status** This bit is set when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TX1024TMAXOCTGBPIS	9	r	**MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Status** This bit is set when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXUCGBPIS	10	r	**MMC Transmit Unicast Good Bad Packet Counter Interrupt Status** This bit is set when the txunicastpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXMCGBPIS	11	r	**MMC Transmit Multicast Good Bad Packet Counter Interrupt Status** The bit is set when the txmulticastpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXBCGBPIS	12	r	**MMC Transmit Broadcast Good Bad Packet Counter Interrupt Status** This bit is set when the txbroadcastpackets_gb counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXUFLOWERPIS	13	r	**MMC Transmit Underflow Error Packet Counter Interrupt Status** This bit is set when the txunderflowerror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
TXSCOLGPIS	14	r	**MMC Transmit Single Collision Good Packet Counter Interrupt Status** This bit is set when the txsinglecol_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXMCOLGPIS	15	r	**MMC Transmit Multiple Collision Good Packet Counter Interrupt Status** This bit is set when the txmulticol_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXDEFPIS	16	r	**MMC Transmit Deferred Packet Counter Interrupt Status** This bit is set when the txdeferred counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXLATCOLPIS	17	r	**MMC Transmit Late Collision Packet Counter Interrupt Status** This bit is set when the txlatecol counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXEXCOLPIS	18	r	**MMC Transmit Excessive Collision Packet Counter Interrupt Status** This bit is set when the txexesscol counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXCARERPIS	19	r	**MMC Transmit Carrier Error Packet Counter Interrupt Status** This bit is set when the txcarriererror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXGOCTIS	20	r	**MMC Transmit Good Octet Counter Interrupt Status** This bit is set when the txoctetcount_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXGPKTIS	21	r	**MMC Transmit Good Packet Counter Interrupt Status** This bit is set when the txpacketcount_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXEXDEFPIS	22	r	**MMC Transmit Excessive Deferral Packet Counter Interrupt Status** This bit is set when the txexcessdef counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
TXPAUSPIS	23	r	**MMC Transmit Pause Packet Counter Interrupt Status** This bit is set when the txpausepacketserror counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXVLANGPIS	24	r	**MMC Transmit VLAN Good Packet Counter Interrupt Status** This bit is set when the txvlanpackets_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXOSIZEGPIS	25	r	**MMC Transmit Oversize Good Packet Counter Interrupt Status** This bit is set when the txoversize_g counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXLPIUSCIS	26	r	**MMC Transmit LPI microsecond counter interrupt status** This bit is set when the Tx_LPI_USEC_Cntr counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
TXLPITRCIS	27	r	**MMC Transmit LPI transition counter interrupt status** This bit is set when the Tx_LPI_Tran_Cntr counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_31_28	31:28	r	**Reserved** Value After Reset: 0x0

MMC Receive Interrupts Mask Register

This register maintains the masks for interrupts generated from all Receive statistics counters.

The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

This register is 32 bit wide.

MMC_RX_INTERRUPT_MASK

MMC Receive Interrupts Mask Register (070C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_28				RXLPITRCIM	RXLPIUSCIM	RXCTRLPIM	RXRCVERRPIM	RXWDOGPIM	RXVLANGBPIM	RXFOVPIM	RXPAUSPIM	RXORANGEPIM	RXLENERPIM	RXUCGPIM	RX1024TMAXOCTGBPIM
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX512T1023OCTGBPIM	RX256T511OCTGBPIM	RX128T255OCTGBPIM	RX65T127OCTGBPIM	RX64OCTGBPIM	RXOSIZEGPIM	RXUSIZEGPIM	RXJABERPIM	RXRUNTPIM	RXALGNERPIM	RXCRCERPIM	RXMCGPIM	RXBCGPIM	RXGOCTIM	RXGBOCTIM	RXGBPKTIM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RXGBPKTIM	0	rw	**MMC Receive Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxpacketcount_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXGBOCTIM	1	rw	**MMC Receive Good Bad Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxoctetcount_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXGOCTIM	2	rw	**MMC Receive Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxoctetcount_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXBCGPIM	3	rw	**MMC Receive Broadcast Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxbroadcastpackets_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXMCGPIM	4	rw	**MMC Receive Multicast Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxmulticastpackets_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXCRCERPIM	5	rw	**MMC Receive CRC Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxcrcerror counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXALGNERPIM	6	rw	**MMC Receive Alignment Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxalignmenterror counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0

Field	Bits	Type	Description
RXRUNTPIM	7	rw	**MMC Receive Runt Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxrunterror counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXJABERPIM	8	rw	**MMC Receive Jabber Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxjabbererror counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXUSIZEGPIM	9	rw	**MMC Receive Undersize Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxundersize_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXOSIZEGPIM	10	rw	**MMC Receive Oversize Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxoversize_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RX64OCTGBPIM	11	rw	**MMC Receive 64 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rx64octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RX65T127OCTGBPIM	12	rw	**MMC Receive 65 to 127 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rx65to127octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RX128T255OCTGBPIM	13	rw	**MMC Receive 128 to 255 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rx128to255octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RX256T511OCTGBPIM	14	rw	**MMC Receive 256 to 511 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rx256to511octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RX512T1023OCTGBPIM	15	rw	**MMC Receive 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rx512to1023octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RX1024TMAXOCTGBPIM	16	rw	**MMC Receive 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask.** Setting this bit masks the interrupt when the rx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0

Field	Bits	Type	Description
RXUCGPIM	17	rw	**MMC Receive Unicast Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxunicastpackets_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXLENERPIM	18	rw	**MMC Receive Length Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxlengtherror counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXORANGEPIM	19	rw	**MMC Receive Out Of Range Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxoutofrangetype counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXPAUSPIM	20	rw	**MMC Receive Pause Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxpausepackets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXFOVPIM	21	rw	**MMC Receive FIFO Overflow Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxfifooverflow counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXVLANGBPIM	22	rw	**MMC Receive VLAN Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxvlanpackets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXWDOGPIM	23	rw	**MMC Receive Watchdog Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxwatchdog counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXRCVERRPIM	24	rw	**MMC Receive Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxrcverror counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXCTRLPIM	25	rw	**MMC Receive Control Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxctrlpackets_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXLPIUSCIM	26	rw	**MMC Receive LPI microsecond counter interrupt Mask** Setting this bit masks the interrupt when the Rx_LPI_USEC_Cntr counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXLPITRCIM	27	rw	**MMC Receive LPI transition counter interrupt Mask** Setting this bit masks the interrupt when the Rx_LPI_Tran_Cntr counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0

Field	Bits	Type	Description
RES_31_28	31:28	r	**Reserved** Value After Reset: 0x0

MMC Transmit Interrupts Mask Register

This register maintains the masks for interrupts generated from all Transmit statistics counters.

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. This register is 32 bit wide.

MMC_TX_INTERRUPT_MASK
MMC Transmit Interrupts Mask Register (0710$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_28				TXLPITRCIM	TXLPIUSCIM	TXOSIZEGPIM	TXVLANGPIM	TXPAUSPIM	TXEXDEFPIM	TXGPKTIM	TXGOCTIM	TXCARERPIM	TXEXCOLPIM	TXLATCOLPIM	TXDEFPIM
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIM	TXSCOLGPIM	TXUFLOWERPIM	TXBCGBPIM	TXMCGBPIM	TXUCGBPIM	TX1024TMAXOCTGBPIM	TX512T1023OCTGBPIM	TX256T511OCTGBPIM	TX128T255OCTGBPIM	TX65T127OCTGBPIM	TX64OCTGBPIM	TXMCGPIM	TXBCGPIM	TXGBPKTIM	TXGBOCTIM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
TXGBOCTIM	0	rw	**MMC Transmit Good Bad Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the txoctetcount_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXGBPKTIM	1	rw	**MMC Transmit Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txpacketcount_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXBCGPIM	2	rw	**MMC Transmit Broadcast Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txbroadcastpackets_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXMCGPIM	3	rw	**MMC Transmit Multicast Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txmulticastpackets_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TX64OCTGBPIM	4	rw	**MMC Transmit 64 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the tx64octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0

Field	Bits	Type	Description
TX65T127OCTGBPIM	5	rw	**MMC Transmit 65 to 127 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the tx65to127octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TX128T255OCTGBPIM	6	rw	**MMC Transmit 128 to 255 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the tx128to255octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TX256T511OCTGBPIM	7	rw	**MMC Transmit 256 to 511 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the tx256to511octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TX512T1023OCTGBPIM	8	rw	**MMC Transmit 512 to 1023 Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the tx512to1023octets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TX1024TMAXOCTGBPIM	9	rw	**MMC Transmit 1024 to Maximum Octet Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the tx1024tomaxoctets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXUCGBPIM	10	rw	**MMC Transmit Unicast Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txunicastpackets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXMCGBPIM	11	rw	**MMC Transmit Multicast Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txmulticastpackets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXBCGBPIM	12	rw	**MMC Transmit Broadcast Good Bad Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txbroadcastpackets_gb counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXUFLOWERPIM	13	rw	**MMC Transmit Underflow Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txunderflowerror counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXSCOLGPIM	14	rw	**MMC Transmit Single Collision Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txsinglecol_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0

Field	Bits	Type	Description
TXMCOLGPIM	15	rw	**MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txmulticol_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXDEFPIM	16	rw	**MMC Transmit Deferred Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txdeferred counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXLATCOLPIM	17	rw	**MMC Transmit Late Collision Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txlatecol counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXEXCOLPIM	18	rw	**MMC Transmit Excessive Collision Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txexcesscol counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXCARERPIM	19	rw	**MMC Transmit Carrier Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txcarriererror counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXGOCTIM	20	rw	**MMC Transmit Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the txoctetcount_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXGPKTIM	21	rw	**MMC Transmit Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txpacketcount_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXEXDEFPIM	22	rw	**MMC Transmit Excessive Deferral Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txexcessdef counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXPAUSPIM	23	rw	**MMC Transmit Pause Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txpausepackets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXVLANGPIM	24	rw	**MMC Transmit VLAN Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txvlanpackets_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXOSIZEGPIM	25	rw	**MMC Transmit Oversize Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the txoversize_g counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0

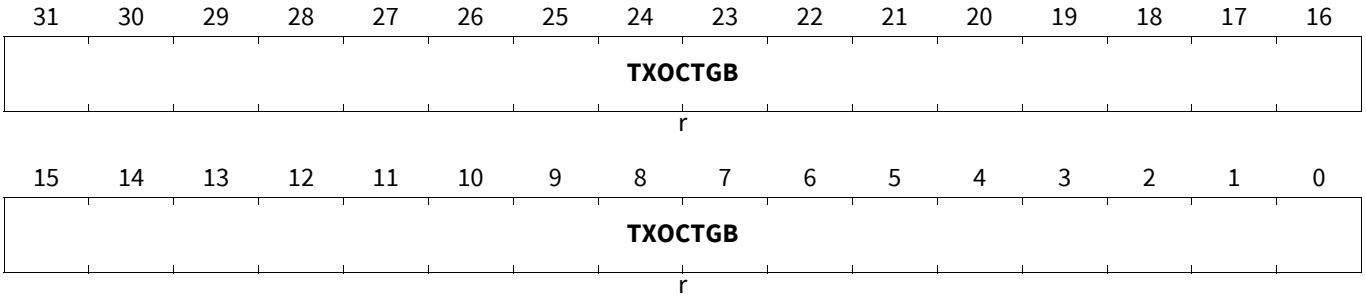
Field	Bits	Type	Description
TXLPIUSCIM	26	rw	**MMC Transmit LPI microsecond counter interrupt Mask** Setting this bit masks the interrupt when the Tx_LPI_USEC_Cntr counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
TXLPITRCIM	27	rw	**MMC Transmit LPI transition counter interrupt Mask** Setting this bit masks the interrupt when the Tx_LPI_Tran_Cntr counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RES_31_28	31:28	r	**Reserved** Value After Reset: 0x0

Good And Bad Transmitted Octet Count Register

This register provides the number of bytes transmitted by the DWC_ether_qos, exclusive of preamble and retried bytes, in good and bad packets.

TX_OCTET_COUNT_GOOD_BAD

Good And Bad Transmitted Octet Count Register(0714_H) **Application Reset Value: $0000\ 0000_H$**



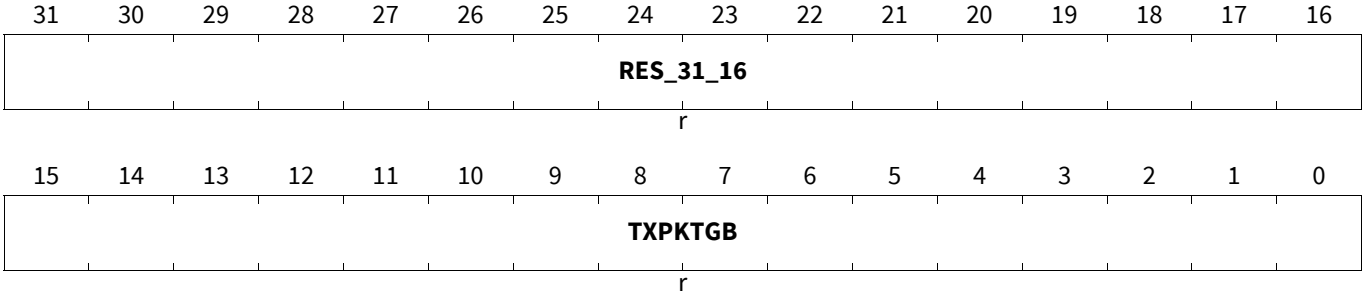
Field	Bits	Type	Description
TXOCTGB	31:0	r	**Tx Octet Count Good Bad** This field indicates the number of bytes transmitted, exclusive of preamble and retried bytes, in good and bad packets. Value After Reset: 0x0

Good And Bad Transmitted Packets Count Register

This register provides the number of good and bad packets transmitted by DWC_ether_qos, exclusive of retried packets.

TX_PACKET_COUNT_GOOD_BAD

Good And Bad Transmitted Packets Count Register(0718_H) **Application Reset Value: $0000\ 0000_H$**



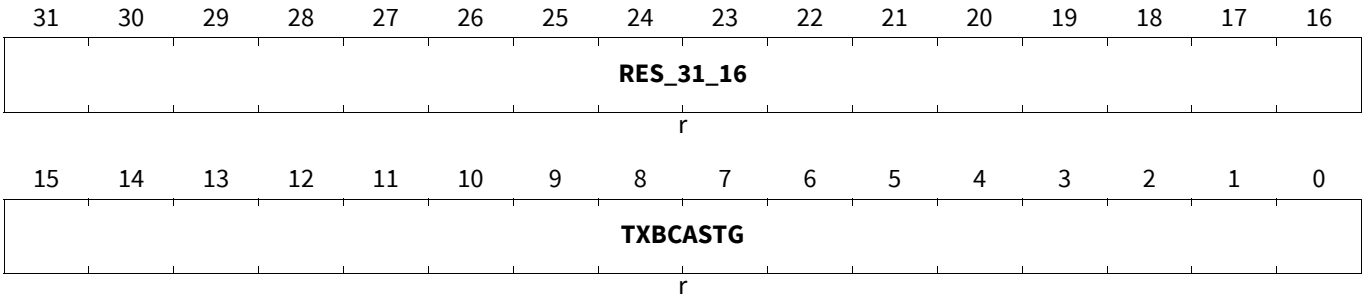
Field	Bits	Type	Description
TXPKTGB	15:0	r	**Tx Packet Count Good Bad** This field indicates the number of good and bad packets transmitted, exclusive of retried packets. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Transmitted Broadcast Packets Count Register

This register provides the number of good broadcast packets transmitted by DWC_ether_qos.

TX_BROADCAST_PACKETS_GOOD

Good Transmitted Broadcast Packets Count Register($071C_H$) **Application Reset Value: $0000\ 0000_H$**



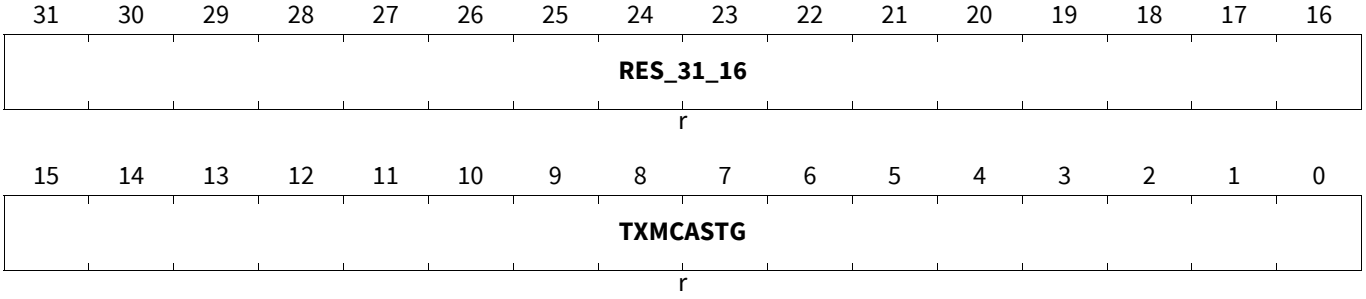
Field	Bits	Type	Description
TXBCASTG	15:0	r	**Tx Broadcast Packets Good** This field indicates the number of good broadcast packets transmitted. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Transmitted Multicast Packets Count Register

This register provides the number of good multicast packets transmitted by DWC_ether_qos.

TX_MULTICAST_PACKETS_GOOD

Good Transmitted Multicast Packets Count Register(0720_H) **Application Reset Value: $0000\ 0000_H$**



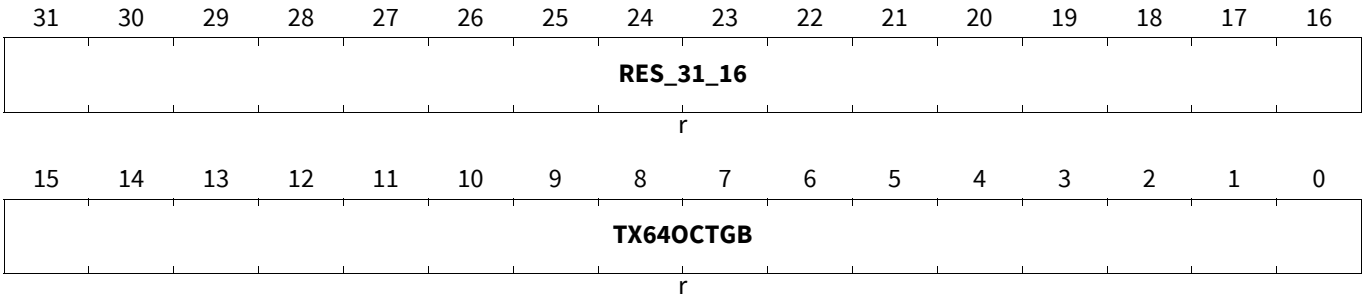
Field	Bits	Type	Description
TXMCASTG	15:0	r	**Tx Multicast Packets Good** This field indicates the number of good multicast packets transmitted. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 64 Octets Packets Transmitted Count Register

This register provides the number of good and bad packets transmitted by DWC_ether_qos with length 64 bytes, exclusive of preamble and retried packets.

TX_64OCTETS_PACKETS_GOOD_BAD

Good And Bad 64 Octets Packets Transmitted Count Register(0724_H) **Application Reset Value: $0000\ 0000_H$**



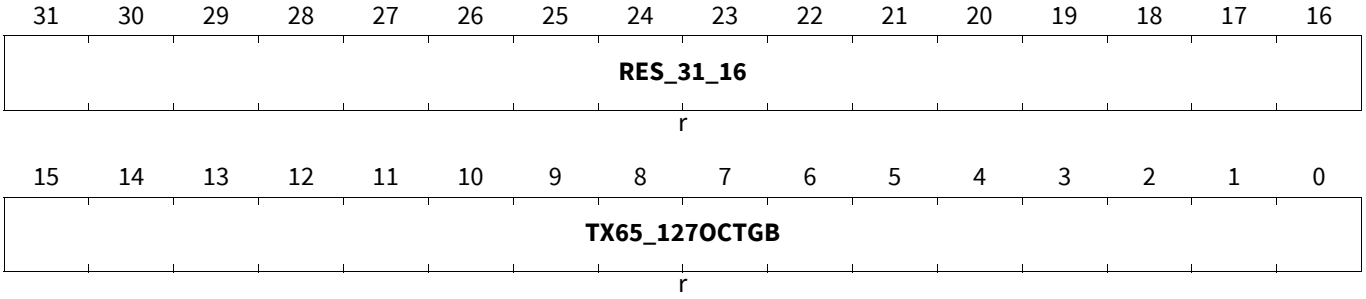
Field	Bits	Type	Description
TX64OCTGB	15:0	r	**Tx 64Octets Packets Good_Bad** This field indicates the number of good and bad packets transmitted with length 64 bytes, exclusive of preamble and retried packets. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 65to127 Octets Packets Transmitted Count Register

This register provides the number of good and bad packets transmitted by DWC_ether_qos with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets.

TX_65TO127OCTETS_PACKETS_GOOD_BAD

Good And Bad 65to127 Octets Packets Transmitted Count Register(0728_H) Application Reset Value: $0000\ 0000_H$



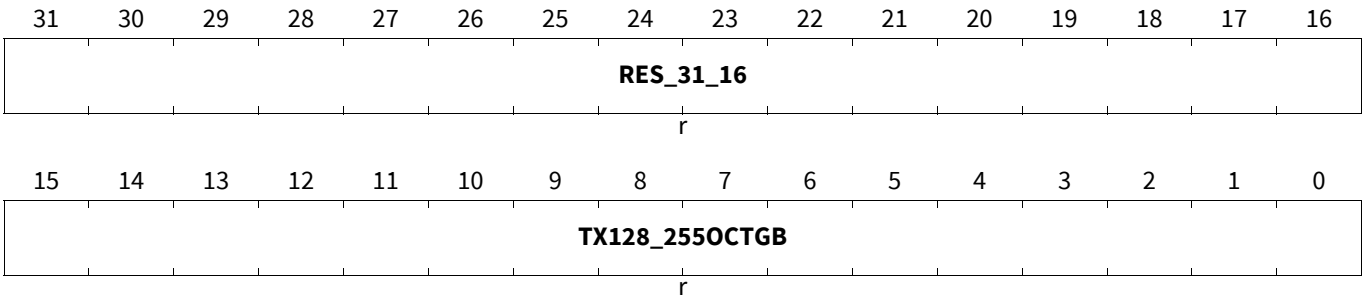
Field	Bits	Type	Description
TX65_127OCTGB	15:0	r	**Tx 65To127Octets Packets Good Bad** This field indicates the number of good and bad packets transmitted with length between 65 and 127 (inclusive) bytes, exclusive of preamble and retried packets. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 128to255 Octets Packets Transmitted Count Register

This register provides the number of good and bad packets transmitted by DWC_ether_qos with length between 128 to 255 (inclusive) bytes, exclusive of preamble and retried packets.

TX_128TO255OCTETS_PACKETS_GOOD_BAD

Good And Bad 128to255 Octets Packets Transmitted Count Register($072C_H$) Application Reset Value: $0000\ 0000_H$

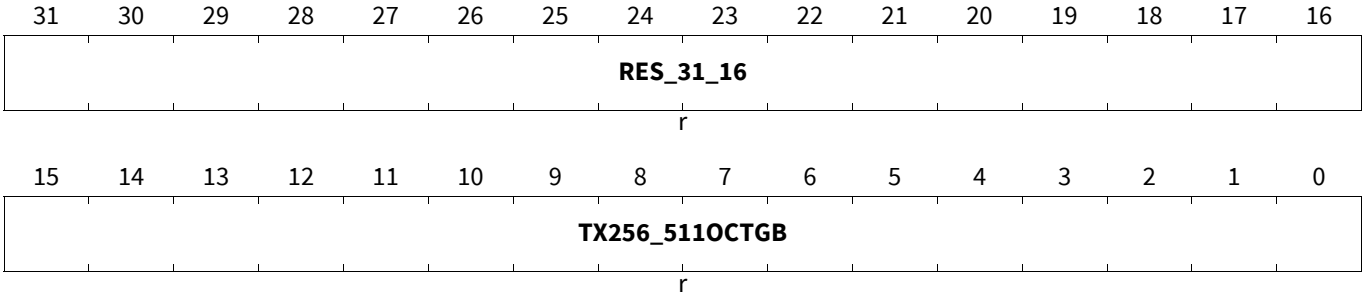


Field	Bits	Type	Description
TX128_255OCTGB	15:0	r	**Tx 128To255Octets Packets Good Bad** This field indicates the number of good and bad packets transmitted with length between 128 and 255 (inclusive) bytes, exclusive of preamble and retried packets. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 256to511 Octets Packets Transmitted Count Register

This register provides the number of good and bad packets transmitted by DWC_ether_qos with length between 256 to 511 (inclusive) bytes, exclusive of preamble and retried packets.

TX_256TO511OCTETS_PACKETS_GOOD_BAD
Good And Bad 256to511 Octets Packets Transmitted Count Register(0730_H) Application Reset Value: $0000\ 0000_H$

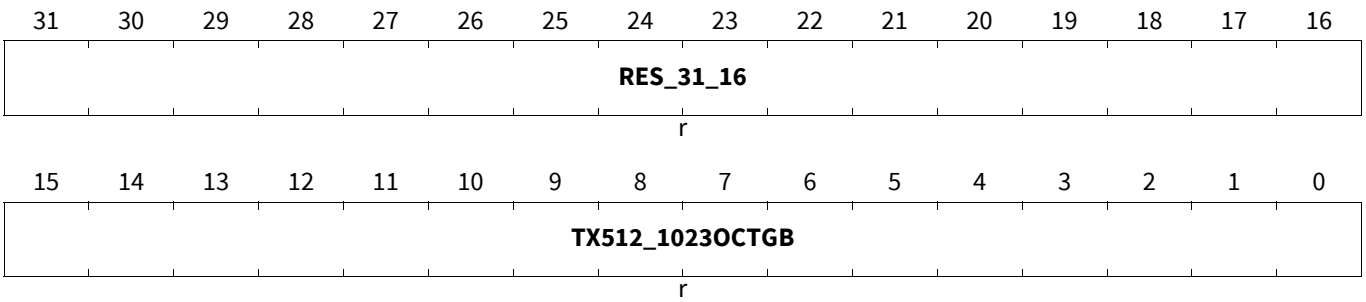


Field	Bits	Type	Description
TX256_511OCTGB	15:0	r	**Tx 256To511Octets Packets Good Bad** This field indicates the number of good and bad packets transmitted with length between 256 and 511 (inclusive) bytes, exclusive of preamble and retried packets. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 512to1023 Octets Packets Transmitted Count Register

This register provides the number of good and bad packets transmitted by DWC_ether_qos with length 512 to 1023 (inclusive) bytes, exclusive of preamble and retried packets.

TX_512TO1023OCTETS_PACKETS_GOOD_BAD
Good And Bad 512to1023 Octets Packets Transmitted Count Register(0734_H)Application Reset Value: $0000\ 0000_H$

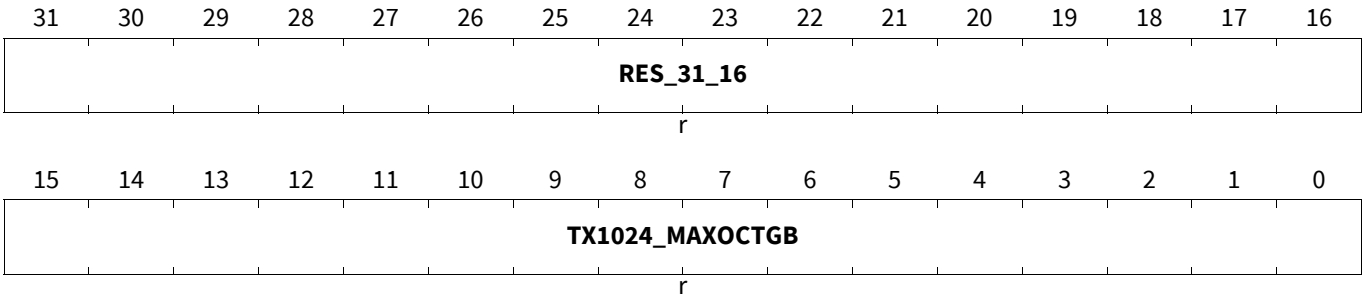


Field	Bits	Type	Description
TX512_1023OCTGB	15:0	r	**Tx 512To1023Octets Packets Good Bad** This field indicates the number of good and bad packets transmitted with length between 512 and 1023 (inclusive) bytes, exclusive of preamble and retried packets. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 1024toMax Octets Packets Transmitted Count Register

This register provides the number of good and bad packets transmitted by DWC_ether_qos with length 1024 to maxsize (inclusive) bytes, exclusive of preamble and retried packets.

TX_1024TOMAXOCTETS_PACKETS_GOOD_BAD
Good And Bad 1024toMax Octets Packets Transmitted Count Register(0738_H) Application Reset Value: $0000\ 0000_H$



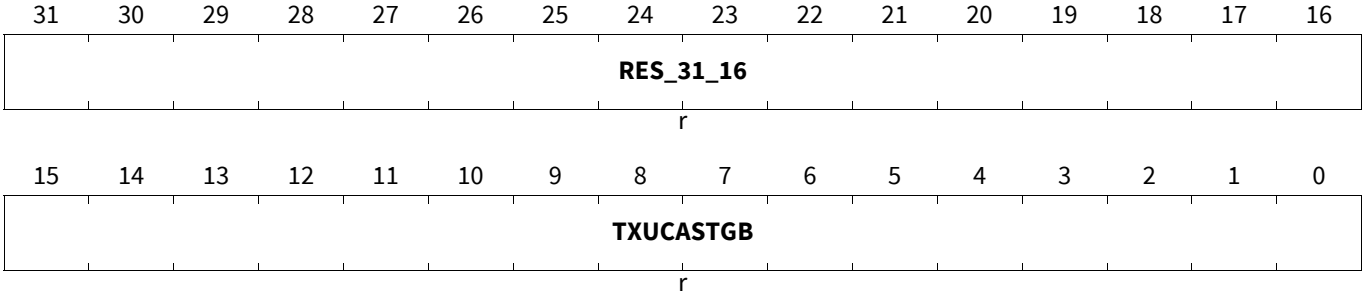
Field	Bits	Type	Description
TX1024_MAXOCTGB	15:0	r	**Tx 1024ToMaxOctets Packets Good Bad** This field indicates the number of good and bad packets transmitted with length between 1024 and maxsize (inclusive) bytes, exclusive of preamble and retried packets. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Transmitted Unicat Packets Count Register

This register provides the number of good and bad unicast packets transmitted by DWC_ether_qos.

TX_UNICAST_PACKETS_GOOD_BAD

Good Transmitted Unicat Packets Count Register($073C_H$) **Application Reset Value: $0000\ 0000_H$**



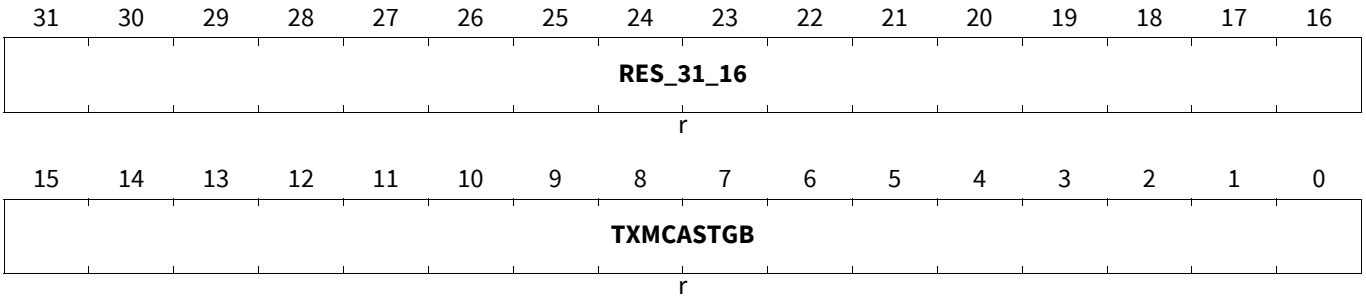
Field	Bits	Type	Description
TXUCASTGB	15:0	r	**Tx Unicast Packets Good Bad** This field indicates the number of good and bad unicast packets transmitted. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad Transmitted Multicast Packets Count Register

This register provides the number of good and bad multicast packets transmitted by DWC_ether_qos.

TX_MULTICAST_PACKETS_GOOD_BAD

Good And Bad Transmitted Multicast Packets Count Register(0740_H) **Application Reset Value: $0000\ 0000_H$**



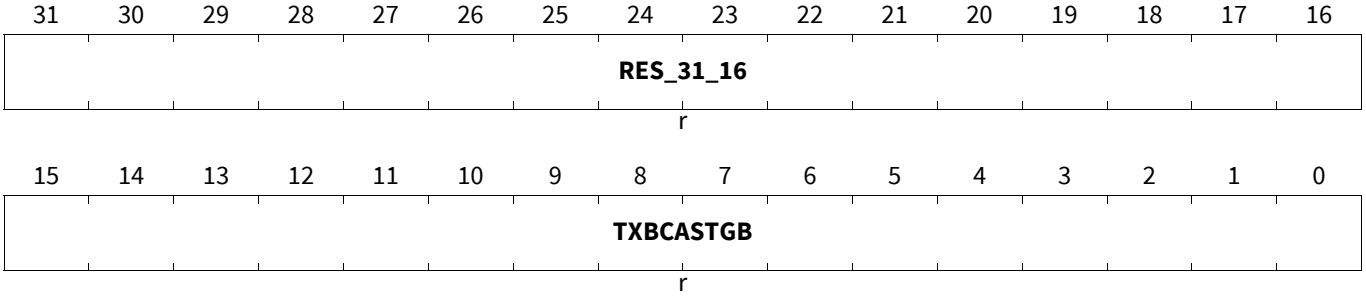
Field	Bits	Type	Description
TXMCASTGB	15:0	r	**Tx Multicast Packets Good Bad** This field indicates the number of good and bad multicast packets transmitted. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad Transmitted Broadcast Packets Count Register

This register provides the number of good and bad broadcast packets transmitted by DWC_ether_qos.

TX_BROADCAST_PACKETS_GOOD_BAD

Good And Bad Transmitted Broadcast Packets Count Register(0744_H) Application Reset Value: $0000\ 0000_H$



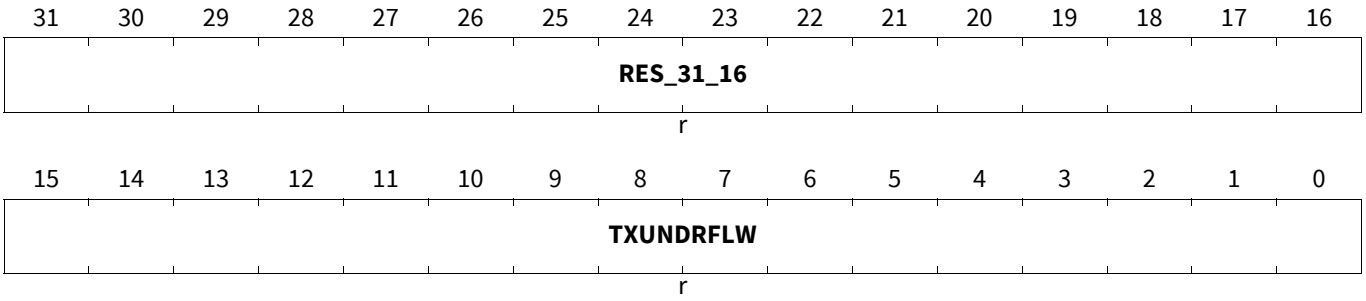
Field	Bits	Type	Description
TXBCASTGB	15:0	r	**Tx Broadcast Packets Good Bad** This field indicates the number of good and bad broadcast packets transmitted. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted Underflow Error Packets Count Register

This register provides the number of packets aborted by DWC_ether_qos because of packets underflow error.

TX_UNDERFLOW_ERROR_PACKETS

Transmitted Underflow Error Packets Count Register(0748_H) Application Reset Value: $0000\ 0000_H$



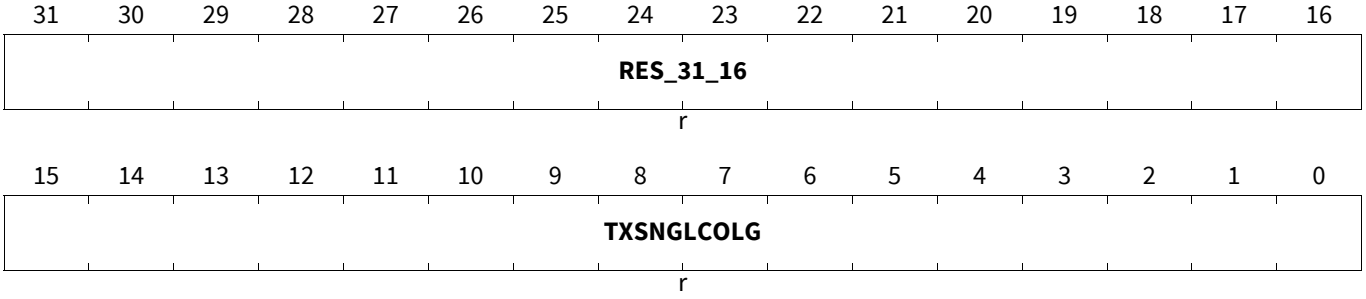
Field	Bits	Type	Description
TXUNDRFLW	15:0	r	**Tx Underflow Error Packets** This field indicates the number of packets aborted because of packets underflow error. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Transmitted Single Collision Count Register

This register provides the number of successfully transmitted packets by DWC_ether_qos after a single collision in the half-duplex mode.

TX_SINGLE_COLLISION_GOOD_PACKETS

Good Transmitted Single Collision Count Register(074C$_H$) **Application Reset Value: 0000 0000$_H$**



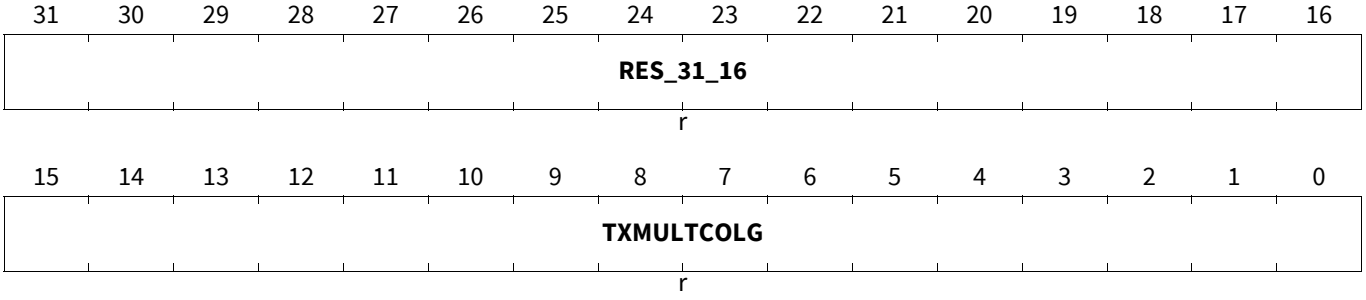
Field	Bits	Type	Description
TXSNGLCOLG	15:0	r	**Tx Single Collision Good Packets** This field indicates the number of successfully transmitted packets after a single collision in the half-duplex mode. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted Multiple Collision Count Register

This register provides the number of successfully transmitted packets by DWC_ether_qos after multiple collisions in the half-duplex mode.

TX_MULTIPLE_COLLISION_GOOD_PACKETS

Transmitted Multiple Collision Count Register (0750$_H$) **Application Reset Value: 0000 0000$_H$**



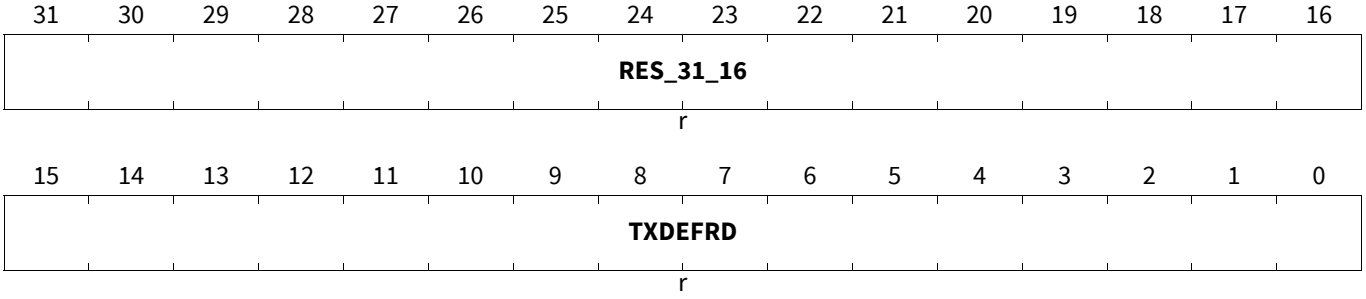
Field	Bits	Type	Description
TXMULTCOLG	15:0	r	**Tx Multiple Collision Good Packets** This field indicates the number of successfully transmitted packets after multiple collisions in the half-duplex mode. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted Deferred Packets Count Register

This register provides the number of successfully transmitted by DWC_ether_qos after a deferral in the half-duplex mode.

TX_DEFERRED_PACKETS

Transmitted Deferred Packets Count Register (0754$_H$) **Application Reset Value: 0000 0000$_H$**



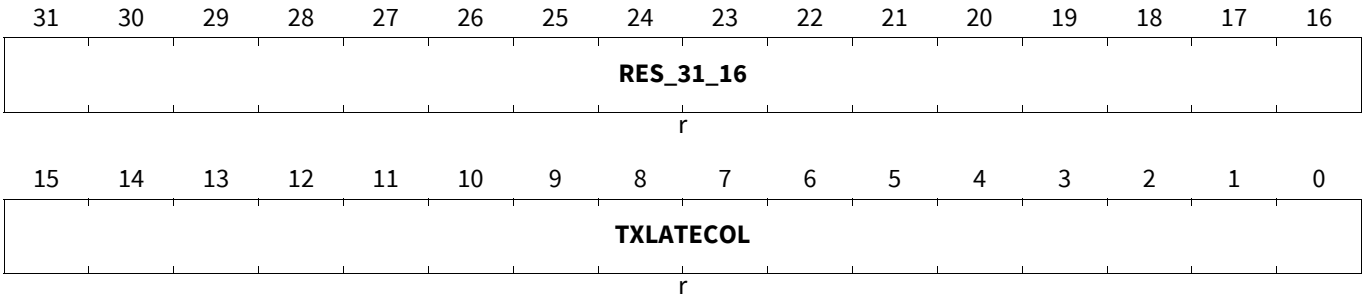
Field	Bits	Type	Description
TXDEFRD	15:0	r	**Tx Deferred Packets** This field indicates the number of successfully transmitted after a deferral in the half-duplex mode. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted Late Collision Packets Count Register

This register provides the number of packets aborted by DWC_ether_qos because of late collision error.

TX_LATE_COLLISION_PACKETS

Transmitted Late Collision Packets Count Register(0758$_H$) **Application Reset Value: 0000 0000$_H$**



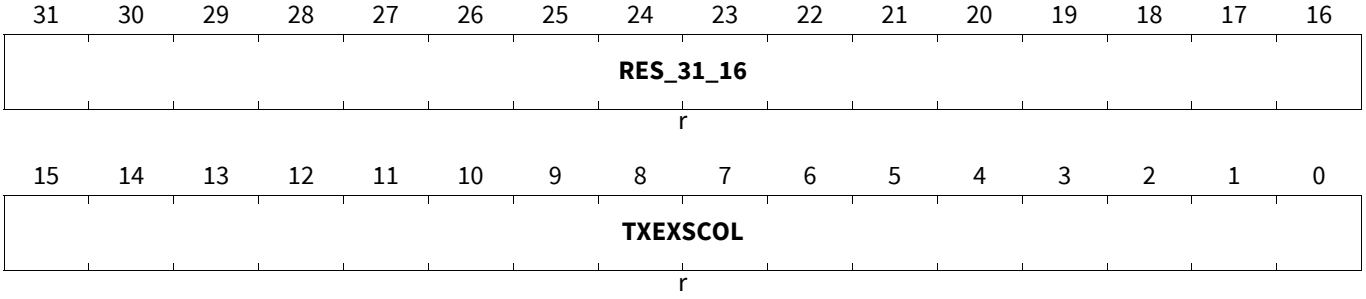
Field	Bits	Type	Description
TXLATECOL	15:0	r	**Tx Late Collision Packets** This field indicates the number of packets aborted because of late collision error. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted Excessive Collision Packets Count Register

This register provides the number of packets aborted by DWC_ether_qos because of excessive (16) collision errors.

TX_EXCESSIVE_COLLISION_PACKETS

Transmitted Excessive Collision Packets Count Register(075C$_H$) **Application Reset Value: 0000 0000$_H$**



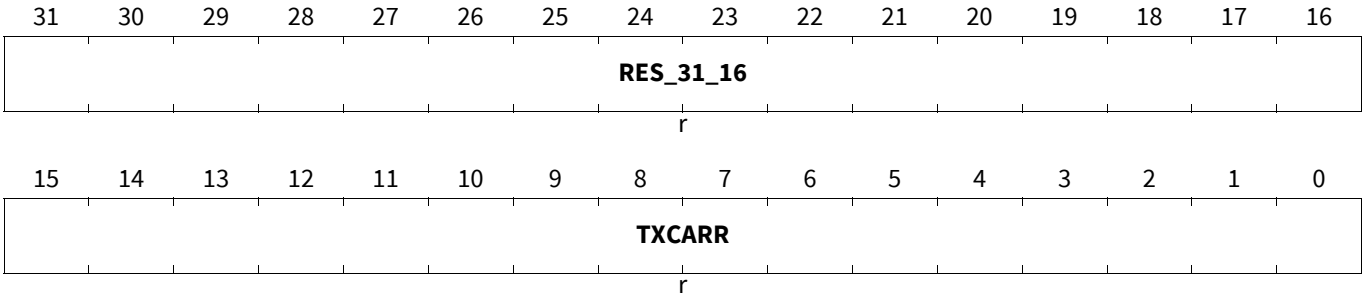
Field	Bits	Type	Description
TXEXSCOL	15:0	r	**Tx Excessive Collision Packets** This field indicates the number of packets aborted because of excessive (16) collision errors. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted Carrier Error Packets Count Register

This register provides the number of packets aborted by DWC_ether_qos because of carrier sense error (no carrier or loss of carrier).

TX_CARRIER_ERROR_PACKETS

Transmitted Carrier Error Packets Count Register(0760$_H$) **Application Reset Value: 0000 0000$_H$**



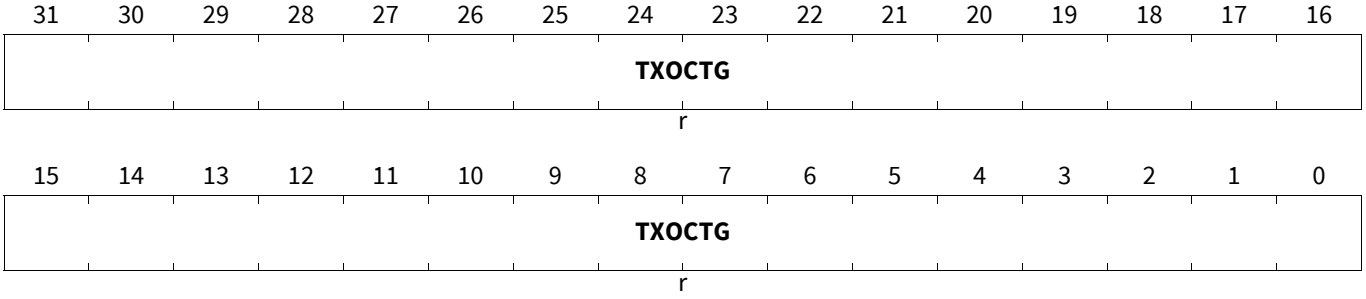
Field	Bits	Type	Description
TXCARR	15:0	r	**Tx Carrier Error Packets** This field indicates the number of packets aborted because of carrier sense error (no carrier or loss of carrier). Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Transmitted Octet Count Register

This register provides the number of bytes transmitted by DWC_ether_qos, exclusive of preamble, only in good packets.

TX_OCTET_COUNT_GOOD

Good Transmitted Octet Count Register (0764$_H$) Application Reset Value: 0000 0000$_H$



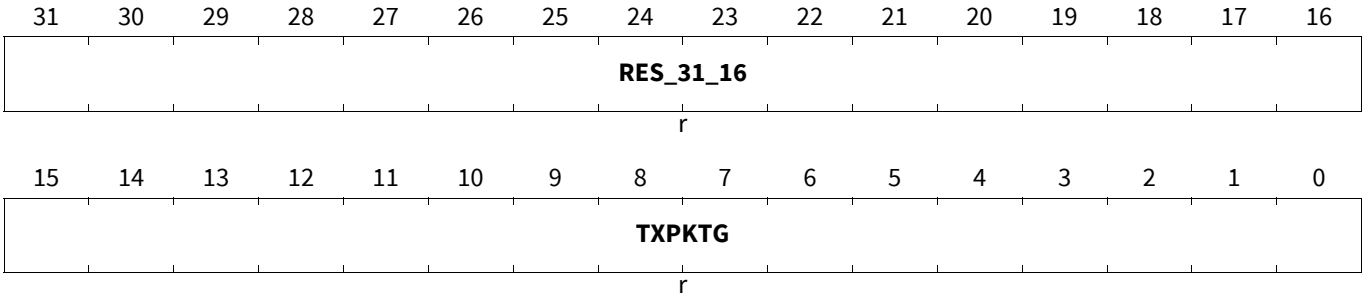
Field	Bits	Type	Description
TXOCTG	31:0	r	**Tx Octet Count Good** This field indicates the number of bytes transmitted, exclusive of preamble, only in good packets. Value After Reset: 0x0

Good Transmitted Packet Count Register

This register provides the number of good packets transmitted by DWC_ether_qos.

TX_PACKET_COUNT_GOOD

Good Transmitted Packet Count Register (0768$_H$) Application Reset Value: 0000 0000$_H$



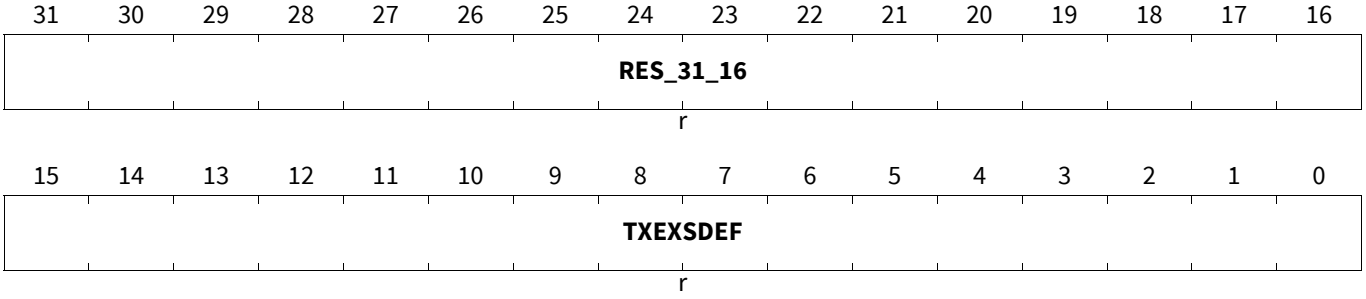
Field	Bits	Type	Description
TXPKTG	15:0	r	**Tx Packet Count Good** This field indicates the number of good packets transmitted. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted Excessive Deferral Error Count Register

This register provides the number of packets aborted by DWC_ether_qos because of excessive deferral error (deferred for more than two max-sized packet times).

TX_EXCESSIVE_DEFERRAL_ERROR

Transmitted Excessive Deferral Error Count Register(076C$_H$) **Application Reset Value: 0000 0000$_H$**



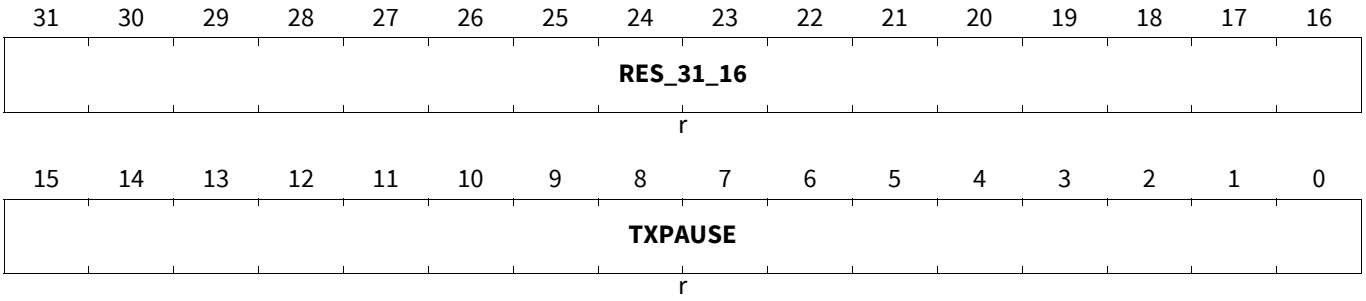
Field	Bits	Type	Description
TXEXSDEF	15:0	r	**Tx Excessive Deferral Error** This field indicates the number of packets aborted because of excessive deferral error (deferred for more than two max-sized packet times). Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted Pause Packets Count Register

This register provides the number of good Pause packets transmitted by DWC_ether_qos.

TX_PAUSE_PACKETS

Transmitted Pause Packets Count Register (0770$_H$) **Application Reset Value: 0000 0000$_H$**



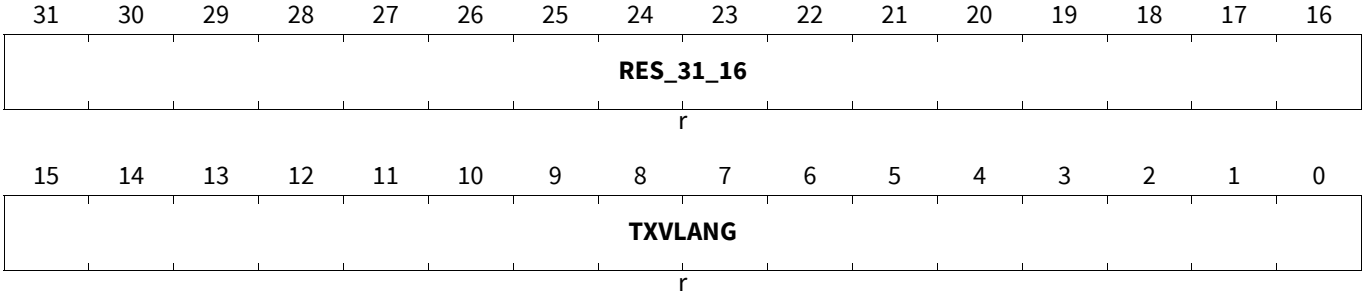
Field	Bits	Type	Description
TXPAUSE	15:0	r	**Tx Pause Packets** This field indicates the number of good Pause packets transmitted. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Transmitted VLAN Packets Count Register

This register provides the number of good VLAN packets transmitted by DWC_ether_qos.

TX_VLAN_PACKETS_GOOD

Good Transmitted VLAN Packets Count Register(0774_H) **Application Reset Value: $0000\ 0000_H$**



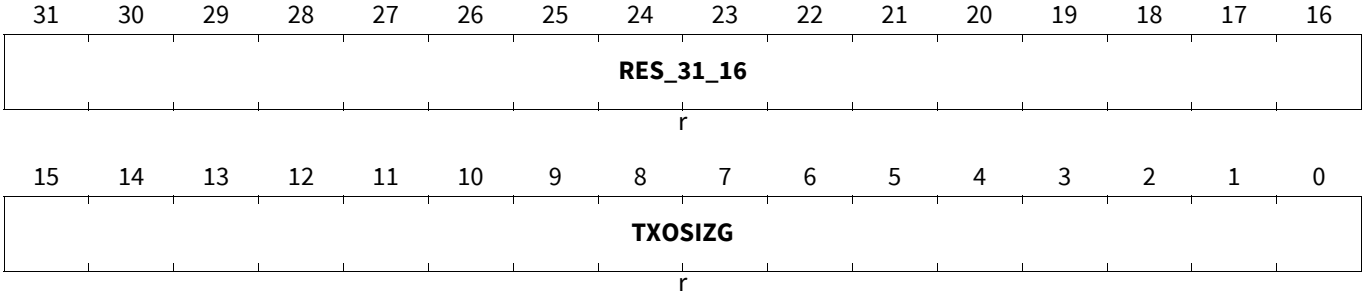
Field	Bits	Type	Description
TXVLANG	15:0	r	**Tx VLAN Packets Good** This field provides the number of good VLAN packets transmitted. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Transmitted Osize Packets Count Register

This register provides the number of packets transmitted by DWC_ether_qos without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in S2KP bit of the MAC_Configuration register).

TX_OSIZE_PACKETS_GOOD

Good Transmitted Osize Packets Count Register(0778_H) **Application Reset Value: $0000\ 0000_H$**



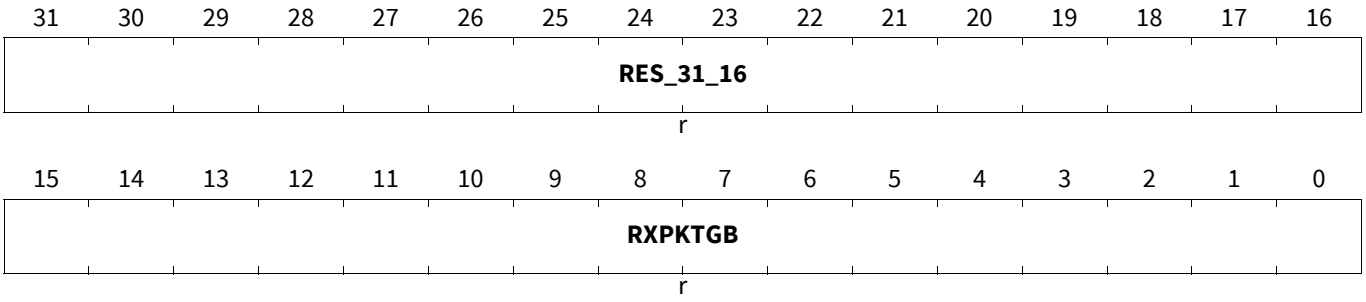
Field	Bits	Type	Description
TXOSIZG	15:0	r	**Tx OSize Packets Good** This field indicates the number of packets transmitted without errors and with length greater than the maxsize (1,518 or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in S2KP bit of the MAC_Configuration register). Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad Received Packets Count Register

This register provides the number of good and bad packets received by DWC_ether_qos.

RX_PACKETS_COUNT_GOOD_BAD

Good And Bad Received Packets Count Register (0780_H) **Application Reset Value: $0000\ 0000_H$**



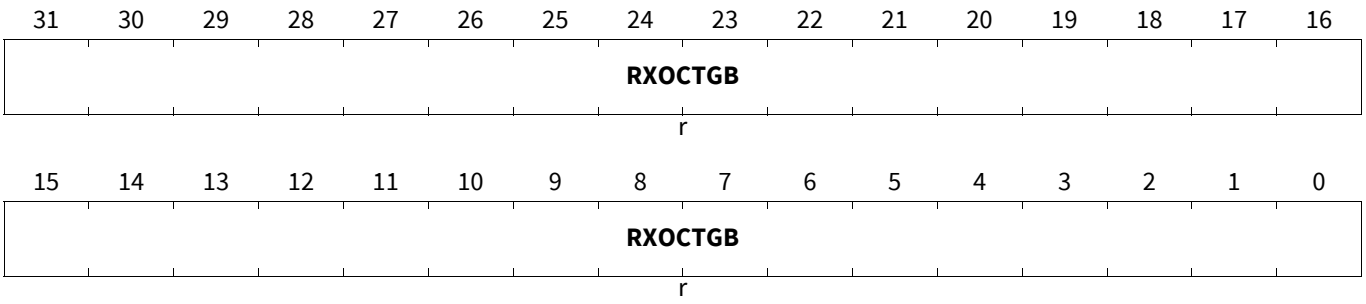
Field	Bits	Type	Description
RXPKTGB	15:0	r	**Rx Packets Count Good Bad** This field indicates the number of good and bad packets received. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad Received Octet Count Register

This register provides the number of bytes received by DWC_ther_qos, exclusive of preamble, in good and bad packets.

RX_OCTET_COUNT_GOOD_BAD

Good And Bad Received Octet Count Register (0784_H) **Application Reset Value: $0000\ 0000_H$**



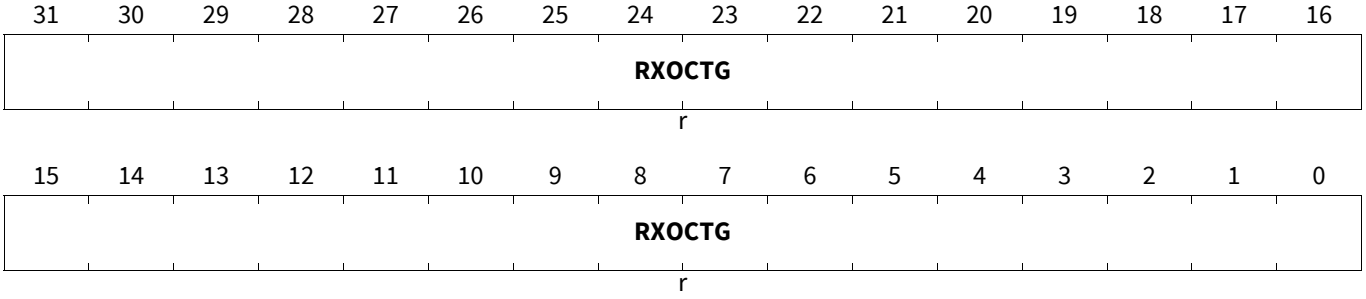
Field	Bits	Type	Description
RXOCTGB	31:0	r	**Rx Octet Count Good Bad** This field indicates the number of bytes received, exclusive of preamble, in good and bad packets. Value After Reset: 0x0

Good Received Octet Count Register

This register provides the number of bytes received by DWC_ether_qos, exclusive of preamble, only in good packets.

RX_OCTET_COUNT_GOOD

Good Received Octet Count Register (0788$_H$) Application Reset Value: 0000 0000$_H$



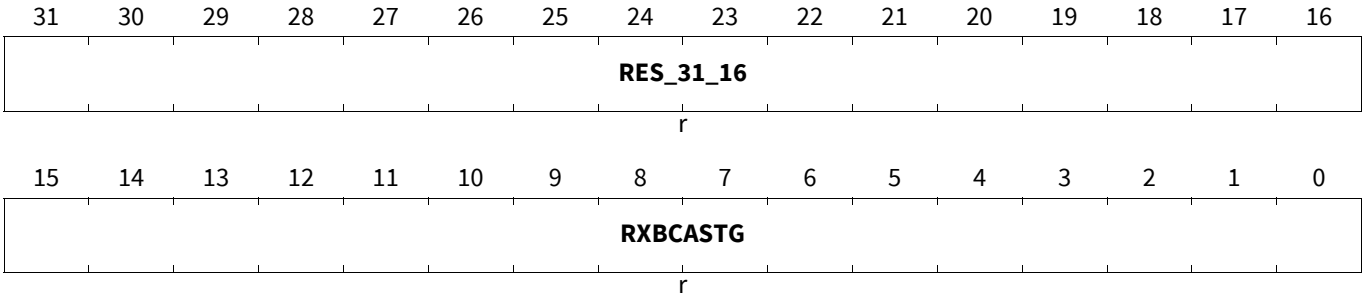
Field	Bits	Type	Description
RXOCTG	31:0	r	**Rx Octet Count Good** This field indicates the number of bytes received, exclusive of preamble, only in good packets. Value After Reset: 0x0

Good Received Broadcast Packets Count Register

This register provides the number of good broadcast packets received by DWC_ether_qos.

RX_BROADCAST_PACKETS_GOOD

Good Received Broadcast Packets Count Register(078C$_H$) Application Reset Value: 0000 0000$_H$



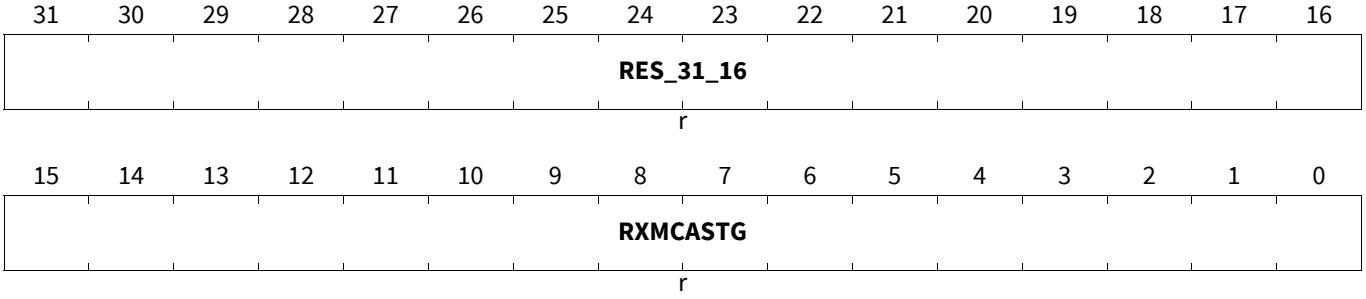
Field	Bits	Type	Description
RXBCASTG	15:0	r	**Rx Broadcast Packets Good** This field indicates the number of good broadcast packets received. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received Multicast Packets Count Register

This register provides the number of good multicast packets received by DWC_ether_qos.

RX_MULTICAST_PACKETS_GOOD

Good Received Multicast Packets Count Register(0790$_H$) **Application Reset Value: 0000 0000$_H$**



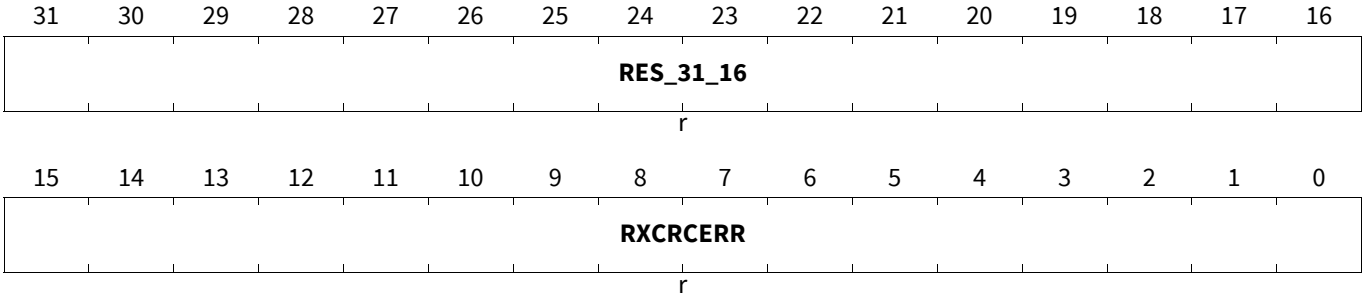
Field	Bits	Type	Description
RXMCASTG	15:0	r	**Rx Multicast Packets Good** This field indicates the number of good multicast packets received. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received CRC Error Packets Count Register

This register provides the number of packets received by DWC_ether_qos with CRC error.

RX_CRC_ERROR_PACKETS

Received CRC Error Packets Count Register (0794$_H$) **Application Reset Value: 0000 0000$_H$**



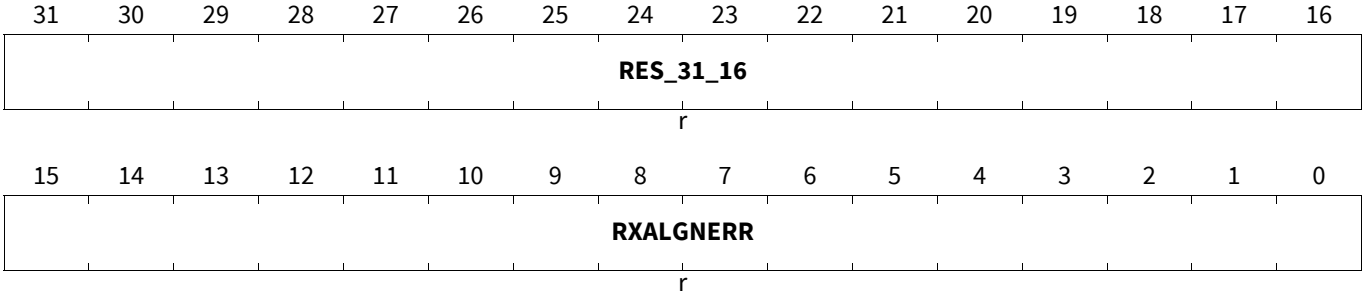
Field	Bits	Type	Description
RXCRCERR	15:0	r	**Rx CRC Error Packets** This field indicates the number of packets received with CRC error. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received Alignment Error Count Register

This register provides the number of packets received by DWC_ether_qos with alignment (dribble) error. It is valid only in 10/100 mode.

RX_ALIGNMENT_ERROR_PACKETS

Received Alignment Error Count Register (0798$_H$) Application Reset Value: 0000 0000$_H$



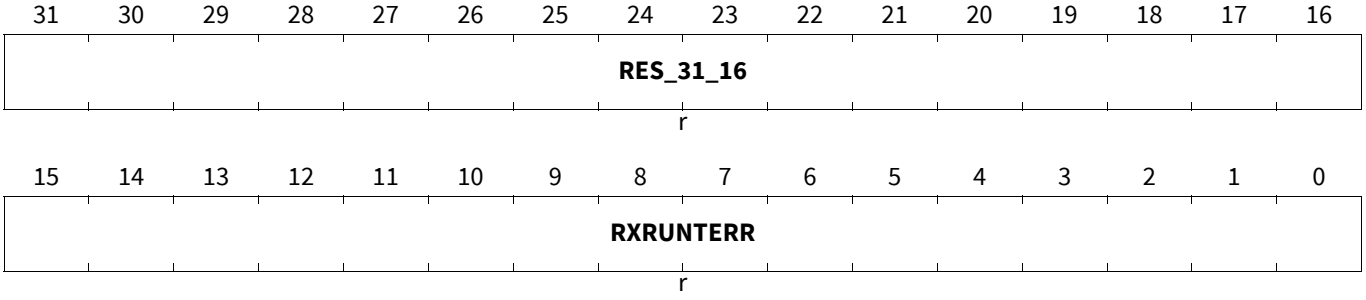
Field	Bits	Type	Description
RXALGNERR	15:0	r	**Rx Alignment Error Packets** This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received Runtime Error Count Register

This register provides the number of packets received by DWC_ether_qos with runt (length less than 64 bytes and CRC error) error.

RX_RUNT_ERROR_PACKETS

Received Runtime Error Count Register (079C$_H$) Application Reset Value: 0000 0000$_H$

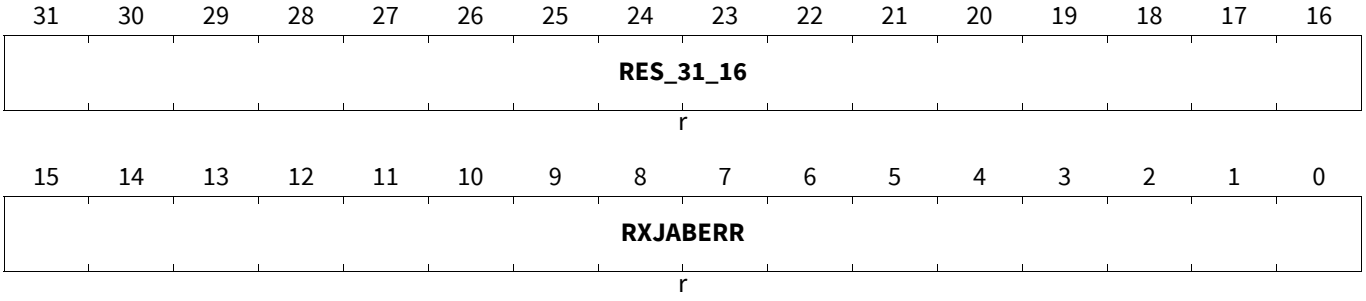


Field	Bits	Type	Description
RXRUNTERR	15:0	r	**Rx Runt Error Packets** This field indicates the number of packets received with runt (length less than 64 bytes and CRC error) error. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received Jabber Error Count Register

This register provides the number of giant packets received by DWC_ether_qos with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets.

RX_JABBER_ERROR_PACKETS
Received Jabber Error Count Register (07A0$_H$) Application Reset Value: 0000 0000$_H$

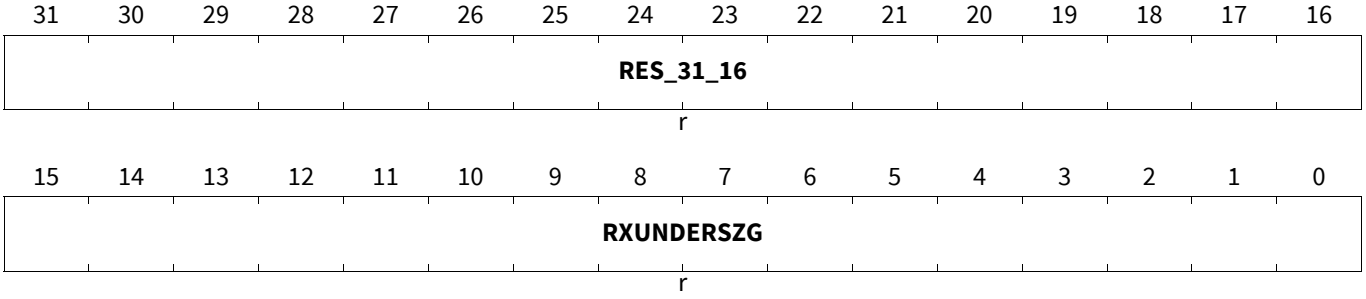


Field	Bits	Type	Description
RXJABERR	15:0	r	**Rx Jabber Error Packets** This field indicates the number of giant packets received with length (including CRC) greater than 1,518 bytes (1,522 bytes for VLAN tagged) and with CRC error. If Jumbo Packet mode is enabled, packets of length greater than 9,018 bytes (9,022 bytes for VLAN tagged) are considered as giant packets. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received Undersized Packets Count Register

This register provides the number of packets received by DWC_ether_qos with length less than 64 bytes, without any errors.

RX_UNDERSIZE_PACKETS_GOOD
Good Received Undersized Packets Count Register(07A4$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RXUNDERSZG	15:0	r	**Rx Undersize Packets Good** This field indicates the number of packets received with length less than 64 bytes, without any errors. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received Oversized Packets Count Register

This register provides the number of packets received by DWC_ether_qos without errors, with length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in the S2KP bit of the MAC_Configuration register).

RX_OVERSIZE_PACKETS_GOOD

Good Received Oversized Packets Count Register($07A8_H$) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXOVERSZG															
r															

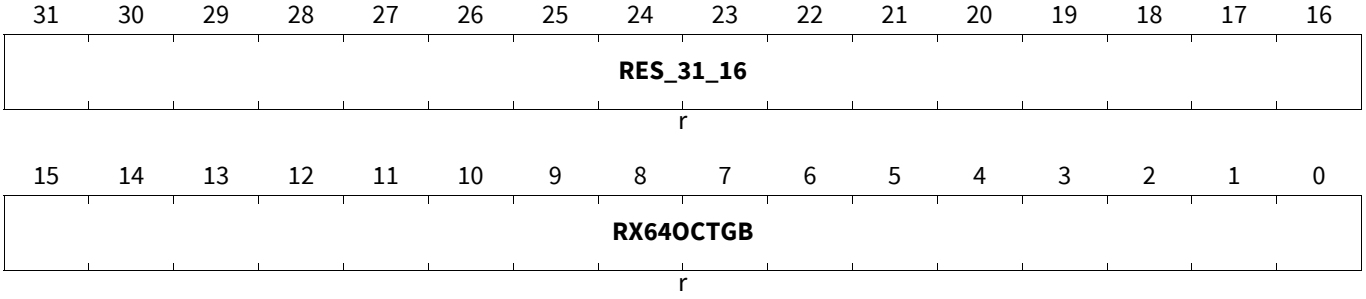
Field	Bits	Type	Description
RXOVERSZG	15:0	r	**Rx Oversize Packets Good** This field indicates the number of packets received without errors, with length greater than the maxsize (1,518 bytes or 1,522 bytes for VLAN tagged packets; 2000 bytes if enabled in the S2KP bit of the MAC_Configuration register). Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 64 Octets Packets Received Count Register

This register provides the number of good and bad packets received by DWC_ether_qos with length 64 bytes, exclusive of the preamble.

RX_64OCTETS_PACKETS_GOOD_BAD

Good And Bad 64 Octets Packets Received Count Register(07AC$_H$) **Application Reset Value: 0000 0000$_H$**



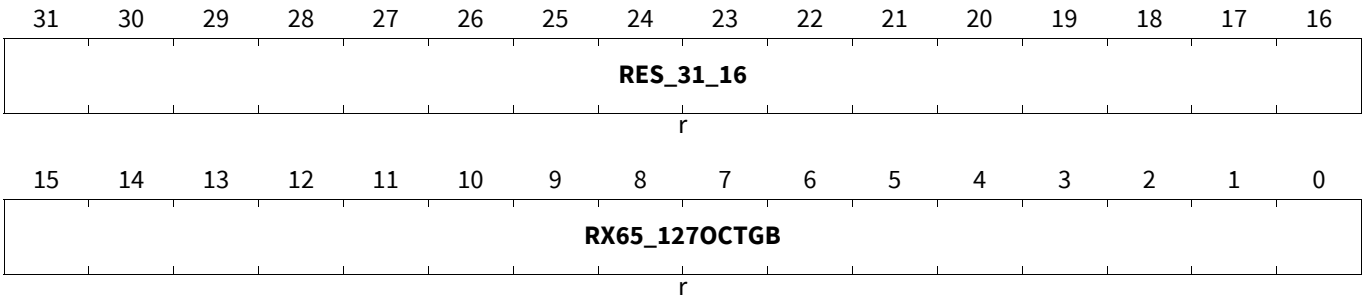
Field	Bits	Type	Description
RX64OCTGB	15:0	r	**Rx 64 Octets Packets Good Bad** This field indicates the number of good and bad packets received with length 64 bytes, exclusive of the preamble. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 65to127 Octets Packets Received Count Register

This register provides the number of good and bad packets received by DWC_ether_qos with length between 65 and 127 (inclusive) bytes, exclusive of the preamble.

RX_65TO127OCTETS_PACKETS_GOOD_BAD

Good And Bad 65to127 Octets Packets Received Count Register(07B0$_H$) **Application Reset Value: 0000 0000$_H$**

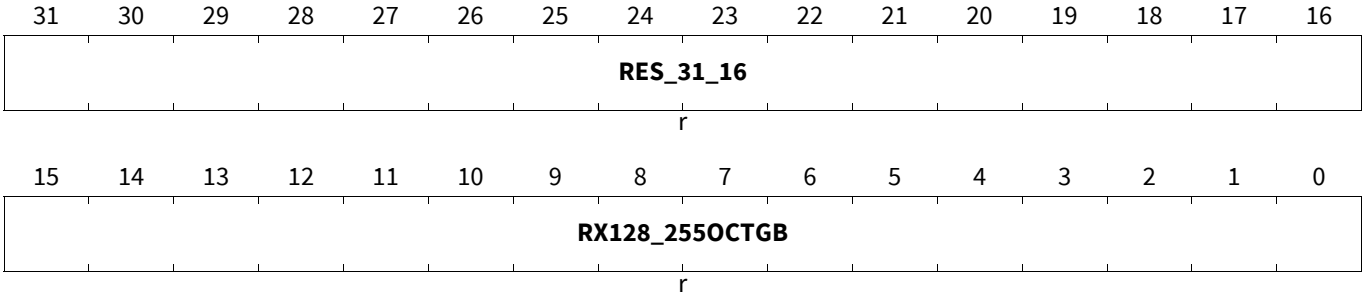


Field	Bits	Type	Description
RX65_127OCTGB	15:0	r	**Rx 65-127 Octets Packets Good Bad** This field indicates the number of good and bad packets received with length between 65 and 127 (inclusive) bytes, exclusive of the preamble. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 128to255 Octets Packets Received Count Register

This register provides the number of good and bad packets received by DWC_ether_qos with length between 128 and 255 (inclusive) bytes, exclusive of the preamble.

RX_128TO255OCTETS_PACKETS_GOOD_BAD
Good And Bad 128to255 Octets Packets Received Count Register($07B4_H$) Application Reset Value: 0000 0000_H

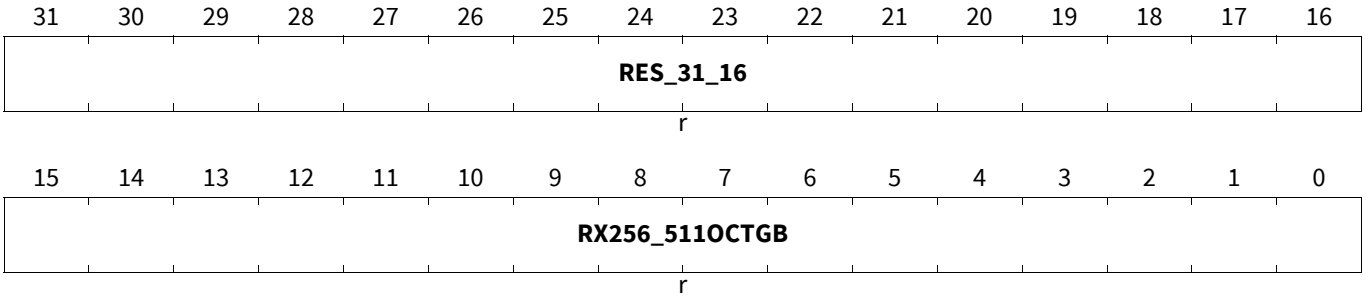


Field	Bits	Type	Description
RX128_255OCTGB	15:0	r	**Rx 128-255 Octets Packets Good Bad** This field indicates the number of good and bad packets received with length between 128 and 255 (inclusive) bytes, exclusive of the preamble. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 256to511 Octets Packets Received Count Register

This register provides the number of good and bad packets received by DWC_ether_qos with length between 256 and 511 (inclusive) bytes, exclusive of the preamble.

RX_256TO511OCTETS_PACKETS_GOOD_BAD
Good And Bad 256to511 Octets Packets Received Count Register($07B8_H$) Application Reset Value: 0000 0000_H



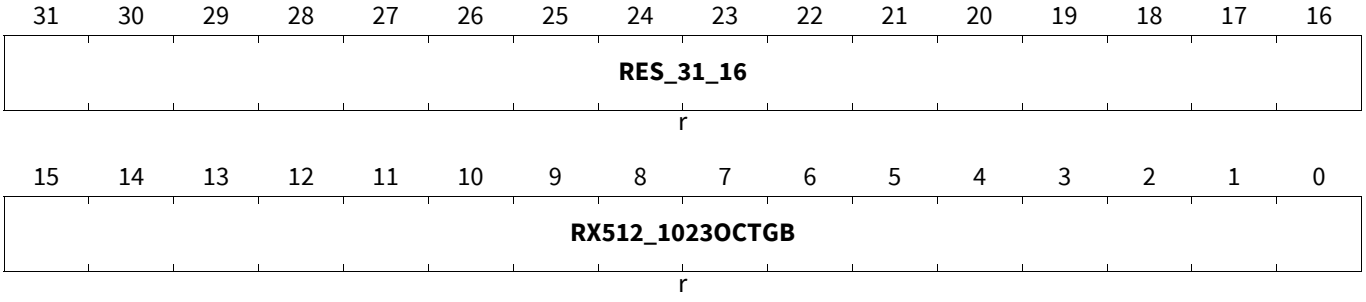
Field	Bits	Type	Description
RX256_511OCTGB	15:0	r	**Rx 256-511 Octets Packets Good Bad** This field indicates the number of good and bad packets received with length between 256 and 511 (inclusive) bytes, exclusive of the preamble. Value After Reset: 0x0

Field	Bits	Type	Description
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 512to1023 Octets Packets Received Count Register

This register provides the number of good and bad packets received by DWC_ether_qos with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble.

RX_512TO1023OCTETS_PACKETS_GOOD_BAD
Good And Bad 512to1023 Octets Packets Received Count Register(07BC$_H$) Application Reset Value: 0000 0000$_H$

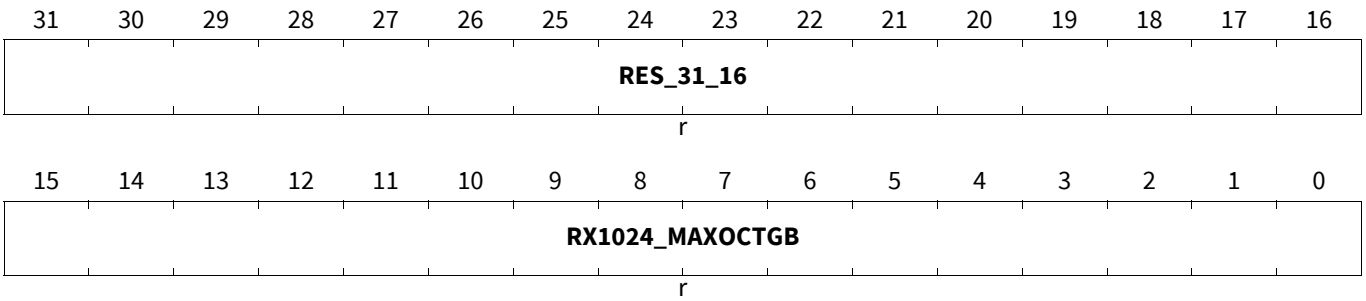


Field	Bits	Type	Description
RX512_1023OCTGB	15:0	r	**RX 512-1023 Octets Packets Good Bad** This field indicates the number of good and bad packets received with length between 512 and 1023 (inclusive) bytes, exclusive of the preamble. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad 1024toMax Octets Packets Received Count Register

This register provides the number of good and bad packets received by DWC_ether_qos with length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble.

RX_1024TOMAXOCTETS_PACKETS_GOOD_BAD
Good And Bad 1024toMax Octets Packets Received Count Register(07C0$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RX1024_MAXOCTGB	15:0	r	**Rx 1024-Max Octets Good Bad** This field indicates the number of good and bad packets received with length between 1024 and maxsize (inclusive) bytes, exclusive of the preamble. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received Unicat Packets Count Register

This register provides the number of good unicast packets received by DWC_ether_qos.

RX_UNICAST_PACKETS_GOOD
Good Received Unicat Packets Count Register (07C4$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUCASTG															
r															

Field	Bits	Type	Description
RXUCASTG	15:0	r	**Rx Unicast Packets Good** This field indicates the number of good unicast packets received. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received Length Error Packets Count Register

This register provides the number of packets received by DWC_ether_qos with length error (Length Type field not equal to packet size), for all packets with valid length field.

RX_LENGTH_ERROR_PACKETS
Received Length Error Packets Count Register (07C8$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLENERR															
r															

Field	Bits	Type	Description
RXLENERR	15:0	r	**Rx Length Error Packets** This field indicates the number of packets received with length error (Length Type field not equal to packet size), for all packets with valid length field. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received Out Of Range Type Count Register

This register provides the number of packets received by DWC_ether_qos with length field not equal to the valid packet size (greater than 1,500 but less than 1,536).

RX_OUT_OF_RANGE_TYPE_PACKETS
Received Out Of Range Type Count Register (07CC$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXOUTOFRNG															
r															

Field	Bits	Type	Description
RXOUTOFRNG	15:0	r	**Rx Out of Range Type Packet** This field indicates the number of packets received with length field not equal to the valid packet size (greater than 1,500 but less than 1,536). Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received Pause Packets Count Register

This register provides the number of good and valid Pause packets received by DWC_ether_qos.

RX_PAUSE_PACKETS
Received Pause Packets Count Register (07D0$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXPAUSEPKT															
r															

Field	Bits	Type	Description
RXPAUSEPKT	15:0	r	**Rx Pause Packets** This field indicates the number of good and valid Pause packets received. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received FIFO Overflow Count Register

This register provides the number of missed received packets because of FIFO overflow in DWC_ether_qos.

RX_FIFO_OVERFLOW_PACKETS
Received FIFO Overflow Count Register (07D4$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFIFOOVFL															
r															

Field	Bits	Type	Description
RXFIFOOVFL	15:0	r	**Rx FIFO Overflow Packets** This field indicates the number of missed received packets because of FIFO overflow. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good And Bad Received VLAN Packets Count Registerv

This register provides the number of good and bad VLAN packets received by DWC_ether_qos.

RX_VLAN_PACKETS_GOOD_BAD
Good And Bad Received VLAN Packets Count Registerv(07D8$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXVLANPKTGB															
r															

Field	Bits	Type	Description
RXVLANPKTGB	15:0	r	**Rx VLAN Packets Good Bad** This field indicates the number of good and bad VLAN packets received. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received Watchdog Error Count Register

This register provides the number of packets received by DWC_ether_qos with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when JE and WD bits are reset in MAC_Configuration register), 10,240 bytes (when JE bit is set and WD bit is reset in MAC_Configuration register), 16,384 bytes (when WD bit is set in MAC_Configuration register) or the value programmed in the MAC_Watchdog_Timeout register).

RX_WATCHDOG_ERROR_PACKETS

Received Watchdog Error Count Register (07DC$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWDGERR															
r															

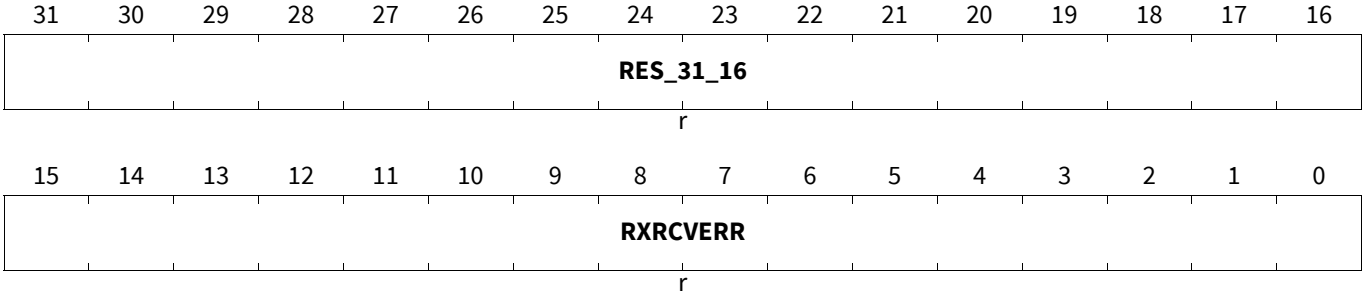
Field	Bits	Type	Description
RXWDGERR	15:0	r	**Rx Watchdog Error Packets** This field indicates the number of packets received with error because of watchdog timeout error (packets with a data load larger than 2,048 bytes (when JE and WD bits are reset in MAC_Configuration register), 10,240 bytes (when JE bit is set and WD bit is reset in MAC_Configuration register), 16,384 bytes (when WD bit is set in MAC_Configuration register) or the value programmed in the MAC_Watchdog_Timeout register). Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received Receive Error Count Register

This register provides the number of packets received by DWC_ether_qos with Receive error or Packet Extension error on the GMII or MII interface.

RX_RECEIVE_ERROR_PACKETS

Received Receive Error Count Register (07E0$_H$) Application Reset Value: 0000 0000$_H$



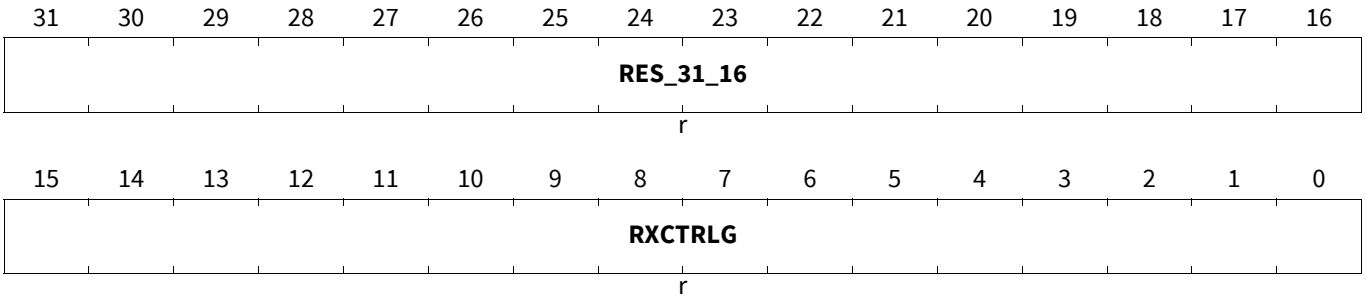
Field	Bits	Type	Description
RXRCVERR	15:0	r	**Rx Receive Error Packets** This field indicates the number of packets received with Receive error or Packet Extension error on the GMII or MII interface. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received Control Packets Count Register

This register provides the number of good control packets received by DWC_ether_qos.

RX_CONTROL_PACKETS_GOOD

Good Received Control Packets Count Register (07E4$_H$) Application Reset Value: 0000 0000$_H$



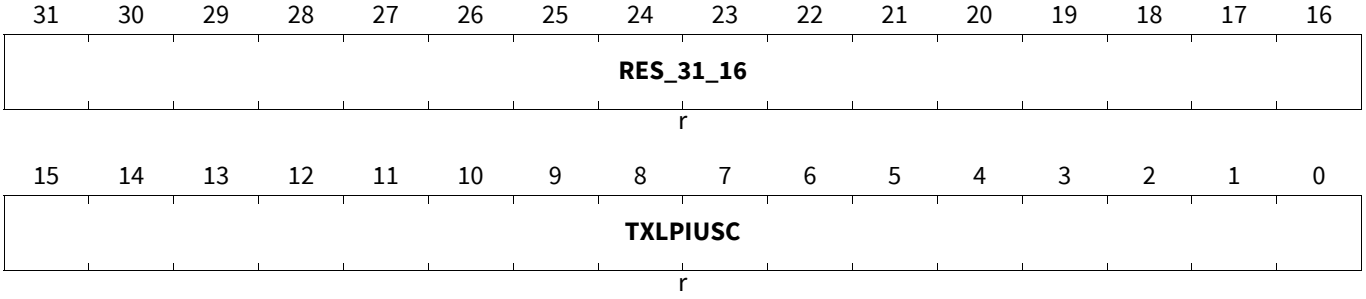
Field	Bits	Type	Description
RXCTRLG	15:0	r	**Rx Control Packets Good** This field indicates the number of good control packets received. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted LPI Microseconds Count Register

This register provides the number of microseconds Tx LPI is asserted by DWC_ether_qos.

TX_LPI_USEC_CNTR

Transmitted LPI Microseconds Count Register (07EC$_H$) Application Reset Value: 0000 0000$_H$



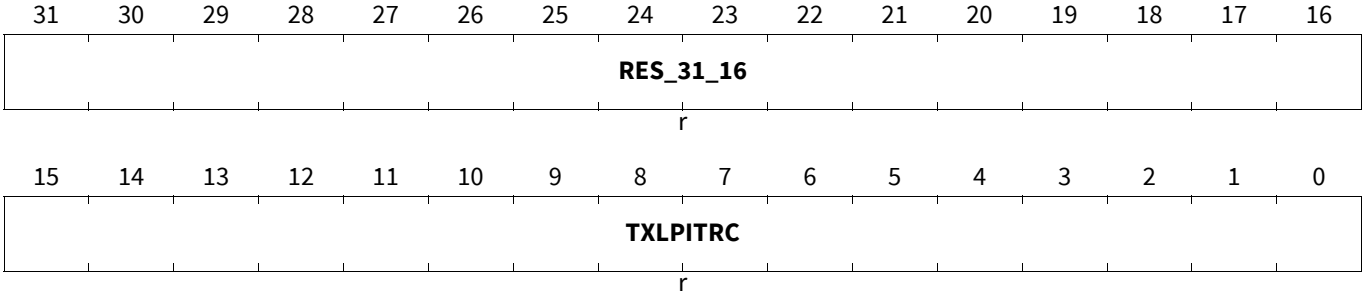
Field	Bits	Type	Description
TXLPIUSC	15:0	r	**Tx LPI Microseconds Counter** This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Transmitted LPI Transition Count Register

This register provides the number of times DWC_ether_qos has entered Tx LPI.

TX_LPI_TRAN_CNTR

Transmitted LPI Transition Count Register (07F0$_H$) Application Reset Value: 0000 0000$_H$



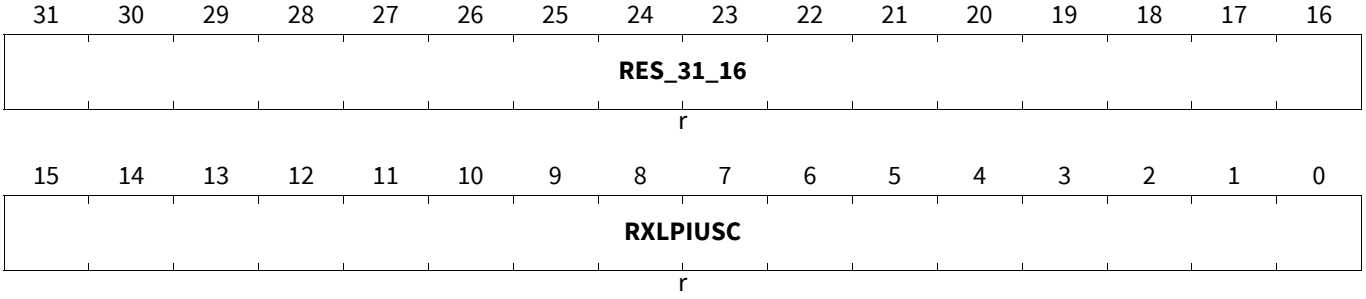
Field	Bits	Type	Description
TXLPITRC	15:0	r	**Tx LPI Transition counter** This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate Mode (because of LPITXA bit set in the LPI Control and Status register), the counter will increment. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received Microseconds LPI Count Register

This register provides the number of microseconds Rx LPI is sampled by DWC_ether_qos.

RX_LPI_USEC_CNTR

Received Microseconds LPI Count Register (07F4$_H$) Application Reset Value: 0000 0000$_H$



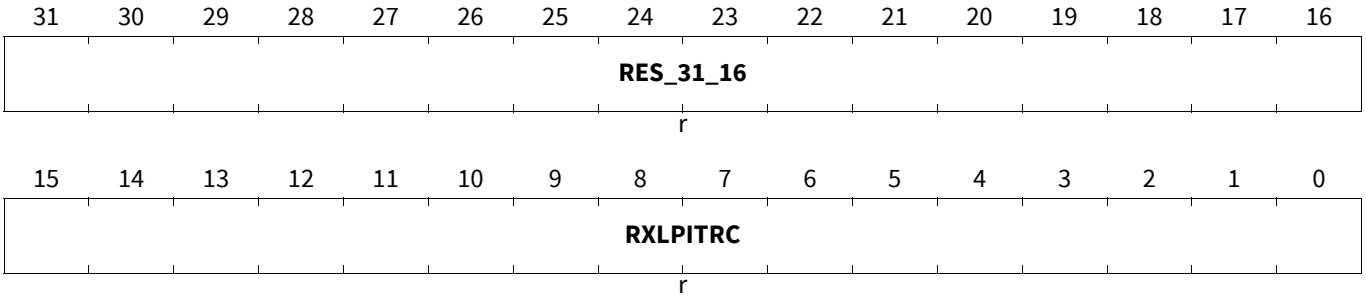
Field	Bits	Type	Description
RXLPIUSC	15:0	r	**Rx LPI Microseconds Counter** This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received LPI Transition Count Register

This register provides the number of times DWC_ether_qos has entered Rx LPI.

RX_LPI_TRAN_CNTR

Received LPI Transition Count Register (07F8$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
RXLPITRC	15:0	r	**Rx LPI Transition counter** This field indicates the number of times Rx LPI Entry has occurred. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

MMC IPC Receive Interrupts Mask Register

This register maintains the mask for the interrupt generated from the receive IPC statistic counters.

The MMC Receive Checksum Off load Interrupt Mask register maintains the masks for the interrupts generated when the receive IPC (Checksum Off load) statistic counters reach half their maximum value, and when they reach their maximum values. This register is 32 bits wide.

MMC_IPC_RX_INTERRUPT_MASK
MMC IPC Receive Interrupts Mask Register (0800_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_30		RXICMPEROIM	RXICMPGOIM	RXTCPEROIM	RXTCPGOIM	RXUDPEROIM	RXUDPGOIM	RXIPV6NOPAYOIM	RXIPV6HEROIM	RXIPV6GOIM	RXIPV4UDSBLOIM	RXIPV4FRAGOIM	RXIPV4NOPAYOIM	RXIPV4HEROIM	RXIPV4GOIM
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_14		RXICMPERPIM	RXICMPGPIM	RXTCPERPIM	RXTCPGPIM	RXUDPERPIM	RXUDPGPIM	RXIPV6NOPAYPIM	RXIPV6HERPIM	RXIPV6GPIM	RXIPV4UDSBLPIM	RXIPV4FRAGPIM	RXIPV4NOPAYPIM	RXIPV4HERPIM	RXIPV4GPIM
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
RXIPV4GPIM	0	rw	**MMC Receive IPV4 Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_gd_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV4HERPIM	1	rw	**MMC Receive IPV4 Header Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_hdrerr_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV4NOPAYPIM	2	rw	**MMC Receive IPV4 No Payload Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_nopay_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV4FRAGPIM	3	rw	**MMC Receive IPV4 Fragmented Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_frag_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV4UDSBLPIM	4	rw	**MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_udsbl_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV6GPIM	5	rw	**MMC Receive IPV6 Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_gd_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0

Field	Bits	Type	Description
RXIPV6HERPIM	6	rw	**MMC Receive IPV6 Header Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_hdrerr_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV6NOPAYPIM	7	rw	**MMC Receive IPV6 No Payload Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_nopay_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXUDPGPIM	8	rw	**MMC Receive UDP Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxudp_gd_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXUDPERPIM	9	rw	**MMC Receive UDP Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxudp_err_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXTCPGPIM	10	rw	**MMC Receive TCP Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxtcp_gd_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXTCPERPIM	11	rw	**MMC Receive TCP Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxtcp_err_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXICMPGPIM	12	rw	**MMC Receive ICMP Good Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxicmp_gd_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXICMPERPIM	13	rw	**MMC Receive ICMP Error Packet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxicmp_err_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RES_15_14	15:14	r	**Reserved** Value After Reset: 0x0
RXIPV4GOIM	16	rw	**MMC Receive IPV4 Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV4HEROIM	17	rw	**MMC Receive IPV4 Header Error Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0

Field	Bits	Type	Description
RXIPV4NOPAYOIM	18	rw	**MMC Receive IPV4 No Payload Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV4FRAGOIM	19	rw	**MMC Receive IPV4 Fragmented Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV4UDSBLOIM	20	rw	**MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV6GOIM	21	rw	**MMC Receive IPV6 Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV6HEROIM	22	rw	**MMC Receive IPV6 Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV6NOPAYOIM	23	rw	**MMC Receive IPV6 Header Error Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXUDPGOIM	24	rw	**MMC Receive IPV6 No Payload Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXUDPEROIM	25	rw	**MMC Receive UDP Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxudp_err_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXTCPGOIM	26	rw	**MMC Receive TCP Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxtcp_gd_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXTCPEROIM	27	rw	**MMC Receive TCP Error Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXICMPGOIM	28	rw	**MMC Receive ICMP Good Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0

Field	Bits	Type	Description
RXICMPEROIM	29	rw	**MMC Receive ICMP Error Octet Counter Interrupt Mask** Setting this bit masks the interrupt when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RES_31_30	31:30	r	**Reserved** Value After Reset: 0x0

MMC IPC Receive Interrupts Register

This register maintains the interrupt that the receive IPC statistic counters generate.

The MMC Receive Checksum Offload Interrupt register maintains the interrupts generated when receive IPC statistic counters reach half their maximum values (0x8000_0000 for 32 bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF_FFFF for 32 bit counter and 0xFFFF for 16 bit counter). When Counter Stop Rollover is set, the interrupts are set but the counter remains at all-ones.

The MMC Receive Checksum Offload Interrupt register is 32 bit wide. When the MMC IPC counter that caused the interrupt is read, its corresponding interrupt bit is cleared. The counter's least-significant byte lane (Bits[7:0]) must be read to clear the interrupt bit.

MMC_IPC_RX_INTERRUPT
MMC IPC Receive Interrupts Register **(0808_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_30		RXICMPEROIS	RXICMPGOIS	RXTCPEROIS	RXTCPGOIS	RXUDPEROIS	RXUDPGOIS	RXIPV6NOPAYOIS	RXIPV6HEROIS	RXIPV6GOIS	RXIPV4UDSBLOIS	RXIPV4FRAGOIS	RXIPV4NOPAYOIS	RXIPV4HEROIS	RXIPV4GOIS
r		r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_14		RXICMPERPIS	RXICMPGPIS	RXTCPERPIS	RXTCPGPIS	RXUDPERPIS	RXUDPGPIS	RXIPV6NOPAYPIS	RXIPV6HERPIS	RXIPV6GPIS	RXIPV4UDSBLPIS	RXIPV4FRAGPIS	RXIPV4NOPAYPIS	RXIPV4HERPIS	RXIPV4GPIS
r		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Field	Bits	Type	Description
RXIPV4GPIS	0	r	**MMC Receive IPV4 Good Packet Counter Interrupt Status** This bit is set when the rxipv4_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV4HERPIS	1	r	**MMC Receive IPV4 Header Error Packet Counter Interrupt Status** This bit is set when the rxipv4_hdrerr_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
RXIPV4NOPAYPIS	2	r	**MMC Receive IPV4 No Payload Packet Counter Interrupt Status** This bit is set when the rxipv4_nopay_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV4FRAGPIS	3	r	**MMC Receive IPV4 Fragmented Packet Counter Interrupt Status** This bit is set when the rxipv4_frag_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV4UDSBLPIS	4	r	**MMC Receive IPV4 UDP Checksum Disabled Packet Counter Interrupt Status** This bit is set when the rxipv4_udsbl_pkts counter reaches half of the maximum value or the maximum value. Value After Reset: 0x0
RXIPV6GPIS	5	r	**MMC Receive IPV6 Good Packet Counter Interrupt Status** This bit is set when the rxipv6_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV6HERPIS	6	r	**MMC Receive IPV6 Header Error Packet Counter Interrupt Status** This bit is set when the rxipv6_hdrerr_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV6NOPAYPIS	7	r	**MMC Receive IPV6 No Payload Packet Counter Interrupt Status** This bit is set when the rxipv6_nopay_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXUDPGPIS	8	r	**MC Receive UDP Good Packet Counter Interrupt Status** This bit is set when the rxudp_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXUDPERPIS	9	r	**MMC Receive UDP Error Packet Counter Interrupt Status** This bit is set when the rxudp_err_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXTCPGPIS	10	r	**MMC Receive TCP Good Packet Counter Interrupt Status** This bit is set when the rxtcp_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
RXTCPERPIS	11	r	**MMC Receive TCP Error Packet Counter Interrupt Status** This bit is set when the rxtcp_err_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXICMPGPIS	12	r	**MMC Receive ICMP Good Packet Counter Interrupt Status** This bit is set when the rxicmp_gd_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXICMPERPIS	13	r	**MMC Receive ICMP Error Packet Counter Interrupt Status** This bit is set when the rxicmp_err_pkts counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_15_14	15:14	r	**Reserved** Value After Reset: 0x0
RXIPV4GOIS	16	r	**MMC Receive IPV4 Good Octet Counter Interrupt Status** This bit is set when the rxipv4_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV4HEROIS	17	r	**MMC Receive IPV4 Header Error Octet Counter Interrupt Status** This bit is set when the rxipv4_hdrerr_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV4NOPAYOIS	18	r	**MMC Receive IPV4 No Payload Octet Counter Interrupt Status** This bit is set when the rxipv4_nopay_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV4FRAGOIS	19	r	**MMC Receive IPV4 Fragmented Octet Counter Interrupt Status** This bit is set when the rxipv4_frag_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV4UDSBLOIS	20	r	**MMC Receive IPV4 UDP Checksum Disabled Octet Counter Interrupt Status** This bit is set when the rxipv4_udsbl_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
RXIPV6GOIS	21	r	**MMC Receive IPV6 Good Octet Counter Interrupt Status** This bit is set when the rxipv6_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV6HEROIS	22	r	**MMC Receive IPV6 Header Error Octet Counter Interrupt Status** This bit is set when the rxipv6_hdrerr_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXIPV6NOPAYOIS	23	r	**MMC Receive IPV6 No Payload Octet Counter Interrupt Status** This bit is set when the rxipv6_nopay_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXUDPGOIS	24	r	**MMC Receive UDP Good Octet Counter Interrupt Status** This bit is set when the rxudp_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXUDPEROIS	25	r	**MMC Receive UDP Error Octet Counter Interrupt Status** This bit is set when the rxudp_err_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXTCPGOIS	26	r	**MMC Receive TCP Good Octet Counter Interrupt Status** This bit is set when the rxtcp_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXTCPEROIS	27	r	**MMC Receive TCP Error Octet Counter Interrupt Status** This bit is set when the rxtcp_err_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXICMPGOIS	28	r	**MMC Receive ICMP Good Octet Counter Interrupt Status** This bit is set when the rxicmp_gd_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RXICMPEROIS	29	r	**MMC Receive ICMP Error Octet Counter Interrupt Status** This bit is set when the rxicmp_err_octets counter reaches half of the maximum value or the maximum value. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

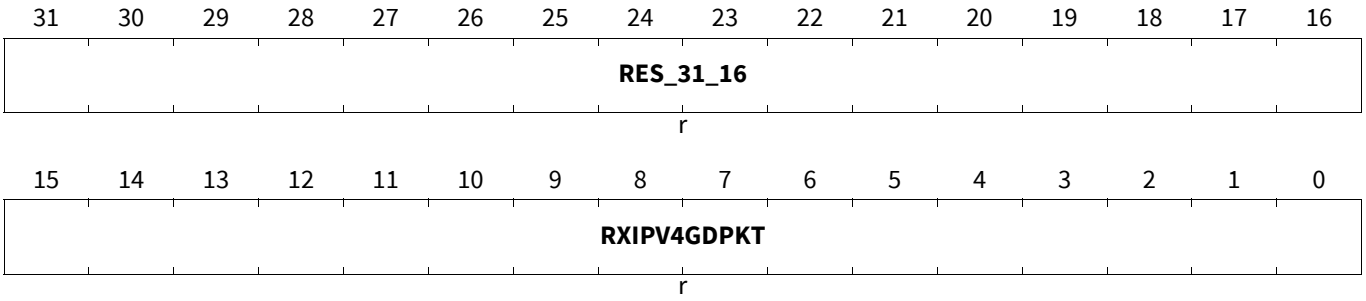
Field	Bits	Type	Description
RES_31_30	31:30	r	**Reserved** Value After Reset: 0x0

Good Received RxIPv4 Packets Count Register

This register provides the number of good IPv4 datagrams received by DWC_ether_qos with the TCP, UDP, or ICMP payload.

RXIPV4_GOOD_PACKETS

Good Received RxIPv4 Packets Count Register (0810_H) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RXIPV4GDPKT	15:0	r	**RxIPv4 Good Packets** This field indicates the number of good IPv4 datagrams received with the TCP, UDP, or ICMP payload. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

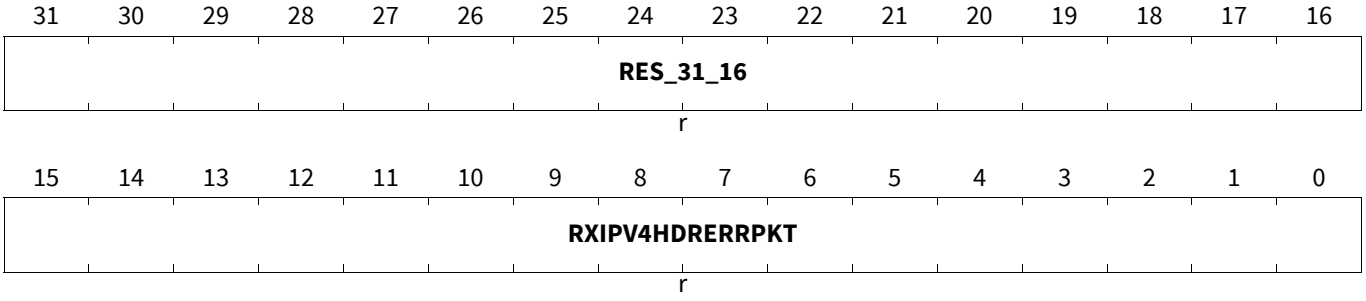
Received IPv4 Header Error Packets Count Register

RxIPv4 Header Error Packets

This register provides the number of IPv4 datagrams received by DWC_ether_qos with header (checksum, length, or version mismatch) errors.

RXIPV4_HEADER_ERROR_PACKETS

Received IPv4 Header Error Packets Count Register(0814_H) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RXIPV4HDRERRPKT	15:0	r	**RxIPv4 Header Error Packets** This field indicates the number of IPv4 datagrams received with header (checksum, length, or version mismatch) errors. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received IPv4 No Payload Packets Count Register

This register provides the number of IPv4 datagram packets received by DWC_ether_qos that did not have a TCP, UDP, or ICMP payload.

RXIPV4_NO_PAYLOAD_PACKETS

Received IPv4 No Payload Packets Count Register(0818_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIPV4NOPAYPKT															
r															

Field	Bits	Type	Description
RXIPV4NOPAYPKT	15:0	r	**RxIPv4 Payload Packets** This field indicates the number of IPv4 datagram packets received that did not have a TCP, UDP, or ICMP payload. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received IPv4 Fragmented Packets Count Register

This register provides the number of good IPv4 datagrams received by DWC_ether_qos with fragmentation.

RXIPV4_FRAGMENTED_PACKETS

Received IPv4 Fragmented Packets Count Register($081C_H$) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIPV4FRAGPKT															
r															

Field	Bits	Type	Description
RXIPV4FRAGPKT	15:0	r	**RxIPv4 Fragmented Packets** This field indicates the number of good IPv4 datagrams received with fragmentation. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received IPv4 UPD Checksum Disabled Packets Count Register

This register provides the number of good IPv4 datagrams received by DWC_ether_qos that had a UDP payload with checksum disabled.

RXIPV4_UDP_CHECKSUM_DISABLED_PACKETS
Received IPv4 UPD Checksum Disabled Packets Count Register(0820_H)Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIPV4UDSBLPKT															
r															

Field	Bits	Type	Description
RXIPV4UDSBLPKT	15:0	r	**RxIPv4 UDP Checksum Disabled Packets** This field indicates the number of good IPv4 datagrams received that had a UDP payload with checksum disabled. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received RxIPv6 Packets Count Register

This register provides the number of good IPv6 datagrams received by DWC_ether_qos with the TCP, UDP, or ICMP payload.

RXIPV6_GOOD_PACKETS
Good Received RxIPv6 Packets Count Register (0824_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIPV6GDPKT															
r															

Field	Bits	Type	Description
RXIPV6GDPKT	15:0	r	**RxIPv6 Good Packets** This field indicates the number of good IPv6 datagrams received with the TCP, UDP, or ICMP payload. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received IPv6 Header Error Packets Count Register

This register provides the number of IPv6 datagrams received by DWC_ether_qos with header (length or version mismatch) errors.

RXIPV6_HEADER_ERROR_PACKETS

Received IPv6 Header Error Packets Count Register(0828_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIPV6HDRERRPKT															
r															

Field	Bits	Type	Description
RXIPV6HDRERRPKT	15:0	r	**RxIPv6 Header Error Packets** This field indicates the number of IPv6 datagrams received with header (length or version mismatch) errors. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received IPv6 No Payload Packets Count Register

This register provides the number of IPv6 datagram packets received by DWC_ether_qos that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers.

RXIPV6_NO_PAYLOAD_PACKETS

Received IPv6 No Payload Packets Count Register($082C_H$) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIPV6NOPAYPKT															
r															

Field	Bits	Type	Description
RXIPV6NOPAYPKT	15:0	r	**RxIPv6 Payload Packets** This field indicates the number of IPv6 datagram packets received that did not have a TCP, UDP, or ICMP payload. This includes all IPv6 datagrams with fragmentation or security extension headers. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received UDP Packets Count Register

This register provides the number of good IP datagrams received by DWC_ether_qos with a good UDP payload. This counter is not updated when the RxIPv4_UDP_Checksum_Disabled_Packets counter is incremented.

RXUDP_GOOD_PACKETS
Good Received UDP Packets Count Register (0830$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUDPGDPKT															
r															

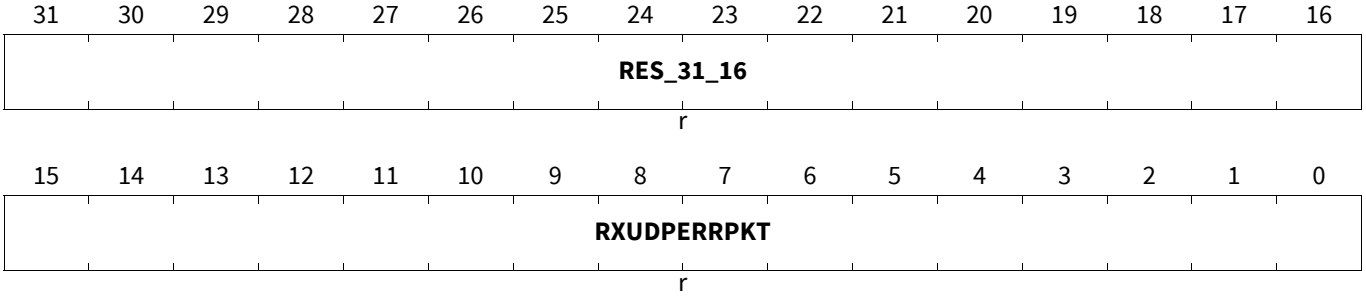
Field	Bits	Type	Description
RXUDPGDPKT	15:0	r	**RxUDP Good Packets** This field indicates the number of good IP datagrams received with a good UDP payload. This counter is not updated when the RxIPv4_UDP_Checksum_Disabled_Packets counter is incremented. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received UDP Error Packets Count Register

This register provides the number of good IP datagrams received by DWC_ether_qos whose UDP payload has a checksum error.

RXUDP_ERROR_PACKETS

Received UDP Error Packets Count Register (0834$_H$) Application Reset Value: 0000 0000$_H$



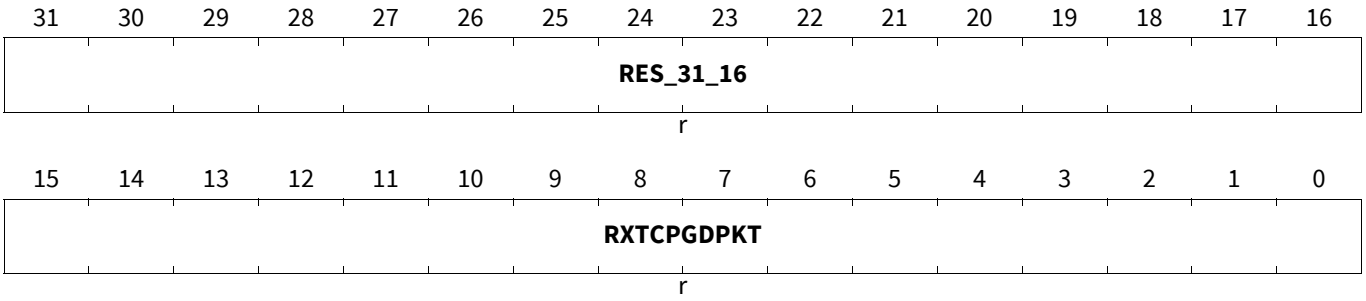
Field	Bits	Type	Description
RXUDPERRPKT	15:0	r	**RxUDP Error Packets** This field indicates the number of good IP datagrams received whose UDP payload has a checksum error. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received TCP Packets Count Register

This register provides the number of good IP datagrams received by DWC_ether_qos with a good TCP payload.

RXTCP_GOOD_PACKETS

Good Received TCP Packets Count Register (0838$_H$) Application Reset Value: 0000 0000$_H$



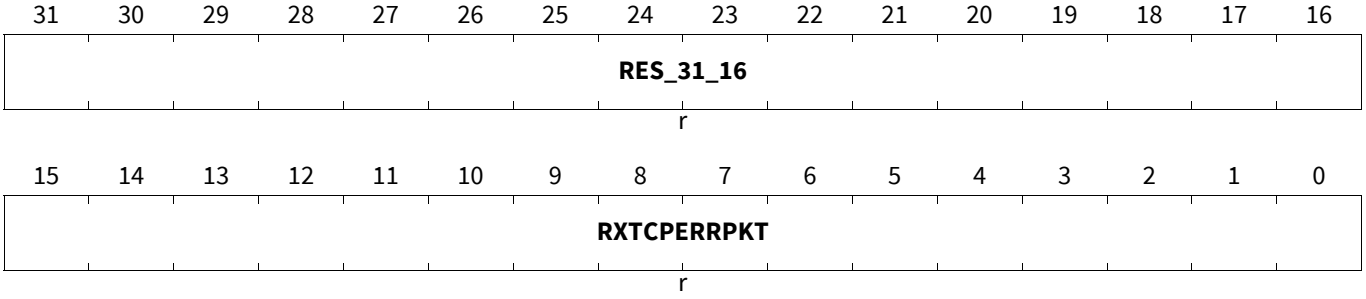
Field	Bits	Type	Description
RXTCPGDPKT	15:0	r	**RxTCP Good Packets** This field indicates the number of good IP datagrams received with a good TCP payload. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received TCP Error Packets Count Register

This register provides the number of good IP datagrams received by DWC_ether_qos whose TCP payload has a checksum error.

RXTCP_ERROR_PACKETS

Received TCP Error Packets Count Register (083C$_H$) Application Reset Value: 0000 0000$_H$



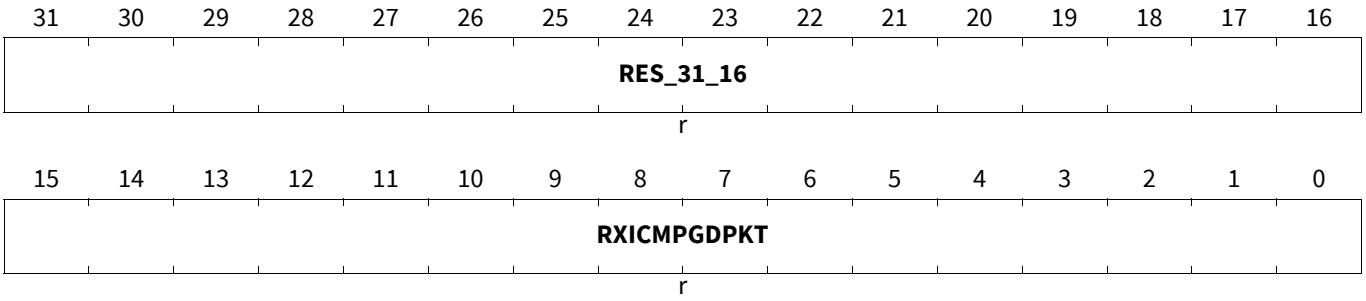
Field	Bits	Type	Description
RXTCPERRPKT	15:0	r	**RxTCP Error Packets** This field indicates the number of good IP datagrams received whose TCP payload has a checksum error. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received ICMP Packets Count Register

This register provides the number of good IP datagrams received by DWC_ether_qos with a good ICMP payload.

RXICMP_GOOD_PACKETS

Good Received ICMP Packets Count Register (0840$_H$) Application Reset Value: 0000 0000$_H$



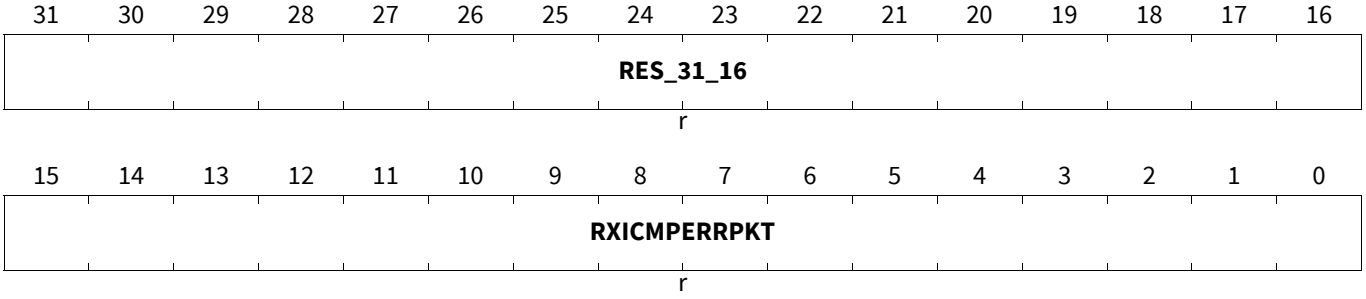
Field	Bits	Type	Description
RXICMPGDPKT	15:0	r	**RxICMP Good Packets** This field indicates the number of good IP datagrams received with a good ICMP payload. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Received ICMP Error Packets Count Register

This register provides the number of good IP datagrams received by DWC_ether_qos whose ICMP payload has a checksum error.

RXICMP_ERROR_PACKETS

Received ICMP Error Packets Count Register (0844$_H$) Application Reset Value: 0000 0000$_H$



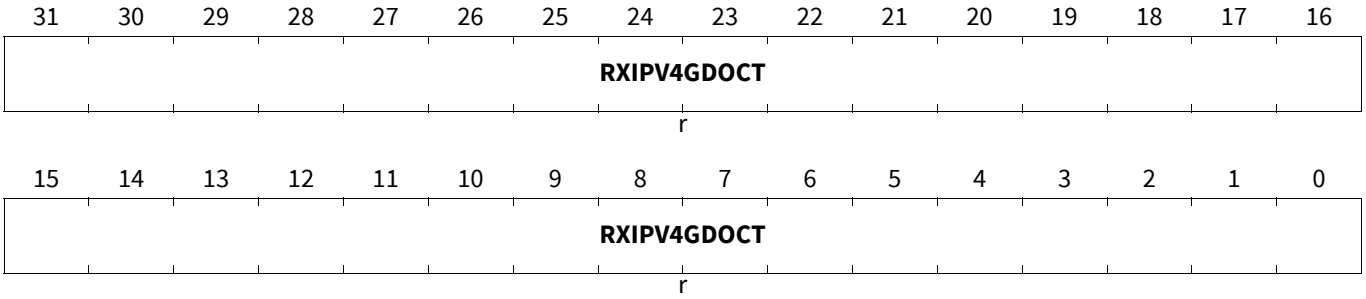
Field	Bits	Type	Description
RXICMPERRPKT	15:0	r	**RxICMP Error Packets** This field indicates the number of good IP datagrams received whose ICMP payload has a checksum error. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

Good Received IPV4 Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV4_GOOD_OCTETS

Good Received IPV4 Octets Count Register (0850$_H$) Application Reset Value: 0000 0000$_H$



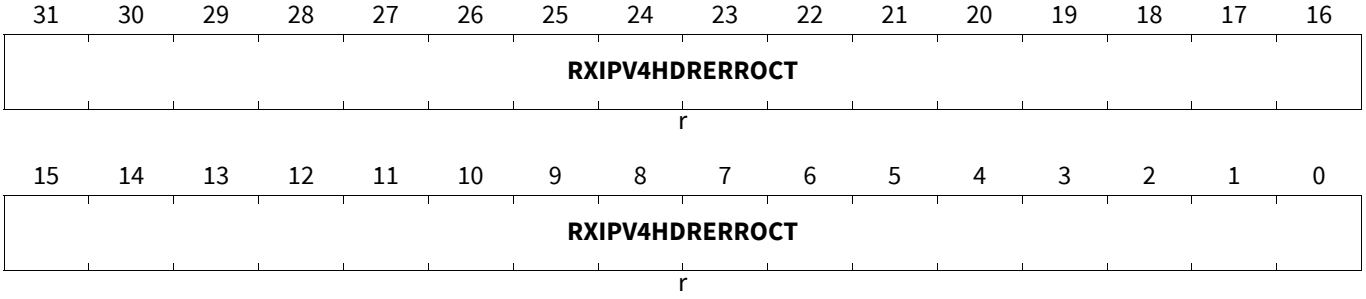
Field	Bits	Type	Description
RXIPV4GDOCT	31:0	r	**RxIPv4 Good Octets** This field indicates the number of bytes received in good IPv4 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter. Value After Reset: 0x0

Received IPV4 Header Error Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV4_HEADER_ERROR_OCTETS

Received IPV4 Header Error Octets Count Register(0854_H) **Application Reset Value: $0000\ 0000_H$**



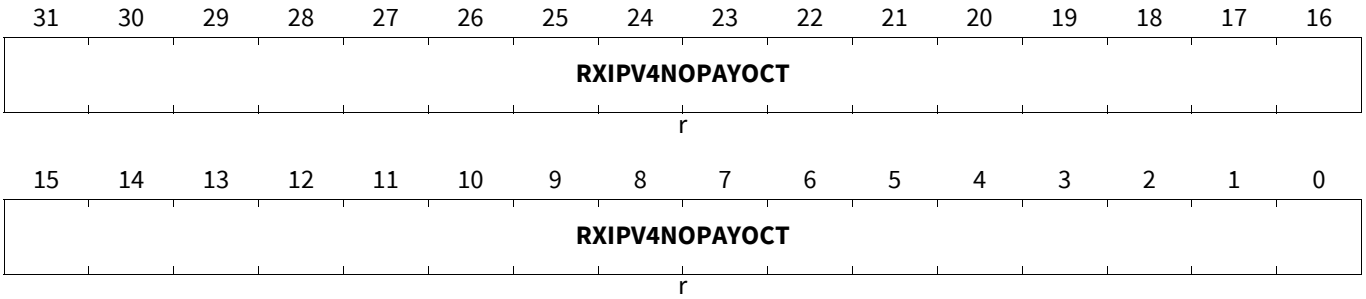
Field	Bits	Type	Description
RXIPV4HDRERROCT	31:0	r	**RxIPv4 Header Error Octets** This field indicates the number of bytes received in IPv4 datagrams with header errors (checksum, length, version mismatch). The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter. Value After Reset: 0x0

Received IPV4 No Payload Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV4_NO_PAYLOAD_OCTETS

Received IPV4 No Payload Octets Count Register(0858_H) **Application Reset Value: $0000\ 0000_H$**



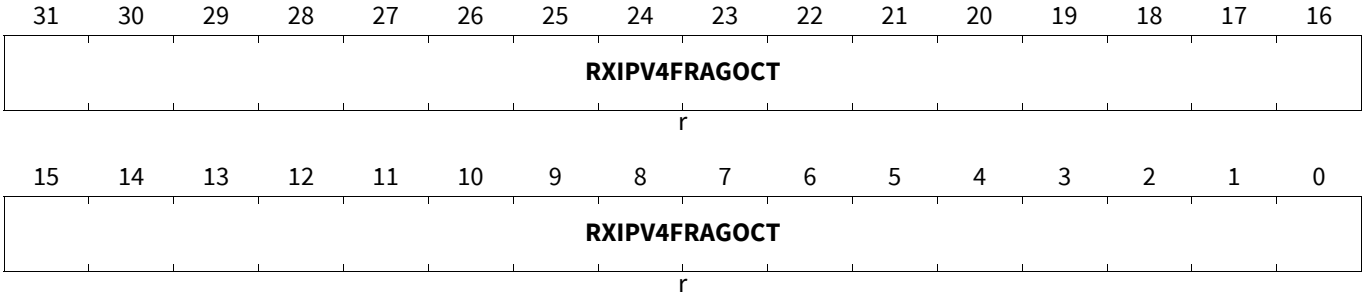
Field	Bits	Type	Description
RXIPV4NOPAYOCT	31:0	r	**RxIPv4 Payload Octets** This field indicates the number of bytes received in IPv4 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter. Value After Reset: 0x0

Received IPV4 Fragmented Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV4_FRAGMENTED_OCTETS

Received IPV4 Fragmented Octets Count Register(085C$_H$) **Application Reset Value: 0000 0000$_H$**



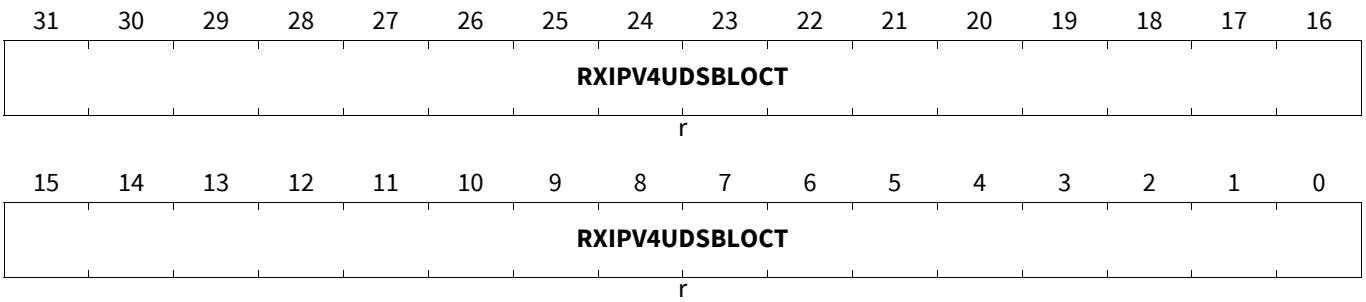
Field	Bits	Type	Description
RXIPV4FRAGOCT	31:0	r	**RxIPv4 Fragmented Octets** This field indicates the number of bytes received in fragmented IPv4 datagrams. The value in the Length field of IPv4 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter. Value After Reset: 0x0

Received IPV4 UPD Checksum Disabled Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV4_UDP_CHECKSUM_DISABLE_OCTETS

Received IPV4 UPD Checksum Disabled Octets Count Register(0860$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RXIPV4UDSBLOCT	31:0	r	**RxIPv4 UDP Checksum Disable Octets** This field indicates the number of bytes received in a UDP segment that had the UDP checksum disabled. This counter does not count IP Header bytes. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter. Value After Reset: 0x0

Good Received IPV6 Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV6_GOOD_OCTETS
Good Received IPV6 Octets Count Register (0864$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXIPV6GDOCT															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIPV6GDOCT															
r															

Field	Bits	Type	Description
RXIPV6GDOCT	31:0	r	**RxIPv6 Good Octets** This field indicates the number of bytes received in good IPv6 datagrams encapsulating TCP, UDP, or ICMP data. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter. Value After Reset: 0x0

Received IPV6 Header Error Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV6_HEADER_ERROR_OCTETS
Received IPV6 Header Error Octets Count Register(0868$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXIPV6HDRERROCT															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIPV6HDRERROCT															
r															

Field	Bits	Type	Description
RXIPV6HDRERROCT	31:0	r	**RxIPv6 Header Error Octets** This field indicates the number of bytes received in IPv6 datagrams with header errors (length, version mismatch). The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter. Value After Reset: 0x0

Received IPV6 No Payload Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter.

RXIPV6_NO_PAYLOAD_OCTETS

Received IPV6 No Payload Octets Count Register($086C_H$) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXIPV6NOPAYOCT															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXIPV6NOPAYOCT															
r															

Field	Bits	Type	Description
RXIPV6NOPAYOCT	31:0	r	**RxIPv6 Payload Octets** This field indicates the number of bytes received in IPv6 datagrams that did not have a TCP, UDP, or ICMP payload. The value in the Length field of IPv6 header is used to update this counter. (Ethernet header, FCS, pad, or IP pad bytes are not included in this counter. Value After Reset: 0x0

Good Received UDP Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in a good UDP segment. This counter does not count IP header bytes.

RXUDP_GOOD_OCTETS

Good Received UDP Octets Count Register (0870_H) **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXUDPGDOCT															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUDPGDOCT															
r															

Field	Bits	Type	Description
RXUDPGDOCT	31:0	r	**RxUDP Good Octets** This field indicates the number of bytes received in a good UDP segment. This counter does not count IP header bytes. Value After Reset: 0x0

Received UDP Error Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in a UDP segment that had checksum errors. This counter does not count IP header bytes.

RXUDP_ERROR_OCTETS
Received UDP Error Octets Count Register (0874$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXUDPERROCT															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUDPERROCT															
r															

Field	Bits	Type	Description
RXUDPERROCT	31:0	r	**RxUDP Error Octets** This field indicates the number of bytes received in a UDP segment that had checksum errors. This counter does not count IP header bytes. Value After Reset: 0x0

Good Received TCP Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in a good TCP segment. This counter does not count IP header bytes.

RXTCP_GOOD_OCTETS
Good Received TCP Octets Count Register (0878$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXTCPGDOCT															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXTCPGDOCT															
r															

Field	Bits	Type	Description
RXTCPGDOCT	31:0	r	**RxTCP Good Octets** This field indicates the number of bytes received in a good TCP segment. This counter does not count IP header bytes. Value After Reset: 0x0

Received TCP Error Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in a TCP segment that had checksum errors. This counter does not count IP header bytes.

RXTCP_ERROR_OCTETS
Received TCP Error Octets Count Register (087C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXTCPERROCT															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXTCPERROCT															
r															

Field	Bits	Type	Description
RXTCPERROCT	31:0	r	**RxTCP Error Octets** This field indicates the number of bytes received in a TCP segment that had checksum errors. This counter does not count IP header bytes. Value After Reset: 0x0

Good Received ICMP Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in a good ICMP segment. This counter does not count IP header bytes.

RXICMP_GOOD_OCTETS
Good Received ICMP Octets Count Register (0880$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXICMPGDOCT															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXICMPGDOCT															
r															

Field	Bits	Type	Description
RXICMPGDOCT	31:0	r	**RxICMP Good Octets** This field indicates the number of bytes received in a good ICMP segment. This counter does not count IP header bytes. Value After Reset: 0x0

Received ICMP Error Octets Count Register

This register provides the number of bytes received by DWC_ether_qos in a ICMP segment that had checksum errors. This counter does not count IP header bytes.

RXICMP_ERROR_OCTETS
Received ICMP Error Octets Count Register **(0884_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXICMPERROCT															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXICMPERROCT															
r															

Field	Bits	Type	Description
RXICMPERROCT	31:0	r	**RxICMP Error Octets** This field indicates the number of bytes received in a ICMP segment that had checksum errors. This counter does not count IP header bytes. Value After Reset: 0x0

MAC Timestamp Control Register

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the Receiver.

MAC_TIMESTAMP_CONTROL
MAC Timestamp Control Register **($0B00_H$)** **Application Reset Value: $0000\ 2000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_29			AV8021ASMEN	RES_27_25			TXTSSTSM	RES_23_21			RES_20	CSC	TSENMACADDR	SNAPTYPSEL	
r			rw	r			rw	r			r	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	RES_7_6		TSADDREG	RES_4	TSUPDT	TSINIT	TSCFUPDT	TSENA
rw	rw	rw	rw	rw	rw	rw	rw	r		rw	r	rw	rw	rw	rw

Field	Bits	Type	Description
TSENA	0	rw	**Enable Timestamp** When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode. On the Receive side, the MAC processes the 1588 packets only if this bit is set. Value After Reset: 0x0
TSCFUPDT	1	rw	**Fine or Coarse Timestamp Update** When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp. Value After Reset: 0x0
TSINIT	2	rw	**Initialize Timestamp** When this bit is set, the system time is initialized (overwritten) with the value specified in the MAC Register 80 (System Time Seconds Update Register) and MAC Register 81 (System Time Nanoseconds Update Register). This bit should be zero before it is updated. This bit is reset when the initialization is complete. The Timestamp Higher Word register (if enabled during core configuration) can only be initialized. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
TSUPDT	3	rw	**Update Timestamp** When this bit is set, the system time is updated (added or subtracted) with the value specified in MAC_System_Time_Seconds_Update and MAC_System_Time_Nanoseconds_Update. This bit should be zero before updating it. This bit is reset when the update is complete in hardware. The Timestamp Higher Word register (if enabled during core configuration) is not updated. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
RES_4	4	r	**Reserved for future use.**
TSADDREG	5	rw	**Update Addend Register** When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
RES_7_6	7:6	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
TSENALL	8	rw	**Enable Timestamp for All Packets** When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC. Value After Reset: 0x0
TSCTRLSSR	9	rw	**Timestamp Digital or Binary Rollover Control** When this bit is set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit. Value After Reset: 0x0
TSVER2ENA	10	rw	**Enable PTP Packet Processing for Version 2 Format** When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'. Value After Reset: 0x0
TSIPENA	11	rw	**Enable Processing of PTP over Ethernet Packets** When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets. Value After Reset: 0x0
TSIPV6ENA	12	rw	**Enable Processing of PTP Packets Sent over IPv6-UDP** When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets. Value After Reset: 0x0
TSIPV4ENA	13	rw	**Enable Processing of PTP Packets Sent over IPv4-UDP** When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default. Value After Reset: 0x1
TSEVNTENA	14	rw	**Enable Timestamp Snapshot for Event Messages** When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see Timestamp Snapshot Dependency on Register Bits Table. Value After Reset: 0x0
TSMSTRENA	15	rw	**Enable Snapshot for Messages Relevant to Master** When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node. Value After Reset: 0x0

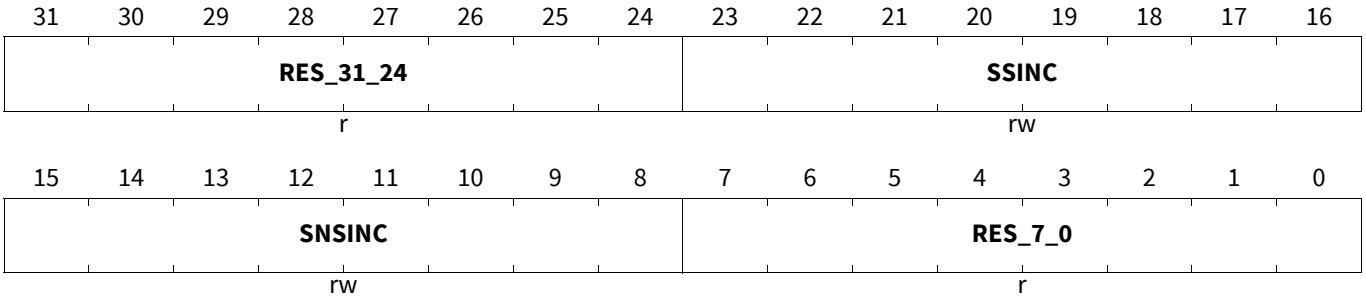
Field	Bits	Type	Description
SNAPTYPSEL	17:16	rw	**Select PTP packets for Taking Snapshots** These bits, along with Bits 15 and 14, decide the set of PTP packet types for which snapshot needs to be taken. The encoding is given in Timestamp Snapshot Dependency on Register Bits Table. Value After Reset: 0x0
TSENMACADDR	18	rw	**Enable MAC Address for PTP Packet Filtering** When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet. When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet. For normal time stamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching. For PTP offload, only MAC address register 0 is considered for unicast destination address matching. Value After Reset: 0x0
CSC	19	rw	**Enable checksum correction during OST for PTP over UDP/IPv4 packets** When this bit is set, the last two bytes of PTP message sent over UDP/IPv4 is updated to keep the UDP checksum correct, for changes made to origin timestamp and/or correction field as part of one step timestamp operation. The application shall form the packet with these two dummy bytes. When reset, no updates are done to keep the UDP checksum correct. The application shall form the packet with UDP checksum set to 0. Value After Reset: 0x0
RES_20	20	r	**Reserved for future use.**
RES_23_21	23:21	r	**Reserved** Value After Reset: 0x0
TXTSSTSM	24	rw	**Transmit Timestamp Status Mode** When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the MAC_TxTimestamp_Status_Nanoseconds register. When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the MAC_TxTimestamp_Status_Nanoseconds register. Value After Reset: 0x0
RES_27_25	27:25	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
AV8021ASMEN	28	rw	**AV 802.1AS Mode Enable** When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS mode of operation. When PTP offload feature is enabled, for the purpose of PTP offload, the transport specific field in the PTP header is generated and checked based on the value of this bit. Value After Reset: 0x0
RES_31_29	31:29	r	**Reserved** Value After Reset: 0x0

MAC Sub-Second Increment Register

This register specifies the value to be added to the internal system time every cycle of clk_ptp_ref_i clock.

MAC_SUB_SECOND_INCREMENT
MAC Sub-Second Increment Register **($0B04_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RES_7_0	7:0	r	**Reserved** Value After Reset: 0x0
SNSINC	15:8	rw	**Sub-nanosecond Increment Value** This field contains the sub-nanosecond increment value, represented in nanoseconds multiplied by 2^8. This value is accumulated with the sub-nanoseconds field of the subsecond register. For example, when TSCTRLSSR field in the MAC_Timestamp_Control register is set. and if the required increment is 5.3ns, then SSINC should be 0x05 and SNSINC should be 0x4C. Value After Reset: 0x0

Field	Bits	Type	Description
SSINC	23:16	rw	**Sub-second Increment Value** The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the sub-second register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [Bit 9 (TSCTRLSSR) is set in MAC_Timestamp_Control]. When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465. Value After Reset: 0x0
RES_31_24	31:24	r	**Reserved** Value After Reset: 0x0

MAC System Time Seconds Register

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

MAC_SYSTEM_TIME_SECONDS

MAC System Time Seconds Register (0B08$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS															
r															

Field	Bits	Type	Description
TSS	31:0	r	**Timestamp Second** The value in this field indicates the current value in seconds of the System Time maintained by the MAC. Value After Reset: 0x0

MAC System Time Nanoseconds Register

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

MAC_SYSTEM_TIME_NANOSECONDS

MAC System Time Nanoseconds Register (0B0C$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31	TSSS														
r	r														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS															
r															

Field	Bits	Type	Description
TSSS	30:0	r	**Timestamp Sub Seconds** The value in this field has the sub-second representation of time, with an accuracy of 0.46 ns. When Bit 9 is set in MAC_Timestamp_Control, each bit represents 1 ns. The maximum value is 0x3B9A_C9FF after which it rolls-over to zero. Value After Reset: 0x0
RES_31	31	r	**Reserved** Value After Reset: 0x0

MAC System Time Seconds Update Register

The System Time Seconds Update register, along with the System Time Nanoseconds Update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in DWC_eqos_top_map/EQOS_MAC/MAC_Timestamp_Control.

MAC_SYSTEM_TIME_SECONDS_UPDATE

MAC System Time Seconds Update Register (0B10$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS															
rw															

Field	Bits	Type	Description
TSS	31:0	rw	**Timestamp Seconds** The value in this field is the seconds part of the update. When ADDSUB is reset, this field must be programmed with the seconds part of the update value. When ADDSUB is set, this field must be programmed with the complement of the seconds part of the update value. For example, if 2.000000001 seconds need to be subtracted from the system time, the TSS field in the MAC_Timestamp_Seconds_Update register must be 0xFFFF_FFFE (that is, 2^32 - 2). Value After Reset: 0x0

MAC System Time Nanoseconds Update Register

MAC System Time Nanoseconds Update register

MAC_SYSTEM_TIME_NANOSECONDS_UPDATE

MAC System Time Nanoseconds Update Register($0B14_H$) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDSUB	**TSSS**														
rw	rw														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS															
rw															

Field	Bits	Type	Description
TSSS	30:0	rw	**Timestamp Sub Seconds** The value in this field is the sub seconds part of the update. When ADDSUB is reset, this field must be programmed with the sub-seconds part of the update value, with an accuracy based on the TSCTRLSSR bit of the MAC_Timestamp_Control register. When ADDSUB is set, this field must be programmed with the complement of the sub-seconds part of the update value as described below. When TSCTRLSSR bit in MAC_Timestamp_Control is set, the programmed value must be 10^9 - <sub-second_value>. When TSCTRLSSR bit in MAC_Timestamp_Control is reset, the programmed value must be 2^31 - <sub-second_value>. When the TSCTRLSSR bit is reset in the MAC_Timestamp_Control register, each bit represents an accuracy of 0.46 ns. When the TSCTRLSSR bit is set in the MAC_Timestamp_Control register, each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF. For example, if 2.000000001 seconds need to be subtracted from the system time, then the TSSS field in the MAC_Timestamp_Nanoseconds_Update register must be 0x7FFF_FFFF (that is, 2^31 - 1), when TSCTRLSSR bit in MAC_Timestamp_Control is reset and 0x3B9A_C9FF (that is, 10^9 - 1), when TSCTRLSSR bit in MAC_Timestamp_Control is set. Value After Reset: 0x0
ADDSUB	31	rw	**Add or Subtract Time** When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register. Value After Reset: 0x0

MAC Timestamp Addend Register

The Timestamp Addend register. This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the MAC_Timestamp_Control register). The content of this register is added to a 32-bit accumulator in every clock cycle (of clk_ptp_ref_i) and the system time is updated whenever the accumulator overflows.

MAC_TIMESTAMP_ADDEND

MAC Timestamp Addend Register **($0B18_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSAR															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSAR															
rw															

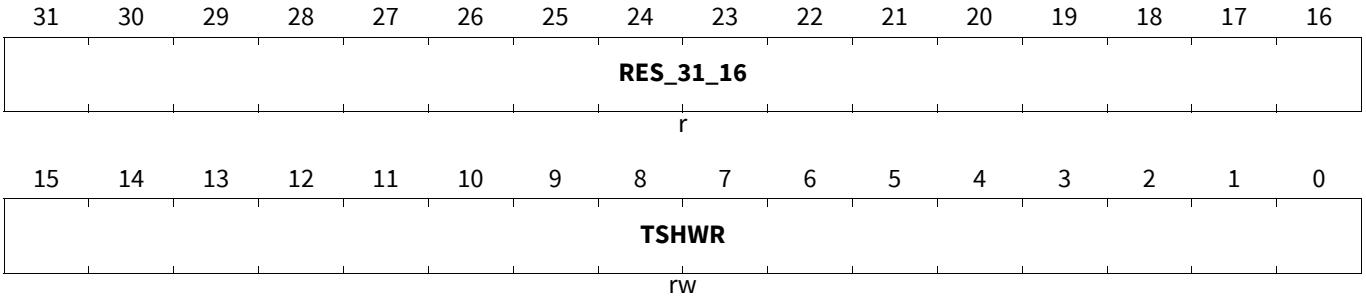
Field	Bits	Type	Description
TSAR	31:0	rw	**Timestamp Addend Register** This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization. Value After Reset: 0x0

MAC System Time Higher Word Seconds Register

The System Time - Higher Word Seconds register.

MAC_SYSTEM_TIME_HIGHER_WORD_SECONDS

MAC System Time Higher Word Seconds Register($0B1C_H$) **Application Reset Value: $0000\ 0000_H$**



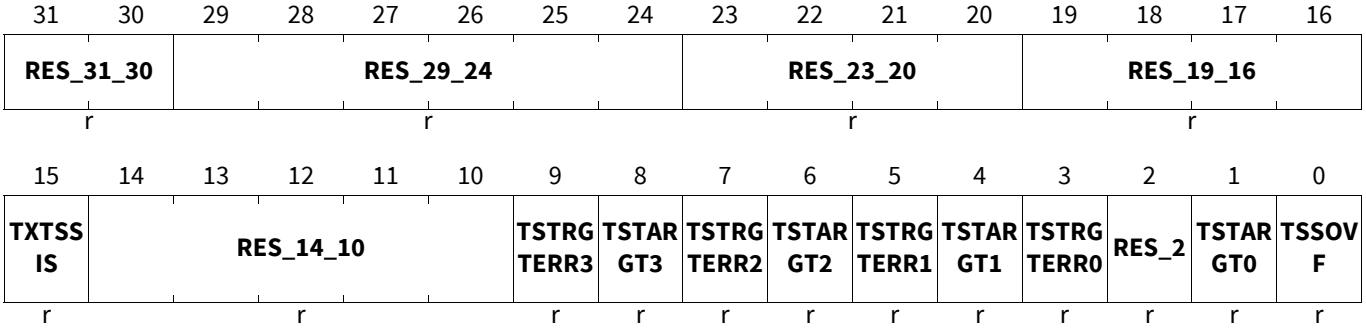
Field	Bits	Type	Description
TSHWR	15:0	rw	**Timestamp Higher Word Register** This field contains the most-significant 16-bits of timestamp seconds value. This register is optional. You can add this register by selecting the Add IEEE 1588 Higher Word Register option. This register is directly written to initialize the value and it is incremented when there is an overflow from 32-bits of the System Time Seconds register. Access restriction applies. Updated based on the event. Setting 1 sets. Setting 0 clears. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

MAC Timestamp Status Register

The Timestamp Status register. All bits except Bits[27:25] gets cleared when the application reads this register.

MAC_TIMESTAMP_STATUS

MAC Timestamp Status Register **(0B20$_H$)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
TSSOVF	0	r	**Timestamp Seconds Overflow** When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
TSTARGT0	1	r	**Timestamp Target Time Reached** When set, this bit indicates that the value of system time is greater than or equal to the value specified in the MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds registers. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
RES_2	2	r	**Reserved for future use.**
TSTRGTERR0	3	r	**Timestamp Target Time Error** This bit is set when the latest target time programmed in the MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds registers elapses. This bit is cleared when the application reads this bit. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
TSTARGT1	4	r	**Timestamp Target Time Reached for Target Time PPS1** When set, this bit indicates that the value of system time is greater than or equal to the value specified in the MAC_PPS1_Target_Time_Seconds and MAC_PPS1_Target_Time_Nanoseconds registers. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0

Field	Bits	Type	Description
TSTRGTERR1	5	r	**Timestamp Target Time Error** This bit is set when the latest target time programmed in the MAC_PPS1_Target_Time_Seconds and MAC_PPS1_Target_Time_Nanoseconds registers elapses. This bit is cleared when the application reads this bit. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
TSTARGT2	6	r	**Timestamp Target Time Reached for Target Time PPS2** When set, this bit indicates that the value of system time is greater than or equal to the value specified in the MAC_PPS2_Target_Time_Seconds and MAC_PPS2_Target_Time_Nanoseconds registers. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
TSTRGTERR2	7	r	**Timestamp Target Time Error** This bit is set when the latest target time programmed in the MAC_PPS2_Target_Time_Seconds and MAC_PPS2_Target_Time_Nanoseconds registers elapses. This bit is cleared when the application reads this bit. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
TSTARGT3	8	r	**Timestamp Target Time Reached for Target Time PPS3** When this bit is set, it indicates that the value of system time is greater than or equal to the value specified in the MAC_PPS3_Target_Time_Seconds and MAC_PPS3_Target_Time_Nanoseconds registers. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
TSTRGTERR3	9	r	**Timestamp Target Time Error** This bit is set when the latest target time programmed in the MAC_PPS3_Target_Time_Seconds and MAC_PPS3_Target_Time_Nanoseconds registers elapses. This bit is cleared when the application reads this bit. Access restriction applies. Clears on read (or this bit is written to 1 when RCWE bit in MAC_CSR_SW_Ctrl register is set). Self-set to 1 on internal event. Value After Reset: 0x0
RES_14_10	14:10	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
TXTSSIS	15	r	**Tx Timestamp Status Interrupt Status** When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the MAC_Tx_Timestamp_Status_Nanoseconds and MAC_Tx_Timestamp_Status_Seconds registers. When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the MAC_Tx_Timestamp_Status_Nanoseconds and MAC_Tx_Timestamp_Status_Seconds registers, for PTO generated Delay Request and Pdelay request packets. This bit is cleared when the MAC_Tx_Timestamp_Status_Seconds register is read (or write to MAC_Tx_Timestamp_Status_Seconds register when RCWE bit of MAC_CSR_SW_Ctrl register is set). Value After Reset: 0x0
RES_19_16	19:16	r	**Reserved for future use.**
RES_23_20	23:20	r	**Reserved** Value After Reset: 0x0
RES_29_24	29:24	r	**Reserved for future use.**
RES_31_30	31:30	r	**Reserved** Value After Reset: 0x0

MAC Transmit Timestamp Nanoseconds Status Register

This register contains the nanosecond part of timestamp captured for Transmit packets when Tx status is disabled.

The MAC_Tx_Timestamp_Status_Nanoseconds register, along with MAC_TxTimestamp_Status_Seconds, gives the 64-bit timestamp captured for the PTP packet successfully transmitted by the MAC. This value is considered to be read by the application when the last byte of MAC_Tx_Timestamp_Status_Nanoseconds is read. In the little-endian mode, this means when bits[31:24] are read; in big-endian mode, bits[7:0] are read.

If the application does not read these registers and timestamp of another packet is captured, then either the current timestamp is lost (overwritten) or the new timestamp is lost (dropped), depending on the setting of the TXTSSTSM bit of the MAC_Timestamp_Control register. The status bit TXTSC bit [15] in MAC_Timestamp_Status register is set whenever the MAC transmitter captures the timestamp.

MAC_TX_TIMESTAMP_STATUS_NANOSECONDS

MAC Transmit Timestamp Nanoseconds Status Register($0B30_H$) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSMIS	**TXTSSLO**														
r	r														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSLO															
r															

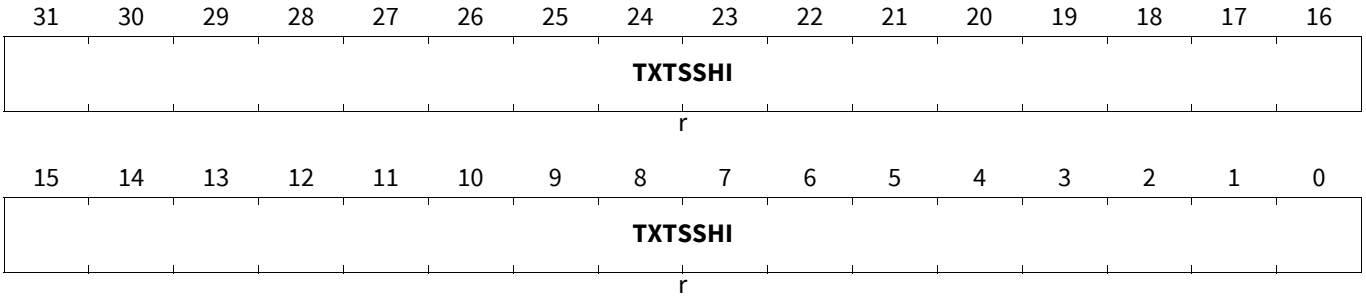
Field	Bits	Type	Description
TXTSSLO	30:0	r	**Transmit Timestamp Status Low** This field contains the 31 bits of the Nanoseconds field of the Transmit packet's captured timestamp. Value After Reset: 0x0
TXTSSMIS	31	r	**Transmit Timestamp Status Missed** When this bit is set, it indicates one of the following: • The timestamp of the current packet is ignored if TXTSSTSM bit of the MAC_Timestamp_Control register is reset • The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the MAC_Timestamp_Control register is set. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

MAC Transmit Timestamp Seconds Status Register

The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

MAC_TX_TIMESTAMP_STATUS_SECONDS

MAC Transmit Timestamp Seconds Status Register($0B34_H$) **Application Reset Value: $0000\ 0000_H$**



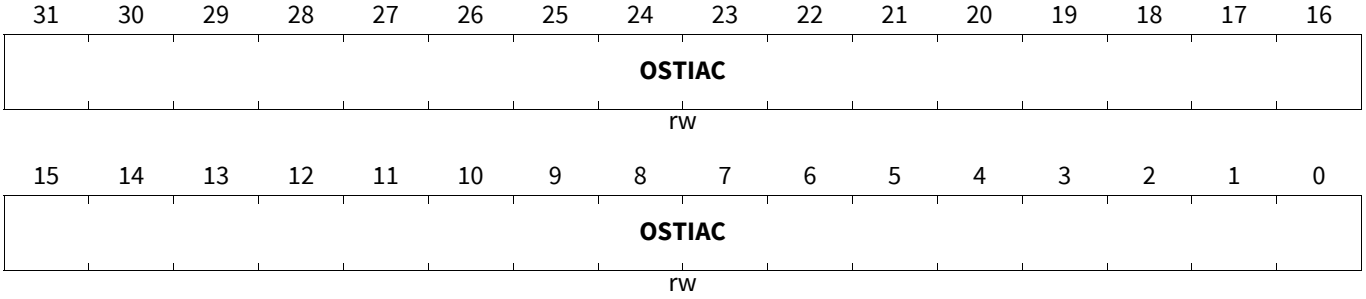
Field	Bits	Type	Description
TXTSSHI	31:0	r	**Transmit Timestamp Status High** This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp. Value After Reset: 0x0

MAC Timestamp Ingress Asymmetry Correction Register

The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

MAC_TIMESTAMP_INGRESS_ASYM_CORR

MAC Timestamp Ingress Asymmetry Correction Register(0B50$_H$) **Application Reset Value: 0000 0000$_H$**



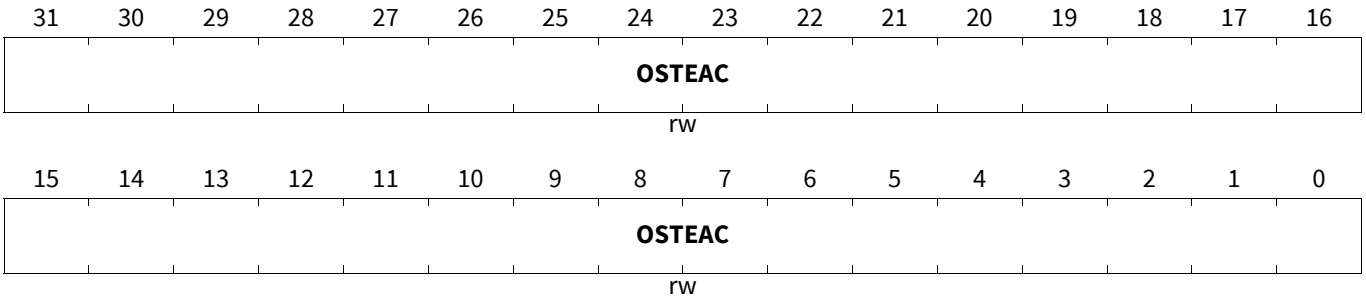
Field	Bits	Type	Description
OSTIAC	31:0	rw	**One-Step Timestamp Ingress Asymmetry Correction** This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^16. For example, 2.5 ns is represented as 0x00028000. The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit. Value After Reset: 0x0

MAC Timestamp Egress Asymmetry Correction Register

The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

MAC_TIMESTAMP_EGRESS_ASYM_CORR

MAC Timestamp Egress Asymmetry Correction Register(0B54$_H$) **Application Reset Value: 0000 0000$_H$**



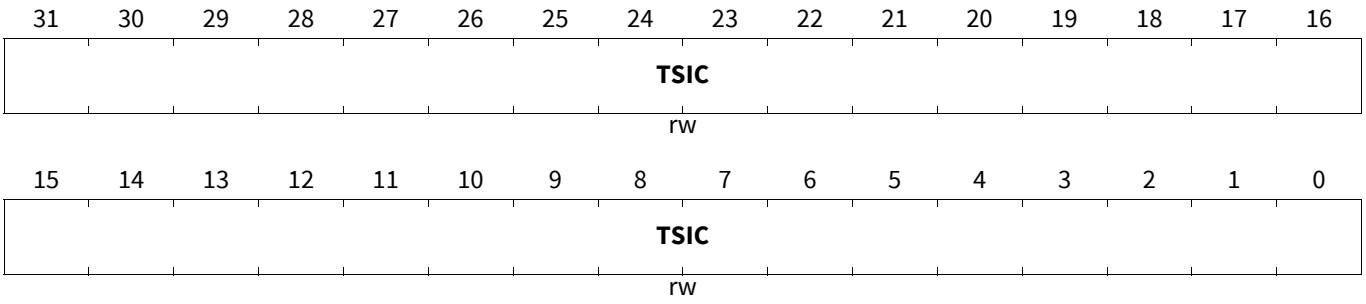
Field	Bits	Type	Description
OSTEAC	31:0	rw	**One-Step Timestamp Egress Asymmetry Correction** This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^16. For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFD_8000, which is the 2's complement of 0x0002_8000(2.5 * 216). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003_4CCC (3.3 * 216). Value After Reset: 0x0

MAC Timestamp Ingress Correction Nanoseconds Register

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

MAC_TIMESTAMP_INGRESS_CORR_NANOSECOND

MAC Timestamp Ingress Correction Nanoseconds Register($0B58_H$) **Application Reset Value: $0000\ 0000_H$**



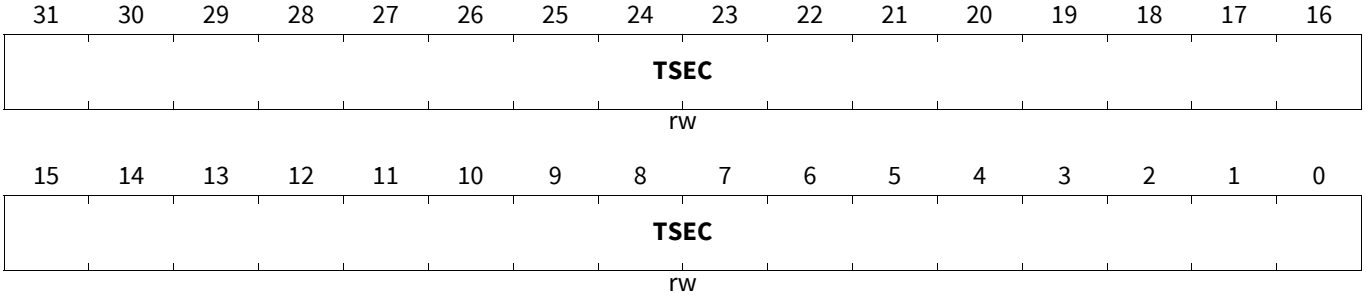
Field	Bits	Type	Description
TSIC	31:0	rw	**Timestamp Ingress Correction** This field contains the ingress path correction value as defined by the Ingress Correction expression. Value After Reset: 0x0

MAC Timestamp Egress Correction Nanoseconds Register

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

MAC_TIMESTAMP_EGRESS_CORR_NANOSECOND

MAC Timestamp Egress Correction Nanoseconds Register(0B5C$_H$) Application Reset Value: 0000 0000$_H$



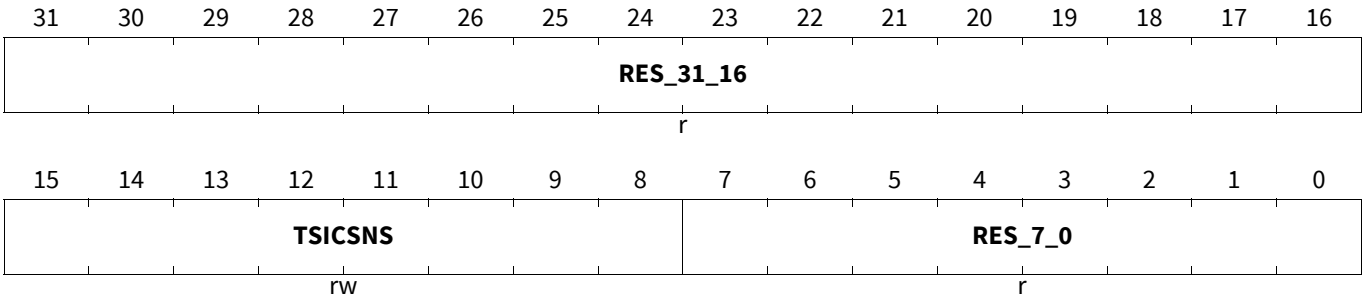
Field	Bits	Type	Description
TSEC	31:0	rw	**Timestamp Egress Correction** This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression. Value After Reset: 0x0

MAC Timestamp Ingress Correction Subnanoseconds Register

This register contains the sub-nanosecond part of the correction value to be used with the captured timestamp value, for ingress direction.

MAC_TIMESTAMP_INGRESS_CORR_SUBNANOSEC

MAC Timestamp Ingress Correction Subnanoseconds Register(0B60$_H$) Application Reset Value: 0000 0000$_H$



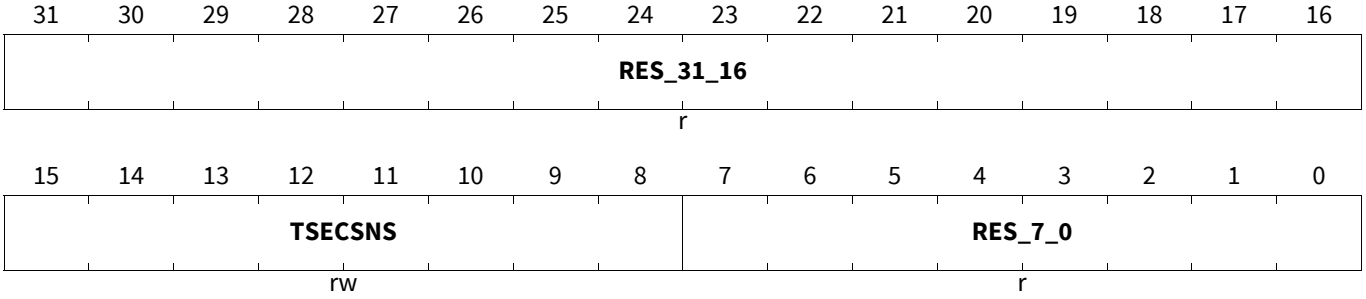
Field	Bits	Type	Description
RES_7_0	7:0	r	**Reserved** Value After Reset: 0x0
TSICSNS	15:8	rw	**Timestamp Ingress Correction, sub-nanoseconds** This field contains the sub-nanoseconds part of the ingress path correction value as defined by the “Ingress Correction” expression. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

MAC Timestamp Egress Correction Subnanoseconds Register

This register contains the sub-nanosecond part of the correction value to be used with the captured timestamp value, for egress direction.

MAC_TIMESTAMP_EGRESS_CORR_SUBNANOSEC

MAC Timestamp Egress Correction Subnanoseconds Register($0B64_H$) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
RES_7_0	7:0	r	**Reserved** Value After Reset: 0x0
TSECSNS	15:8	rw	**Timestamp Egress Correction, sub-nanoseconds** This field contains the sub-nanoseconds part of the egress path correction value as defined by the “Egress Correction” expression. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

MAC PPS Control Register

PPS Control register.

Bits[30:24] of this register are valid only when four Flexible PPS outputs are selected. Bits[22:16] are valid only when three or more Flexible PPS outputs are selected. Bits[14:8] are valid only when two or more Flexible PPS outputs are selected. Bits[6:4] are valid only when Flexible PPS feature is selected.

Notes on Bits [3:0] in the binary rollover mode, PPSEN0 = 0

The PPS output (ptp_pps_o) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:

When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.

When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of one clock of 50 percent duty cycle and 537 ms period and a second clock of 463 ms period (268 ms low and 195 ms high)

When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of three clocks of 50 percent duty cycle and 268 ms period and a fourth clock of 195 ms period (134 ms low and 61 ms high)

This behavior is because of the non-linear toggling of bits in the digital rollover mode in the MAC_System_Time_Nanoseconds register.

Notes on Bits [3:0]: In the mode “Flexible PPS Output (ptp_pps_o[0]) Control”, PPSEN0 = 1

Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are 'all-zero'. The following list describes the values of PPSCMD0:

0000: No Command

0001: START Single Pulse

This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS0 Width Register.

0010: START Pulse Train

This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS0 Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands.

0011: Cancel START

This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time.

0100: STOP Pulse train at time

This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010) after the time programmed in the Target Time registers elapses.

0101: STOP Pulse Train immediately

This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010).

0110: Cancel STOP Pulse train

This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.

0111-1111: Reserved

MAC_PPS_CONTROL
MAC PPS Control Register **($0B70_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30:29	28:27	26:25:24	23	22:21	20:19	18:17:16
RES_31	RES_30_29	RES_28_27	RES_26_24	RES_23	RES_22_21	RES_20_19	RES_18_16
r	r	r	r	r	r	r	r

15	14:13	12:11	10:9:8	7	6:5	4	3:2:1:0
RES_15	RES_14_13	RES_12_11	RES_10_8	RES_7	TRGTMODSEL0	PPSEN0	PPSCTRL_PPSCMD
r	r	r	r	r	rw	rw	rw

Field	Bits	Type	Description
PPSCTRL_PPSCMD	3:0	rw	**Flexible PPS Output (ptp_pps_o[0]) Control** This field controls the frequency of the PPS0 output (ptp_pps_o[0]) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies: LOST SEQUENCE DEFINITION Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0 1_H The binary rollover is 2 Hz, and the digital rollover is 1 Hz. 2_H The binary rollover is 4 Hz, and the digital rollover is 2 Hz. 3_H The binary rollover is 8 Hz, and the digital rollover is 4 Hz. 4_H The binary rollover is 16 Hz, and the digital rollover is 8 Hz. F_H The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz.
PPSEN0	4	rw	**Flexible PPS Output Mode Enable** When this bit is set, Bits[3:0] function as PPSCMD. When this bit is reset, Bits[3:0] function as PPSCTRL (Fixed PPS mode). Value After Reset: 0x0
TRGTMODSEL0	6:5	rw	**Target Time Register Mode for PPS0 Output** This field indicates the Target Time registers (MAC_PPS0_Target_Time_Seconds and MAC_PPS0_Target_Time_Nanoseconds) mode for PPS0 output signal: Value After Reset: 0x0 00_B Target Time registers are programmed only for generating the interrupt event. The Flexible PPS function must not be enabled in this mode, otherwise spurious transitions may be observed on the corresponding ptp_pps_o output port. 01_B Reserved 10_B Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS0 output signal generation. 11_B Target Time registers are programmed only for starting or stopping the PPS0 output signal generation. No interrupt is asserted.
RES_7	7	r	**Reserved** Value After Reset: 0x0
RES_10_8	10:8	r	**Reserved for future use.**
RES_12_11	12:11	r	**Reserved** Value After Reset: 0x0
RES_14_13	14:13	r	**Reserved for future use.**
RES_15	15	r	**Reserved** Value After Reset: 0x0
RES_18_16	18:16	r	**Reserved for future use.**
RES_20_19	20:19	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
RES_22_21	22:21	r	**Reserved for future use.**
RES_23	23	r	**Reserved** Value After Reset: 0x0
RES_26_24	26:24	r	**Reserved for future use.**
RES_28_27	28:27	r	**Reserved** Value After Reset: 0x0
RES_30_29	30:29	r	**Reserved for future use.**
RES_31	31	r	**Reserved** Value After Reset: 0x0

MAC PPS 0 Target Time Seconds Register

The PPS Target Time Seconds register, along with PPS Target Time Nanoseconds register, is used to schedule an interrupt event [Bit 1 of MAC_Timestamp_Status] when the system time exceeds the value programmed in these registers.

MAC_PPS0_TARGET_TIME_SECONDS
MAC PPS 0 Target Time Seconds Register (0B80$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSTRH0															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSTRH0															
rw															

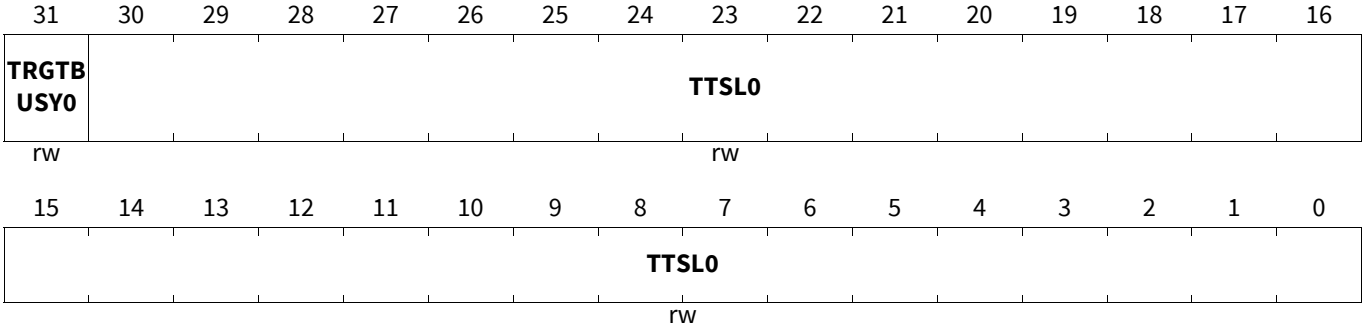
Field	Bits	Type	Description
TSTRH0	31:0	rw	**PPS Target Time Seconds Register** This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the MAC_PPS_Control register. Value After Reset: 0x0

MAC PPS 0 Target Time Nanoeconds Register

PPS0 Target Time Nanoseconds register.

MAC_PPS0_TARGET_TIME_NANOSECONDS

MAC PPS 0 Target Time Nanoeconds Register (0B84$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TTSL0	30:0	rw	**Target Time Low for PPS Register** This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in MAC_PPS_Control. When the TSCTRLSSR bit is reset in the MAC_Timestamp_Control register, this value should be (time in ns / 0.465). The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value. When the TSCTRLSSR bit is set in the MAC_Timestamp_Control register, this value should not exceed 0x3B9A_C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
TRGTBUSY0	31	rw	**PPS Target Time Register Busy** The MAC sets this bit when the PPSCMD0 field in the MAC_PPS_Control register is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers to the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted. Value After Reset: 0x0

MAC PPS 0 Interval Register

The PPS0 Interval register contains the number of units of sub-second increment value between the rising edges of PPS0 signal output (ptp_pps_o[0]).

MAC_PPS0_INTERVAL

MAC PPS 0 Interval Register **(0B88$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSINT0															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSINT0															
rw															

Field	Bits	Type	Description
PPSINT0	31:0	rw	**PPS Output Signal Interval** These bits store the interval between the rising edges of PPS0 signal output. The interval is stored in terms of number of units of sub-second increment value. You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS0 signal output is 100 ns (that is, 5 units of sub-second increment value), you should program value 4 (5-1) in this register. Value After Reset: 0x0

MAC PPS 0 Width Register

The PPS0 Width register contains the number of units of sub-second increment value between the rising and corresponding falling edges of PPS0 signal output (ptp_pps_o[0]).

MAC_PPS0_WIDTH

MAC PPS 0 Width Register **(0B8C$_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSWIDTH0															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSWIDTH0															
rw															

Field	Bits	Type	Description
PPSWIDTH0	31:0	rw	**PPS Output Signal Width** These bits store the width between the rising edge and corresponding falling edge of PPS0 signal output. The width is stored in terms of number of units of sub-second increment value. You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS0 signal output is 80 ns (that is, four units of sub-second increment value), you should program value 3 (4-1) in this register. **Note**: The value programmed in this register must be lesser than the value programmed in MAC_PPS0_Interval. Value After Reset: 0x0

MTL Operation Mode Register

The Operation Mode register establishes the Transmit and Receive operating modes and commands.

MTL_OPERATION_MODE
MTL Operation Mode Register **($0C00_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_10															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_31_10						CNTCLR	CNTPRST	RES_7	SCHALG		RES_4_3		RAA	DTXSTS	RES_0
r						rw	rw	r	rw		r		rw	rw	r

Field	Bits	Type	Description
RES_0	0	r	**Reserved** Value After Reset: 0x0
DTXSTS	1	rw	**Drop Transmit Status** When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application. Value After Reset: 0x0
RAA	2	rw	**Receive Arbitration Algorithm** This field is used to select the arbitration algorithm for the Rx side. Queue 0 has the lowest priority and the last queue has the highest priority. Value After Reset: 0x0 0_B Strict priority (SP) 1_B Weighted Strict Priority (WSP)
RES_4_3	4:3	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
SCHALG	6:5	rw	**Tx Scheduling Algorithm** This field indicates the algorithm for Tx scheduling: Value After Reset: 0x0 00_B WRR algorithm 01_B Reserved 10_B Reserved. 11_B Strict priority algorithm.
RES_7	7	r	**Reserved** Value After Reset: 0x0
CNTPRST	8	rw	**Counters Preset** When this bit is set, • MTL_TxQ[0-7]_Underflow register is initialized/preset to 12'h7F0. • Missed Packet and Overflow Packet counters in MTL_RxQ[0-7]_Missed_Packet_Overflow_Cnt register is initialized/preset to 12'h7F0. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
CNTCLR	9	rw	**Counters Reset** When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNTPRESET bit, CNTPRESET has precedence. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
RES_31_10	31:10	r	**Reserved** Value After Reset: 0x0

MTL Interrupt Status Register

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

MTL_INTERRUPT_STATUS

MTL Interrupt Status Register **($0C20_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_18														RES_17_16	
r														r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_8								RES_7_4				Q3IS	Q2IS	Q1IS	Q0IS
r								r				r	r	r	r

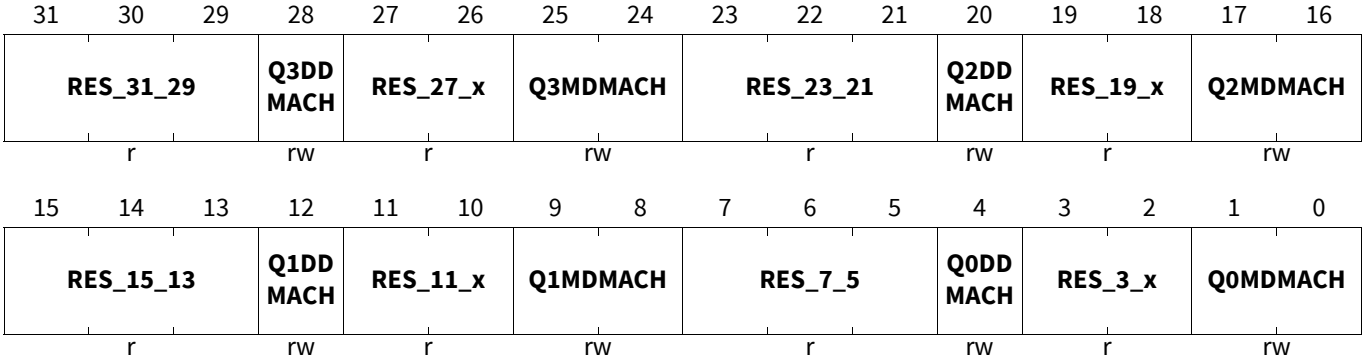
Field	Bits	Type	Description
Q0IS	0	r	**Queue 0 Interrupt status** This bit indicates that there is an interrupt from Queue 0. To reset this bit, the application must read Queue 0 Interrupt Control and Status register to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0
Q1IS	1	r	**Queue 1 Interrupt status** This bit indicates that there is an interrupt from Queue 1. To reset this bit, the application must read the MTL_Q1_Interrupt_Control_Status register to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0
Q2IS	2	r	**Queue 2 Interrupt status** This bit indicates that there is an interrupt from Queue 2. To reset this bit, the application must read the MTL_Q2_Interrupt_Control_Status register to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0
Q3IS	3	r	**Queue 3 Interrupt status** This bit indicates that there is an interrupt from Queue 3. To reset this bit, the application must read the MTL_Q3_Interrupt_Control_Status register to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0
RES_7_4	7:4	r	**Reserved for future use.**
RES_15_8	15:8	r	**Reserved** Value After Reset: 0x0
RES_17_16	17:16	r	**Reserved for future use.**
RES_31_18	31:18	r	**Reserved** Value After Reset: 0x0

MTL Receive Queue and DMA Channel Mapping 0 Register

The Receive Queue and DMA Channel Mapping 0 register is reserved in EQOS-CORE and EQOS-MTL configurations.

MTL_RXQ_DMA_MAP0

MTL Receive Queue and DMA Channel Mapping 0 Register($0C30_H$) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
Q0MDMACH	1:0	rw	**Queue 0 Mapped to DMA Channel** This field controls the routing of the packet received in Queue 0 to the DMA channel: This field is valid when the Q0DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values may be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved. Value After Reset: 0x0 00_B DMA Channel 0 01_B DMA Channel 1 10_B DMA Channel 2 11_B DMA Channel 3
RES_3_x	3:2	r	**Reserved** Value After Reset: 0x0
Q0DDMACH	4	rw	**Queue 0 Enabled for DA-based DMA Channel Selection** When set, this bit indicates that the packets received in Queue 0 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 0 are routed to the DMA Channel programmed in the Q0MDMACH field. Value After Reset: 0x0
RES_7_5	7:5	r	**Reserved** Value After Reset: 0x0
Q1MDMACH	9:8	rw	**Queue 1 Mapped to DMA Channel** This field controls the routing of the received packet in Queue 1 to the DMA channel: This field is valid when the Q1DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values may be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the other bits are reserved. Value After Reset: 0x0 00_B DMA Channel 0 01_B DMA Channel 1 10_B DMA Channel 2 11_B DMA Channel 3
RES_11_x	11:10	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
Q1DDMACH	12	rw	**Queue 1 Enabled for DA-based DMA Channel Selection** When set, this bit indicates that the packets received in Queue 1 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 1 are routed to the DMA Channel programmed in the Q1MDMACH field (Bits[10:8]). Value After Reset: 0x0
RES_15_13	15:13	r	**Reserved** Value After Reset: 0x0
Q2MDMACH	17:16	rw	**Queue 2 Mapped to DMA Channel** This field controls the routing of the received packet in Queue 2 to the DMA channel: This field is valid when the Q2DDMACH field is reset. Value After Reset: 0x0 00_B DMA Channel 0 01_B DMA Channel 1 10_B DMA Channel 2 11_B DMA Channel 3
RES_19_x	19:18	r	**Reserved** Value After Reset: 0x0
Q2DDMACH	20	rw	**Queue 2 Enabled for DA-based DMA Channel Selection** When set, this bit indicates that the packets received in Queue 2 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. **When reset, this bit indicates that the packets received in Queue 2 are routed to the DMA Channel programmed in the Q2MDMACH field (Bits[18:16]).** Value After Reset: 0x0
RES_23_21	23:21	r	**Reserved** Value After Reset: 0x0
Q3MDMACH	25:24	rw	**Queue 3 Mapped to DMA Channel** This field controls the routing of the received packet in Queue 3 to the DMA channel: This field is valid when the Q3DDMACH field is reset. The width of this field depends on the number of RX DMA channels and not all the values may be valid in some configurations. For example, if the number of RX DMA channels selected is 2, only 000 and 001 are valid, the others are reserved Value After Reset: 0x0 00_B DMA Channel 0 01_B DMA Channel 1 10_B DMA Channel 2 11_B DMA Channel 3

Field	Bits	Type	Description
RES_27_x	27:26	r	**Reserved** Value After Reset: 0x0
Q3DDMACH	28	rw	**Queue 3 Enabled for Dynamic (per packet) DMA Channel Selection** When set, this bit indicates that the packets received in Queue 3 are routed to a particular DMA channel as decided in the MAC Receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address. When reset, this bit indicates that the packets received in Queue 3 are routed to the DMA Channel programmed in the Q3MDMACH field (Bits[26:24]). Value After Reset: 0x0
RES_31_29	31:29	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Transmit Operation Mode Register

The Queue 0 Transmit Operation Mode register establishes the Transmit queue operating modes and commands.

MTL_TXQ0_OPERATION_MODE

MTL Queue 0 Transmit Operation Mode Register($0D00_H$) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_x												TQS			
r												rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_7									TTC			TXQEN		TSF	FTQ
r									rw			rw		rw	rw

Field	Bits	Type	Description
FTQ	0	rw	**Flush Transmit Queue** When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the MTL_TxQ1_Operation_Mode register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. **Note**: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0

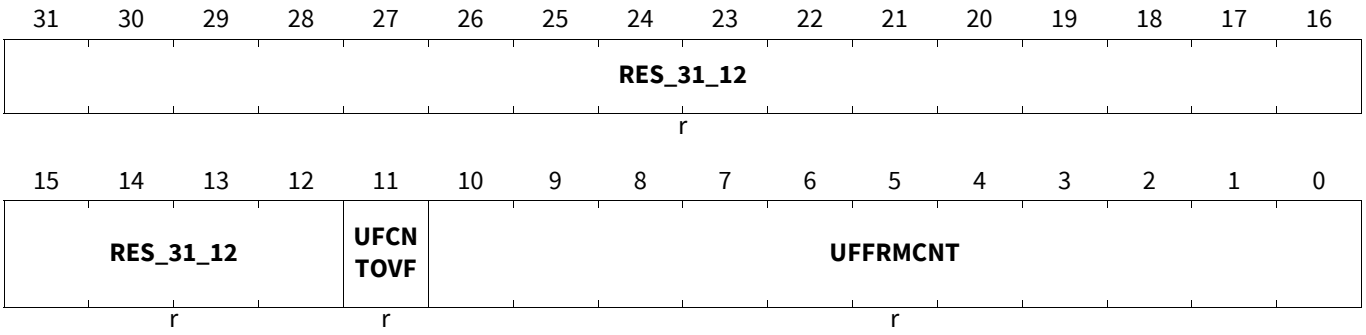
Field	Bits	Type	Description
TSF	1	rw	**Transmit Store and Forward** When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped. Value After Reset: 0x0
TXQEN	3:2	rw	**Transmit Queue Enable** This field is used to enable/disable the transmit queue 0. This field is Read Only in Single Queue configurations and Read Write in Multiple Queue configurations. **Note**: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field. Value After Reset: 0x0 00_B Not enabled 01_B Reserved 10_B Enabled 11_B Reserved
TTC	6:4	rw	**Transmit Threshold Control** These bits control the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset. Value After Reset: 0x0 000_B 32 001_B 64 010_B 96 011_B 128 100_B 192 101_B 256 110_B 384 111_B 512
RES_15_7	15:7	r	**Reserved** Value After Reset: 0x0
TQS	19:16	rw	**Transmit Queue Size** This field indicates the size of the allocated Transmit queues in blocks of 256 bytes. The TQS field is read-write only if the number of Tx Queues more than one, the reset value is 0x0 and indicates size of 256 bytes. When the number of Tx Queues is one, the field is read-only and the configured TX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Tx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(2048/256) = LOG2(8) = 3 bits Value After Reset: 0x0
RES_31_x	31:20	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Transmit Underflow Counter Register

The Queue 0 Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush

MTL_TXQ0_UNDERFLOW

MTL Queue 0 Transmit Underflow Counter Register(0D04$_H$) **Application Reset Value: 0000 0000$_H$**



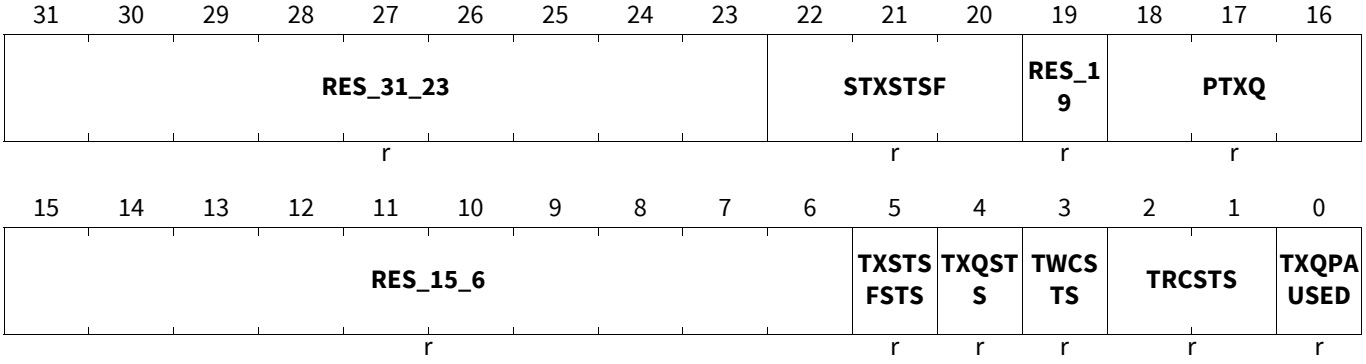
Field	Bits	Type	Description
UFFRMCNT	10:0	r	**Underflow Packet Counter** This field indicates the number of packets aborted by the controller because of Tx Queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
UFCNTOVF	11	r	**Overflow Bit for Underflow Packet Counter** This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_31_12	31:12	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Transmit Debug Register

The Queue 0 Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

MTL_TXQ0_DEBUG

MTL Queue 0 Transmit Debug Register (0D08$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
TXQPAUSED	0	r	**Transmit Queue in Pause** When this bit is high and the Rx flow control is enabled, it indicates that the Tx Queue is in the Pause condition (in the full-duplex only mode) because of the following: • Reception of the PFC packet for the priorities assigned to the Tx Queue when PFC is enabled • Reception of 802.3x Pause packet when PFC is disabled Value After Reset: 0x0
TRCSTS	2:1	r	**MTL Tx Queue Read Controller Status** This field indicates the state of the Tx Queue Read Controller: Value After Reset: 0x0 00_B Idle state 01_B Read state (transferring data to the MAC transmitter) 10_B Waiting for pending Tx Status from the MAC transmitter 11_B Flushing the Tx queue because of the Packet Abort request from the MAC
TWCSTS	3	r	**MTL Tx Queue Write Controller Status** When high, this bit indicates that the MTL Tx Queue Write Controller is active, and it is transferring the data to the Tx Queue. Value After Reset: 0x0
TXQSTS	4	r	**MTL Tx Queue Not Empty Status** When this bit is high, it indicates that the MTL Tx Queue is not empty and some data is left for transmission. Value After Reset: 0x0
TXSTSFSTS	5	r	**MTL Tx Status FIFO Full Status** When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission. Value After Reset: 0x0
RES_15_6	15:6	r	**Reserved** Value After Reset: 0x0

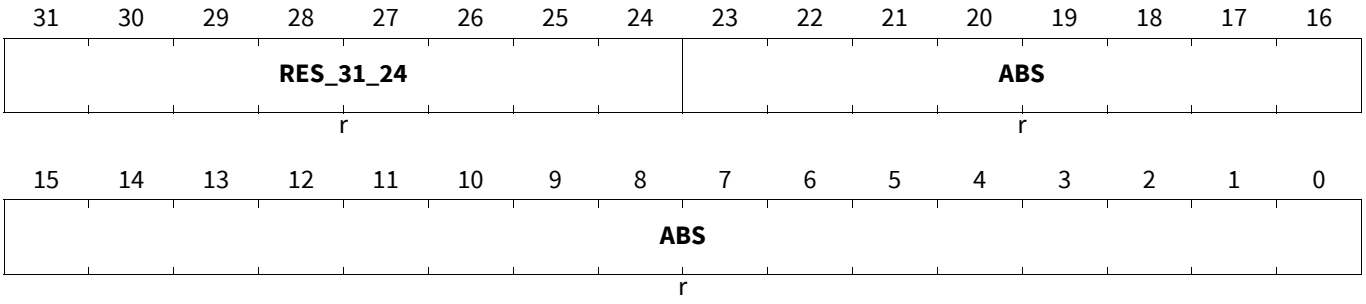
Field	Bits	Type	Description
PTXQ	18:16	r	**Number of Packets in the Transmit Queue** This field indicates the current number of packets in the Tx Queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of packets in the Transmit queue. Value After Reset: 0x0
RES_19	19	r	**Reserved** Value After Reset: 0x0
STXSTSF	22:20	r	**Number of Status Words in Tx Status FIFO of Queue** This field indicates the current number of status in the Tx Status FIFO of this queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of status words in Tx Status FIFO. Value After Reset: 0x0
RES_31_23	31:23	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Transmit Status Register

The Queue 0 ETS Status register provides the average traffic transmitted in Queue 0.

MTL_TXQ0_ETS_STATUS

MTL Queue 0 Transmit Status Register (0D14$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
ABS	23:0	r	**Average Bits per Slot This field contains the average transmitted bits per slot.** When the DCB operation is enabled for Queue 0, this field is computed over every 10 million bit times slot (10 ms in 1000 Mbps; 100 ms in 100 Mbps). The maximum value is 0x989680. Value After Reset: 0x0
RES_31_24	31:24	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Transmit Quantum or Weights Register

The Queue 0 Quantum or Weights register contains the quantum value for Deficit Weighted Round Robin (DWRR), weights for the Weighted Round Robin (WRR), and Weighted Fair Queuing (WFQ) for Queue 0.

MTL_TXQ0_QUANTUM_WEIGHT

MTL Queue 0 Transmit Quantum or Weights Register(0D18$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_21											ISCQW				
r											rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISCQW															
rw															

Field	Bits	Type	Description
ISCQW	20:0	rw	**Quantum or Weights** When the DCB operation is enabled with DWRR algorithm for Queue 0 traffic, this field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. When DCB operation is enabled with WFQ algorithm for Queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits[20:14] must be written to zero. When DCB operation or generic queueing operation is enabled with WRR algorithm for Queue 0 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [20:7] must be written to zero. Value After Reset: 0x0
RES_31_21	31:21	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Interrupt Control Status Register

This register contains the interrupt enable and status bits for the queue 0 interrupts.

MTL_Q0_INTERRUPT_CONTROL_STATUS

MTL Queue 0 Interrupt Control Status Register (0D2C$_H$) **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_25							RXOIE	RES_23_17							RXOVFIS
r							rw	r							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_10						ABPSIE	TXUIE	RES_7_2						ABPSIS	TXUNFIS
r						rw	rw	r						rw	rw

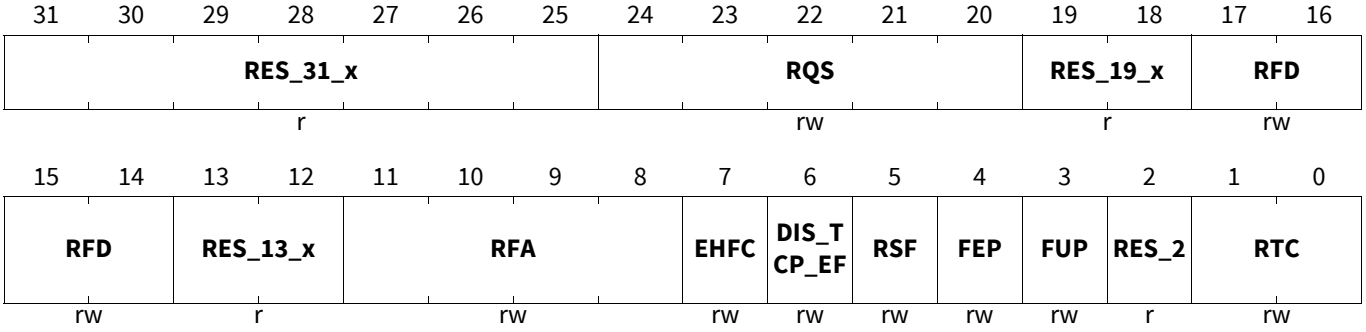
Field	Bits	Type	Description
TXUNFIS	0	rw	**Transmit Queue Underflow Interrupt Status** This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
ABPSIS	1	rw	**Average Bits Per Slot Interrupt Status** When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
RES_7_2	7:2	r	**Reserved** Value After Reset: 0x0
TXUIE	8	rw	**Transmit Queue Underflow Interrupt Enable** When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled. Value After Reset: 0x0
ABPSIE	9	rw	**Average Bits Per Slot Interrupt Enable** When this bit is set, the MAC asserts the sbd_intr_o or mci_intr_o interrupt when the average bits per slot status is updated. **When this bit is cleared, the interrupt is not asserted for such an event.** Value After Reset: 0x0
RES_15_10	15:10	r	**Reserved** Value After Reset: 0x0
RXOVFIS	16	rw	**Receive Queue Overflow Interrupt Status** This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
RES_23_17	23:17	r	**Reserved** Value After Reset: 0x0
RXOIE	24	rw	**Receive Queue Overflow Interrupt Enable** When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled. Value After Reset: 0x0
RES_31_25	31:25	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Receive Operation Mode Register

The Queue 0 Receive Operation Mode register establishes the Receive queue operating modes and command. The RFA and RFD fields are not backward compatible with the RFA and RFD fields of 4.00a release

MTL_RXQ0_OPERATION_MODE

MTL Queue 0 Receive Operation Mode Register (0D30$_H$) **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
RTC	1:0	rw	**Receive Queue Threshold Control** These bits control the threshold level of the MTL Rx queue (in bytes): The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1. Value After Reset: 0x0 00_B 64 01_B 32 10_B 96 11_B 128
RES_2	2	r	**Reserved** Value After Reset: 0x0
FUP	3	rw	**Forward Undersized Good Packets** When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01. Value After Reset: 0x0

Field	Bits	Type	Description
FEP	4	rw	**Forward Error Packets** When this bit is reset, the Rx queue drops packets with error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA. Value After Reset: 0x0
RSF	5	rw	**Receive Queue Store and Forward** When this bit is set, the DWC_ether_qos reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register. Value After Reset: 0x0
DIS_TCP_EF	6	rw	**Disable Dropping of TCP/IP Checksum Error Packets** When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset. Value After Reset: 0x0
EHFC	7	rw	**Enable Hardware Flow Control** When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled. Value After Reset: 0x0
RFA	11:8	rw	**Threshold for Activating Flow Control (in half-duplex and full-duplex** These bits control the threshold (fill-level of Rx queue) at which the flow control is activated: For more information on encoding for this field, see RFD. Value After Reset: 0x0
RES_13_x	13:12	r	**Reserved** Value After Reset: 0x0

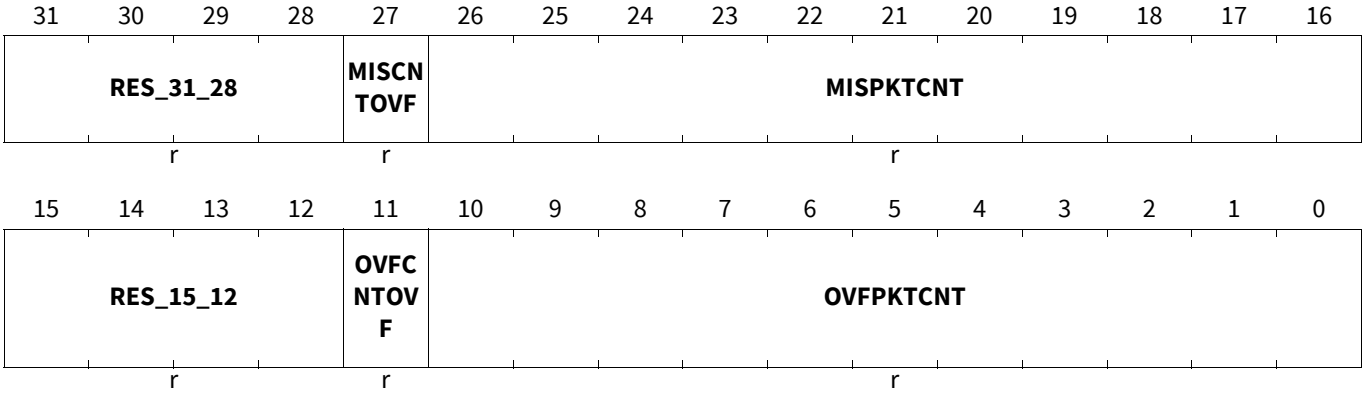
Field	Bits	Type	Description
RFD	17:14	rw	**Threshold for Deactivating Flow Control (in half-duplex and full-duplex modes)** These bits control the threshold (fill-level of Rx queue) at which the flow control is de-asserted after activation: LOST SEQUENCE DEFINITION The de-assertion is effective only after flow control is asserted. **Note:** The value must be programmed in such a way to make sure that the threshold is a positive number. When the EHFC is set high, these values are applicable only when the Rx queue size determined by the RQS field of this register, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size determined by the RQS field of this register. The width of this field depends on RX FIFO size selected during the configuration. Remaining bits are reserved and read only. Value After Reset: 0x0 0_H Full minus 1 KB, that is, FULL 1 KB 1_H Full minus 1.5 KB, that is, FULL 1.5 KB 2_H Full minus 2 KB, that is, FULL 2 KB 3_H Full minus 2.5 KB, that is, FULL 2.5 KB E_H Full minus 8 KB, that is, FULL 8 KB F_H Full minus 8.5 KB, that is, FULL 8.5 KB
RES_19_x	19:18	r	**Reserved** Value After Reset: 0x0
RQS	24:20	rw	**Receive Queue Size** This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The RQS field is read-write only if the number of Rx Queues more than one, the reset value is 0x0 and indicates size of 256 bytes. When the number of Rx Queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Rx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(2048/256) = LOG2(8) = 3 bits Value After Reset: 0x0
RES_31_x	31:25	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Receive Missed Packet and Overflow Counter Register

The Queue 0 Missed Packet and Overflow Counter register contains the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

MTL_RXQ0_MISSED_PACKET_OVERFLOW_CNT

MTL Queue 0 Receive Missed Packet and Overflow Counter Register($0D34_H$) Application Reset Value: $0000\ 0000_H$

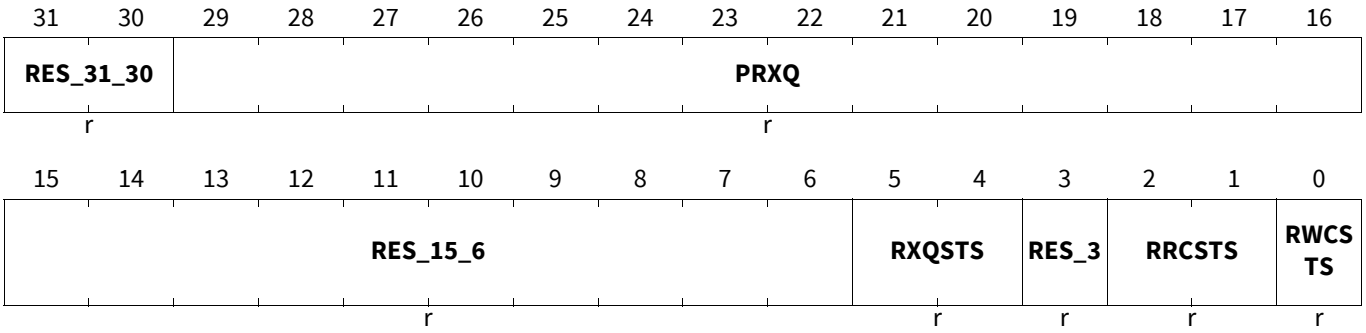


Field	Bits	Type	Description
OVFPKTCNT	10:0	r	**Overflow Packet Counter** This field indicates the number of packets discarded by the DWC_ether_qos because of Receive queue overflow. This counter is incremented each time the DWC_ether_qos discards an incoming packet because of overflow. This counter is reset when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
OVFCNTOVF	11	r	**Overflow Counter Overflow Bit** When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_15_12	15:12	r	**Reserved** Value After Reset: 0x0
MISPKTCNT	26:16	r	**Missed Packet Counter** This field indicates the number of packets missed by the DWC_ether_qos because the application asserted ari_pkt_flush_i[] for this queue. This counter is reset when this register is read with mci_be_i[0] at 1b1. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
MISCNTOVF	27	r	**Missed Packet Counter Overflow Bit** When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_31_28	31:28	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Receive Debug Register

The Queue 0 Receive Debug register gives the debug status of various blocks related to the Receive queue.

MTL_RXQ0_DEBUG
MTL Queue 0 Receive Debug Register (0D38$_H$) Application Reset Value: 0000 0000$_H$



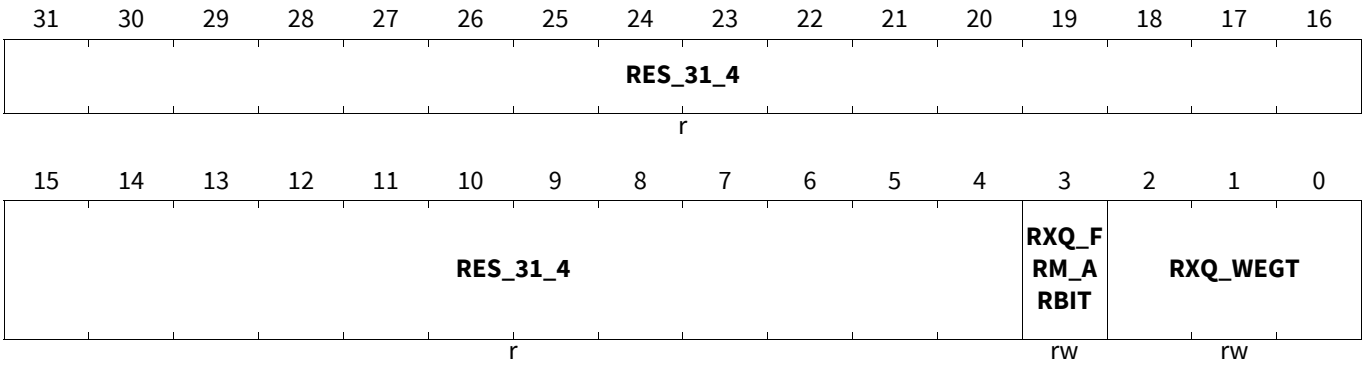
Field	Bits	Type	Description
RWCSTS	0	r	**MTL Rx Queue Write Controller Active Status** When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx Queue. Value After Reset: 0x0
RRCSTS	2:1	r	**MTL Rx Queue Read Controller State** This field gives the state of the Rx queue Read controller: Value After Reset: 0x0 00_B Idle state 01_B Reading packet data 10_B Reading packet status (or timestamp) 11_B Flushing the packet data and status
RES_3	3	r	**Reserved** Value After Reset: 0x0
RXQSTS	5:4	r	**MTL Rx Queue Fill-Level Status** This field gives the status of the fill-level of the Rx Queue: Value After Reset: 0x0 00_B Rx Queue empty 01_B Rx Queue fill-level below flow-control deactivate threshold 10_B Rx Queue fill-level above flow-control activate threshold 11_B Rx Queue full
RES_15_6	15:6	r	**Reserved** Value After Reset: 0x0
PRXQ	29:16	r	**Number of Packets in Receive Queue** This field indicates the current number of packets in the Rx Queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size. Value After Reset: 0x0
RES_31_30	31:30	r	**Reserved** Value After Reset: 0x0

MTL Queue 0 Receive Control Register

The Queue Receive Control register controls the receive arbitration and passing of received packets to the application.

MTL_RXQ0_CONTROL

MTL Queue 0 Receive Control Register (0D3C$_H$) Application Reset Value: 0000 0000$_H$



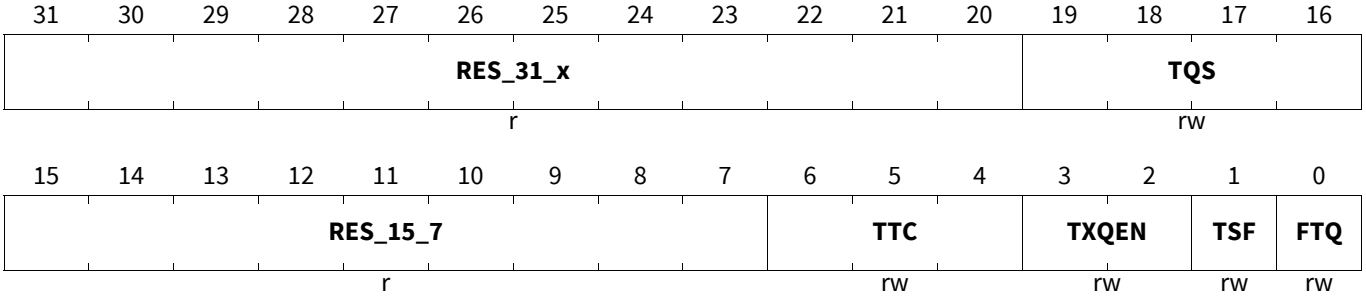
Field	Bits	Type	Description
RXQ_WEGT	2:0	rw	**Receive Queue Weight** This field indicates the weight assigned to the Rx Queue 0. The weight is used as the number of continuous PBL or packets requests (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle. Value After Reset: 0x0
RXQ_FRM_ARBIT	3	rw	**Receive Queue Packet Arbitration** When this bit is set, the DWC_ether_qos drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When this bit is reset, the DWC_ether_qos drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: • PBL amount of data (indicated by ari_qN_pbl_i[]) or • Complete data of a packet The status and the timestamp are not a part of the PBL data. Therefore, the DWC_ether_qos drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode). Value After Reset: 0x0
RES_31_4	31:4	r	**Reserved** Value After Reset: 0x0

MTL Queue i Transmit Operation Mode Register

The Queue 1 Transmit Operation Mode register establishes the Transmit queue operating modes and commands.

MTL_TXQi_OPERATION_MODE (i=1-3)

MTL Queue i Transmit Operation Mode Register($0D40_H$+(i-1)*40_H) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
FTQ	0	rw	**Flush Transmit Queue** When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the MTL_TxQ1_Operation_Mode register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission. **Note**: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (clk_tx_i) should be active. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
TSF	1	rw	**Transmit Store and Forward** When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped. Value After Reset: 0x0
TXQEN	3:2	rw	**Transmit Queue Enable** This field is used to enable/disable the transmit queue 0. **Note**: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field. Value After Reset: 0x0 00_B Not enabled 01_B Enable in AV mode 10_B Enabled 11_B Reserved

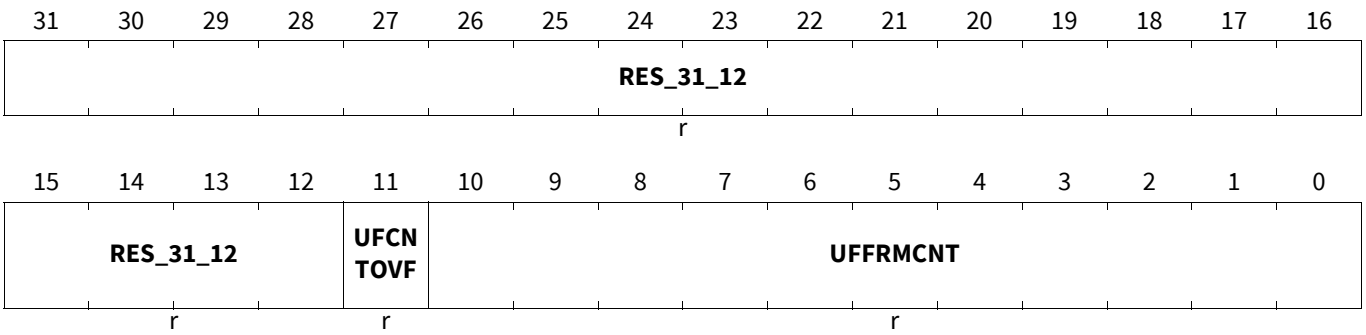
Field	Bits	Type	Description
TTC	6:4	rw	**Transmit Threshold Control** These bits control the threshold level of the MTL Tx Queue. The transmission starts when the packet size within the MTL Tx Queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset. Value After Reset: 0x0 000_B 32 001_B 64 010_B 96 011_B 128 100_B 192 101_B 256 110_B 384 111_B 512
RES_15_7	15:7	r	**Reserved** Value After Reset: 0x0
TQS	19:16	rw	**Transmit Queue Size** This field indicates the size of the allocated Transmit queues in blocks of 256 bytes. The TQS field is read-write only if the number of Tx Queues more than one, the reset value is 0x0 and indicates size of 256 bytes. When the number of Tx Queues is one, the field is read-only and the configured TX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Tx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(2048/256) = LOG2(8) = 3 bits Value After Reset: 0x0
RES_31_x	31:20	r	**Reserved** Value After Reset: 0x0

MTL Queue i Transmit Underflow Counter Register

The Queue 1 Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush

MTL_TXQi_UNDERFLOW (i=1-3)

MTL Queue i Transmit Underflow Counter Register($0D44_H$+(i-1)*40_H) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
UFFRMCNT	10:0	r	**Underflow Packet Counter** This field indicates the number of packets aborted by the controller because of Tx Queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
UFCNTOVF	11	r	**Overflow Bit for Underflow Packet Counter** This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_31_12	31:12	r	**Reserved** Value After Reset: 0x0

MTL Queue i Transmit Debug Register

The Queue 1 Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

MTL_TXQi_DEBUG (i=1-3)

MTL Queue i Transmit Debug Register **($0D48_H+(i-1)*40_H$)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_23									STXSTSF			RES_19	PTXQ		
r									r			r	r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_6										TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS		TXQPAUSED
r										r	r	r	r		r

Field	Bits	Type	Description
TXQPAUSED	0	r	**Transmit Queue in Pause** When this bit is high and the Rx flow control is enabled, it indicates that the Tx Queue is in the Pause condition (in the full-duplex only mode) because of the following: • Reception of the PFC packet for the priorities assigned to the Tx Queue when PFC is enabled • Reception of 802.3x Pause packet when PFC is disabled Value After Reset: 0x0

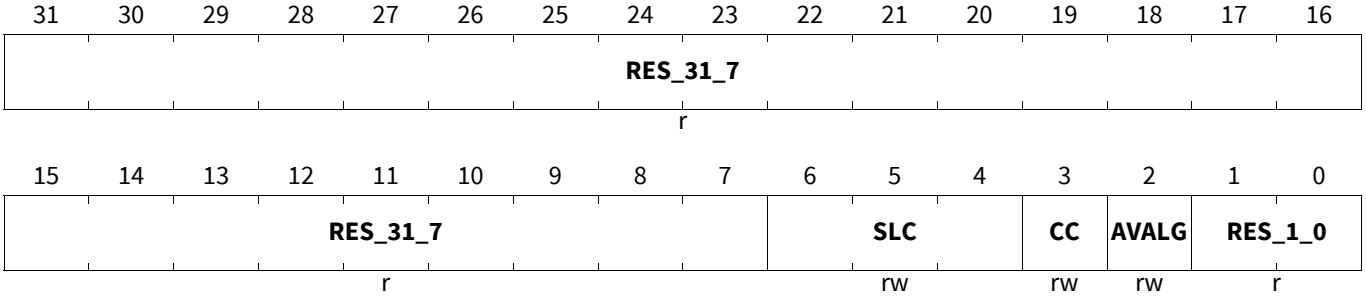
Field	Bits	Type	Description
TRCSTS	2:1	r	**MTL Tx Queue Read Controller Status** This field indicates the state of the Tx Queue Read Controller: Value After Reset: 0x0 00_B Idle state 01_B Read state (transferring data to the MAC transmitter) 10_B Waiting for pending Tx Status from the MAC transmitter 11_B Flushing the Tx queue because of the Packet Abort request from the MAC
TWCSTS	3	r	**MTL Tx Queue Write Controller Status** When high, this bit indicates that the MTL Tx Queue Write Controller is active, and it is transferring the data to the Tx Queue. Value After Reset: 0x0
TXQSTS	4	r	**MTL Tx Queue Not Empty Status** When this bit is high, it indicates that the MTL Tx Queue is not empty and some data is left for transmission. Value After Reset: 0x0
TXSTSFSTS	5	r	**MTL Tx Status FIFO Full Status** When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission. Value After Reset: 0x0
RES_15_6	15:6	r	**Reserved** Value After Reset: 0x0
PTXQ	18:16	r	**Number of Packets in the Transmit Queue** This field indicates the current number of packets in the Tx Queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of packets in the Transmit queue. Value After Reset: 0x0
RES_19	19	r	**Reserved** Value After Reset: 0x0
STXSTSF	22:20	r	**Number of Status Words in Tx Status FIFO of Queue** This field indicates the current number of status in the Tx Status FIFO of this queue. When the DTXSTS bit of MTL_Operation_Mode register is set to 1, this field does not reflect the number of status words in Tx Status FIFO. Value After Reset: 0x0
RES_31_23	31:23	r	**Reserved** Value After Reset: 0x0

MTL Queue i Transmit ETS Control Register

The Queue ETS Control register controls the enhanced transmission selection operation.

MTL_TXQi_ETS_CONTROL (i=1-3)

MTL Queue i Transmit ETS Control Register($0D50_H$+(i-1)*40_H) **Application Reset Value: $0000\ 0000_H$**



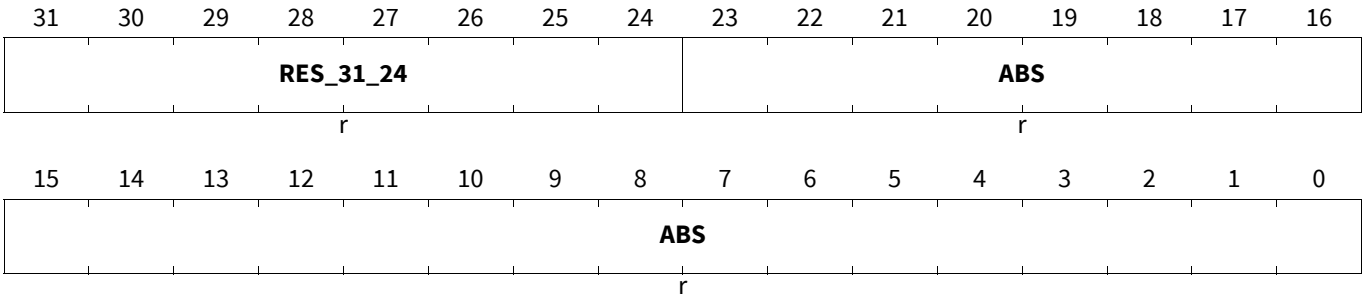
Field	Bits	Type	Description
RES_1_0	1:0	r	**Reserved** Value After Reset: 0x0
AVALG	2	rw	**AV Algorithm** When Queue 1 is programmed for AV, this field configures the scheduling algorithm for this queue: This bit when set, indicates credit based shaper algorithm (CBS) is selected for Queue 1 traffic. When reset, strict priority is selected. Value After Reset: 0x0
CC	3	rw	**Credit Control** When this bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in Channel 1. The credit accumulates even when there is no packet waiting in Channel 1 and another channel is transmitting. When this bit is reset, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in Channel 1. When there is no packet waiting in Channel 1 and other channel is transmitting, no credit is accumulated. Value After Reset: 0x0
SLC	6:4	rw	**Slot Count** If the credit-based shaper algorithm is enabled, the software can program the number of slots (of duration programmed in DMA_CH[n]_Slot_Interval register if present or 125us) over which the average transmitted bits per slot, provided in the MTL_TxQ[n]_ETS_Status register, need to be computed for Queue. The encoding is as follows: 101_B-111_B Reserved Value After Reset: 0x0 000_B 1 Slot 001_B 2 Slots 010_B 4 Slots 011_B 8 Slots 100_B 16 Slots
RES_31_7	31:7	r	**Reserved** Value After Reset: 0x0

MTL Queue i Transmit ETS Status Register

The Queue 1 ETS Status register provides the average traffic transmitted in Queue 1.

MTL_TXQi_ETS_STATUS (i=1-3)

MTL Queue i Transmit ETS Status Register (0D54$_H$+(i-1)*40$_H$) Application Reset Value: 0000 0000$_H$



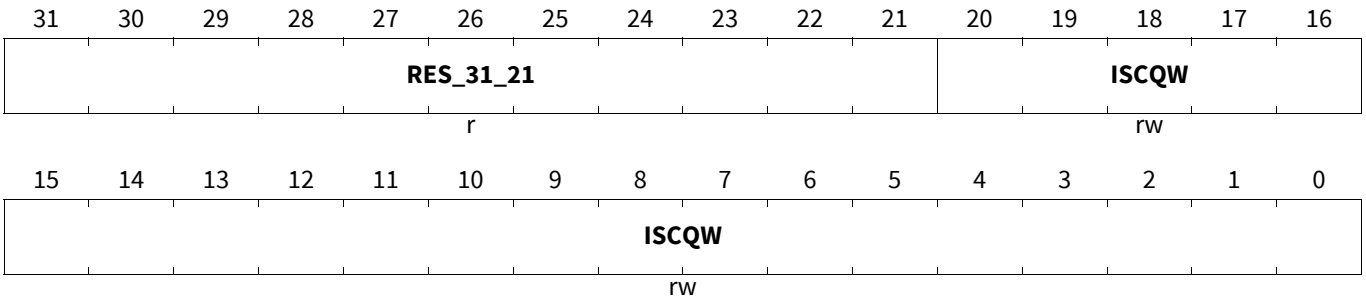
Field	Bits	Type	Description
ABS	23:0	r	**Average Bits per Slot** This field contains the average transmitted bits per slot. If AV operation is enabled, this field is computed over number of slots, programmed in the SLC field of MTL_TxQ[n]_ETS_CONTROL register. When Enable Slot Interval feature is selected, the maximum value of this field is 0x6_4000 in 100 Mbps, 0x3E_8000 in 1000 Mbps and 9C_4000 in 2500 Mbps mode respectively. Otherwise, the maximum value of this field is 0x30D4 in 100 Mbs, 0x1E848 in 1000 Mbps and 0x4C4B4 in 2500 Mbps respectively. When the DCB operation is enabled for Queue, this field is computed over every 10 million bit times slot (4 ms in 2500 Mbps; 10 ms in 1000 Mbps; 100 ms in 100 Mbps). The maximum value is 0x989680. This field is reserved in configurations with only one transmit queue. Value After Reset: 0x0
RES_31_24	31:24	r	**Reserved** Value After Reset: 0x0

MTL Queue i Transmit Quantum or Weights Register

The Queue 1 idleSlopeCredit, Quantum or Weights register provides the average traffic transmitted in Queue 1.

MTL_TXQi_QUANTUM_WEIGHT (i=1-3)

MTL Queue i Transmit Quantum or Weights Register(0D58$_H$+(i-1)*40$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
ISCQW	20:0	rw	**idleSlopeCredit, Quantum or Weights** **idleSlopeCredit** When AV feature is enabled, this field contains the idleSlopeCredit value required for the creditbased shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1000 Mbps; 3.2 ns for 2500 Mbps) when the credit is increasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000/2500 Mbps mode and 0x1000 in 100 Mbps mode. Bits[20:14] must be written to zero. **Quantum** When the DCB operation is enabled with DWRR algorithm for Queue 1 traffic, this field contains the quantum value in bytes to be added to credit during every queue scanning cycle. The maximum value is 0x1312D0 bytes. **Weights** When DCB operation is enabled with WFQ algorithm for Queue 1 traffic, this field contains the weight for this queue. The maximum value is 0x3FFF where weight of 0 indicates 100% bandwidth. Bits[20:14] must be written to zero. The higher the programmed weights lesser the bandwidth allocated for the particular Transmit Queue. This is because the weights are used to compute the packet finish time (weights*packet_size). Lesser the finish time, higher the probability of the packet getting scheduled first and using more bandwidth. When DCB operation or generic queueing operation is enabled with WRR algorithm for Queue 1 traffic, this field contains the weight for this queue. The maximum value is 0x64. Bits [20:7] must be written to zero. Note 1: In multiple Queue configuration this field in respective per queue register must be programmed to some non-zero value when multiple queues are enabled or single queue other than Q0 is enabled. This field need not be programmed when only Q0 is enabled. In general, when WRR algorithm is selected a non-zero value must be programmed on both Receive and Transmit. In Receive, the register is MTL_Operation_Mode register. Note 2: For WFQ algorithm, higher the programmed weights lesser the bandwidth allocated for that Transmit Queue. The finish time is not a function of particular packet alone but it is as per the formula: (previous_finish_time of particular Transmit Queue + (weights*packet_size)) Note 3: The weights programmed do not correspond to the number of packets but the fraction of bandwidth or time allocated for particular queue w.r.t. total BW or time. Value After Reset: 0x0
RES_31_21	31:21	r	**Reserved** Value After Reset: 0x0

MTL Queue i Transmit SendSlopeCredit Register

The sendSlopeCredit register contains the sendSlope credit value required for the credit-based shaper algorithm for the Queue.

MTL_TXQi_SENDSLOPECREDIT (i=1-3)

MTL Queue i Transmit SendSlopeCredit Register($0D5C_H$+(i-1)*40_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_14															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_31_14		SSC													
r		rw													

Field	Bits	Type	Description
SSC	13:0	rw	**sendSlopeCredit Value** When AV operation is enabled, this field contains the sendSlopeCredit value required for credit-based shaper algorithm for Queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps and 8 ns 1000 Mbps) when the credit is decreasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode. This field should be programmed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when Channel 1 is selected for transmission. Value After Reset: 0x0
RES_31_14	31:14	r	**Reserved** Value After Reset: 0x0

MTL Queue i Transmit HiCredit Register

The hiCredit register contains the hiCredit value required for the credit-based shaper algorithm for the Queue.

MTL_TXQi_HICREDIT (i=1-3)

MTL Queue i Transmit HiCredit Register **($0D60_H$+(i-1)*40_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_29			HC												
r			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HC															
rw															

Field	Bits	Type	Description
HC	28:0	rw	**hiCredit Value** When the AV feature is enabled, this field contains the hiCredit value required for the credit-based shaper algorithm. This is the maximum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16,384 bytes or 131,072 bits). The value to be specified is 131,072 * 1,024 = 134,217,728 or 0x0800_0000. Value After Reset: 0x0
RES_31_29	31:29	r	**Reserved** Value After Reset: 0x0

MTL Queue i Transmit LoCredit Register

The loCredit register contains the loCredit value required for the credit-based shaper algorithm for the Queue.

MTL_TXQi_LOCREDIT (i=1-3)

MTL Queue i Transmit LoCredit Register (0D64$_H$+(i-1)*40$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_29			LC												
r			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LC															
rw															

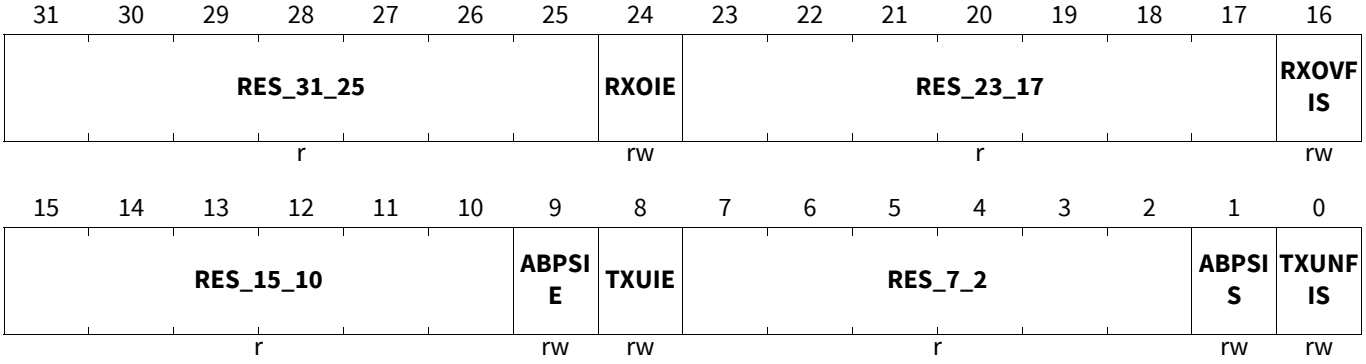
Field	Bits	Type	Description
LC	28:0	rw	**loCredit Value** When AV operation is enabled, this field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1,024. The maximum value to be programmed corresponds to twice the maxFrameSize transmitted from this queue. If the maxFrameSize is 8192 bytes, then (8192*2) * 8 * 1024 = 134,217,728 or 0x0800_0000. Because it is a negative value, the programmed value is 2's complement of the value, that is, 0x1800_0000. Value After Reset: 0x0
RES_31_29	31:29	r	**Reserved** Value After Reset: 0x0

MTL Queue i Interrupt Status Register

This register contains the interrupt enable and status bits for the queue 1 interrupts.

MTL_Qi_INTERRUPT_CONTROL_STATUS (i=1-3)

MTL Queue i Interrupt Status Register **($0D6C_H$+(i-1)*40_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
TXUNFIS	0	rw	**Transmit Queue Underflow Interrupt Status** This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
ABPSIS	1	rw	**Average Bits Per Slot Interrupt Status** When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
RES_7_2	7:2	r	**Reserved** Value After Reset: 0x0
TXUIE	8	rw	**Transmit Queue Underflow Interrupt Enable** When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled. Value After Reset: 0x0
ABPSIE	9	rw	**Average Bits Per Slot Interrupt Enable** When this bit is set, the MAC asserts the sbd_intr_o or mci_intr_o interrupt when the average bits per slot status is updated. When this bit is cleared, the interrupt is not asserted for such an event. Value After Reset: 0x0
RES_15_10	15:10	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
RXOVFIS	16	rw	**Receive Queue Overflow Interrupt Status** This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
RES_23_17	23:17	r	**Reserved** Value After Reset: 0x0
RXOIE	24	rw	**Receive Queue Overflow Interrupt Enable** When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled. Value After Reset: 0x0
RES_31_25	31:25	r	**Reserved** Value After Reset: 0x0

MTL Queue i Receive Operation Mode Register

The Queue 1 Receive Operation Mode register establishes the Receive queue operating modes and command. The RFA and RFD fields are not backward compatible with the RFA and RFD fields of 4.00a release

MTL_RXQi_OPERATION_MODE (i=1-3)

MTL Queue i Receive Operation Mode Register($0D70_H$+(i-1)*40_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_x							RQS					RES_19_x		RFD	
r							rw					r		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFD		RES_13_x		RFA				EHFC	DIS_TCP_EF	RSF	FEP	FUP	RES_2	RTC	
rw		r		rw				rw	rw	rw	rw	rw	r	rw	

Field	Bits	Type	Description
RTC	1:0	rw	**Receive Queue Threshold Control** These bits control the threshold level of the MTL Rx queue (in bytes): The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred. This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1. Value After Reset: 0x0 00_B 64 01_B 32 10_B 96 11_B 128
RES_2	2	r	**Reserved** Value After Reset: 0x0
FUP	3	rw	**Forward Undersized Good Packets** When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01. Value After Reset: 0x0
FEP	4	rw	**Forward Error Packets** When this bit is reset, the Rx queue drops packets with error status (CRC error, GMII_ER, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped. When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA. Value After Reset: 0x0
RSF	5	rw	**Receive Queue Store and Forward** When this bit is set, the DWC_ether_qos reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register. Value After Reset: 0x0

Field	Bits	Type	Description
DIS_TCP_EF	6	rw	**Disable Dropping of TCP/IP Checksum Error Packets** When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC. When this bit is reset, all error packets are dropped if the FEP bit is reset. Value After Reset: 0x0
EHFC	7	rw	**Enable Hardware Flow Control** When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled. This bit is not used (reserved and always reset) when the Rx queue is less than 4 KB. Value After Reset: 0x0
RFA	11:8	rw	**Threshold for Activating Flow Control (in half-duplex and full-duplex** These bits control the threshold (fill-level of Rx queue) at which the flow control is activated: For more information on encoding for this field, see RFD. Value After Reset: 0x0
RES_13_x	13:12	r	**Reserved** Value After Reset: 0x0
RFD	17:14	rw	**Threshold for Deactivating Flow Control (in half-duplex and full-duplex modes)** These bits control the threshold (fill-level of Rx queue) at which the flow control is de-asserted after activation: LOST SEQUENCE DEFINITION The de-assertion is effective only after flow control is asserted. **Note:** The value must be programmed in such a way to make sure that the threshold is a positive number. When the EHFC is set high, these values are applicable only when the Rx queue size determined by the RQS field of this register, is equal to or greater than 4 KB. For a given queue size, the values ranges between 0 and the encoding for FULL minus (QSIZE - 0.5 KB) and all other values are illegal. Here the term FULL and QSIZE refers to the queue size determined by the RQS field of this register. The width of this field depends on RX FIFO size selected during the configuration. Remaining bits are reserved and read only. Value After Reset: 0x0 0_H Full minus 1 KB, that is, FULL 1 KB 1_H Full minus 1.5 KB, that is, FULL 1.5 KB 2_H Full minus 2 KB, that is, FULL 2 KB 3_H Full minus 2.5 KB, that is, FULL 2.5 KB E_H Full minus 8 KB, that is, FULL 8 KB F_H Full minus 8.5 KB, that is, FULL 8.5 KB
RES_19_x	19:18	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
RQS	24:20	rw	**Receive Queue Size** This field indicates the size of the allocated Receive queues in blocks of 256 bytes. The RQS field is read-write only if the number of Rx Queues more than one and the reset value is 0x0 and indicates size of 256 bytes. When the number of Rx Queues is one, the field is read-only and the configured RX FIFO size in blocks of 256 bytes is reflected in the reset value. The width of this field depends on the Rx memory size selected in your configuration. For example, if the memory size is 2048, the width of this field is 3 bits: LOG2(2048/256) = LOG2(8) = 3 bits Value After Reset: 0x0
RES_31_x	31:25	r	**Reserved** Value After Reset: 0x0

MTL Queue i Receive Missed Packet and Overflow Counter Register

The Queue 1 Missed Packet and Overflow Counter register contains the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

MTL_RXQi_MISSED_PACKET_OVERFLOW_CNT (i=1-3)
MTL Queue i Receive Missed Packet and Overflow Counter Register($0D74_H+(i-1)*40_H$) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_28				MISCNTOVF	MISPKTCNT										
r				r	r										

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_12				OVFCNTOVF	OVFPKTCNT										
r				r	r										

Field	Bits	Type	Description
OVFPKTCNT	10:0	r	**Overflow Packet Counter** This field indicates the number of packets discarded by the DWC_ether_qos because of Receive queue overflow. This counter is incremented each time the DWC_ether_qos discards an incoming packet because of overflow. This counter is reset when this register is read with mci_be_i[0] at 1'b1. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0

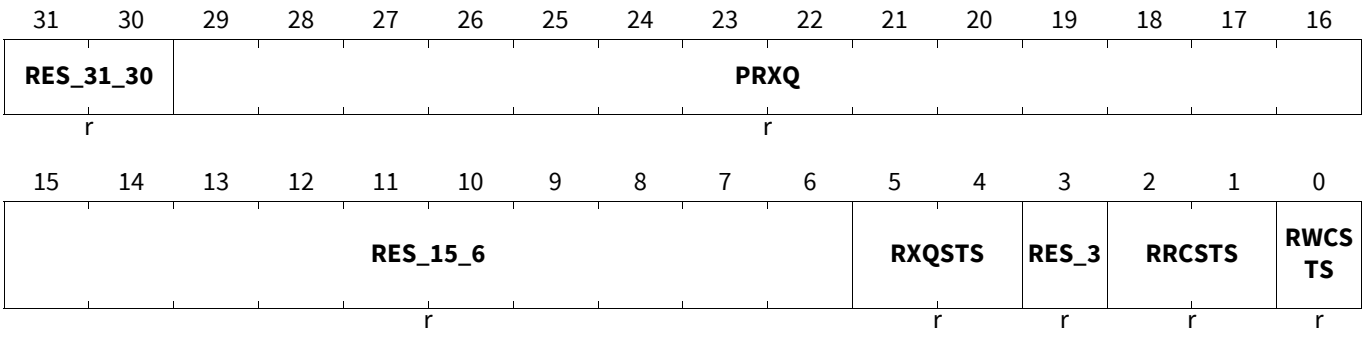
Field	Bits	Type	Description
OVFCNTOVF	11	r	**Overflow Counter Overflow Bit** When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_15_12	15:12	r	**Reserved** Value After Reset: 0x0
MISPKTCNT	26:16	r	**Missed Packet Counter** This field indicates the number of packets missed by the DWC_ether_qos because the application asserted ari_pkt_flush_i[] for this queue. This counter is reset when this register is read with mci_be_i[0] at 1b1. This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
MISCNTOVF	27	r	**Missed Packet Counter Overflow Bit** When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_31_28	31:28	r	**Reserved** Value After Reset: 0x0

MTL Queue i Receive Debug Register

The Queue 1 Receive Debug register gives the debug status of various blocks related to the Receive queue.

MTL_RXQi_DEBUG (i=1-3)

MTL Queue i Receive Debug Register **($0D78_H+(i-1)*40_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RWCSTS	0	r	**MTL Rx Queue Write Controller Active Status** When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx Queue. Value After Reset: 0x0

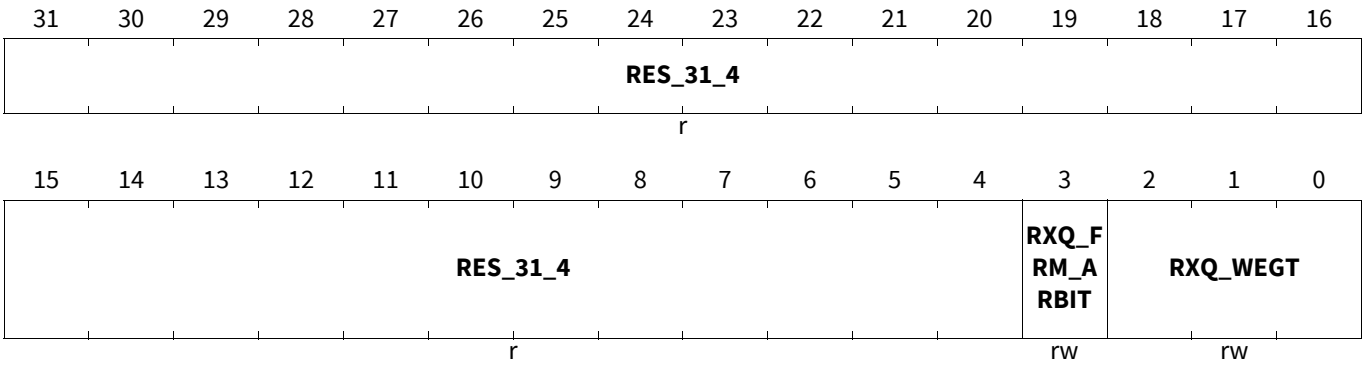
Field	Bits	Type	Description
RRCSTS	2:1	r	**MTL Rx Queue Read Controller State** This field gives the state of the Rx queue Read controller: Value After Reset: 0x0 00_B Idle state 01_B Reading packet data 10_B Reading packet status (or timestamp) 11_B Flushing the packet data and status
RES_3	3	r	**Reserved** Value After Reset: 0x0
RXQSTS	5:4	r	**MTL Rx Queue Fill-Level Status** This field gives the status of the fill-level of the Rx Queue: Value After Reset: 0x0 00_B Rx Queue empty 01_B Rx Queue fill-level below flow-control deactivate threshold 10_B Rx Queue fill-level above flow-control activate threshold 11_B Rx Queue full
RES_15_6	15:6	r	**Reserved** Value After Reset: 0x0
PRXQ	29:16	r	**Number of Packets in Receive Queue** This field indicates the current number of packets in the Rx Queue. The theoretical maximum value for this field is 256KB/16B = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size. Value After Reset: 0x0
RES_31_30	31:30	r	**Reserved** Value After Reset: 0x0

MTL Queue i Receive Control Register

The Queue Receive Control register controls the receive arbitration and passing of received packets to the application.

MTL_RXQi_CONTROL (i=1-3)

MTL Queue i Receive Control Register **($0D7C_H$+(i-1)*40_H)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
RXQ_WEGT	2:0	rw	**Receive Queue Weight** This field indicates the weight assigned to the Rx Queue 0. The weight is used as the number of continuous PBL or packets requests (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle. Value After Reset: 0x0
RXQ_FRM_ARBIT	3	rw	**Receive Queue Packet Arbitration** When this bit is set, the DWC_ether_qos drives the packet data to the ARI interface such that the entire packet data of currently-selected queue is transmitted before switching to other queue. When this bit is reset, the DWC_ether_qos drives the packet data to the ARI interface such that the following amount of data of currently-selected queue is transmitted before switching to other queue: • PBL amount of data (indicated by ari_qN_pbl_i[]) or • Complete data of a packet The status and the timestamp are not a part of the PBL data. Therefore, the DWC_ether_qos drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode). Value After Reset: 0x0
RES_31_4	31:4	r	**Reserved** Value After Reset: 0x0

DMA Bus Mode Register

The Bus Mode register establishes the bus operating modes for the DMA.

DMA_MODE

DMA Bus Mode Register **(1000_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_18_31														INTM	
r														rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15	PR			TXPR	RES_10	ARBC	RES_8_5				TAA			DA	SWR
r	rw			rw	r	rw	r				rw			rw	rw

Field	Bits	Type	Description
SWR	0	rw	**Software Reset** When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all DWC_ether_qos clock domains. Before reprogramming any DWC_ether_qos register, a value of zero should be read in this bit. This bit must be read at least 4 CSR clock cycles after it is written to 1. **Note**: The reset operation is complete only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock. Access restriction applies. Setting 1 sets. Self-cleared. Setting 0 has no effect. Value After Reset: 0x0
DA	1	rw	**DMA Tx or Rx Arbitration Scheme** This bit specifies the arbitration scheme between the Transmit and Receive paths of all channels: The priority between the paths is according to the priority specified in Bits[14:12] and the priority weight is specified in the TXPR bit. The Tx path has priority over the Rx path when the TXPR bit is set. Otherwise, the Rx path has priority over the Tx path. Value After Reset: 0x0 0_B Weighted Round-Robin with Rx:Tx or Tx:Rx 1_B Fixed Priority
TAA	4:2	rw	**Transmit Arbitration Algorithm** This field is used to select the arbitration algorithm for the Transmit side when multiple Tx DMAs are selected. $011\text{-}111_B$ Reserved Value After Reset: 0x0 000_B Fixed priority. In fixed priority, Channel 0 has the lowest priority and the last channel has the highest priority. 001_B Weighted Strict Priority (WSP) 010_B Weighted Round-Robin (WRR)
RES_8_5	8:5	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
ARBC	9	rw	**Arbitration Control** When this bit is reset to 0, the DMA arbitrates as soon as the current burst transaction is completed. So, in this mode, the arbiter will miss the next/subsequent request from the same DMA since there are inherent delays between the completion of 1 transfer request and the next transaction request generated from the same DMA. Consequently, the WRR ratio becomes 1:1 irrespective of the programmed ratio. When this bit is set to 1, the DMA arbitration is delayed for a few clock cycles after the completion of current transaction so that the next request from the same DMA can be considered. This allows the arbiter to maintain WRR ratio as programmed.
RES_10	10	r	**Reserved** Value After Reset: 0x0
TXPR	11	rw	**Transmit Priority** When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus. Value After Reset: 0x0
PR	14:12	rw	**Priority Ratio** These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether the TXPR bit is reset or set. Value After Reset: 0x0 000_B The priority ratio is 1:1 001_B The priority ratio is 2:1 010_B The priority ratio is 3:1 011_B The priority ratio is 4:1 100_B The priority ratio is 5:1 101_B The priority ratio is 6:1 110_B The priority ratio is 7:1 111_B The priority ratio is 8:1
RES_15	15	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
INTM	17:16	rw	**Interrupt Mode** This field defines the interrupt mode of DWC_ether_qos. The behavior of the following outputs changes depending on the following settings: • sbd_perch_tx_intr_o[] (Transmit Per Channel Interrupt) • sbd_perch_rx_intr_o[] (Receive Per Channel Interrupt) • sbd_intr_o (Common Interrupt) It also changes the behavior of the RI/TI bits in the DMA_CH0_Status. For more details please refer Table "DWC_ether_qos Transfer Complete Interrupt Behavior". Value After Reset: 0x0 00_B sbd_perch_* are pulse signals for each completion events. sbd_intr_o is also asserted and cleared only when software clears the corresponding RI/TI status bits 01_B sbd_perch_* are level signals asserted on corresponding event and de-asserted when the software clears the corresponding RI/TI status bits. The sbd_intr_o is not asserted for these packet transfer completion events. 10_B sbd_perch_* are level signals asserted on corresponding event and de-asserted when the software clears the corresponding RI/TI status bits. However, the signal is asserted again if the same event occurred again before it was cleared. The sbd_intr_o is not asserted for these packet transfer completion events. 11_B Reserved
RES_18_31	31:18	r	**Reserved** Value After Reset: 0x0

DMA System Bus Mode Register

The System Bus mode register controls the behavior of the AHB or AXI master. It mainly controls burst splitting and number of outstanding requests.

DMA_SYSBUS_MODE

DMA System Bus Mode Register **(1004_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_30		RES_28_x			RES_26_24			RES_23_x					RES_18_16		
r		r			r			r					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RB	MB	RES_13	AAL	RES_11_10		RES_9_8		RES_7_1							FB
rw	rw	r	rw	r		r		r							rw

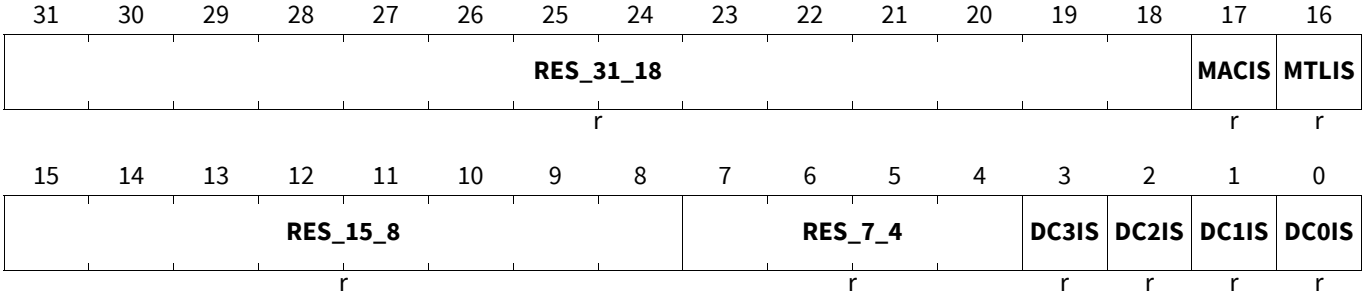
Field	Bits	Type	Description
FB	0	rw	**Fixed Burst Length** When this bit is set to 1, the AHB master initiates burst transfers of specified length (INCRx or SINGLE). When this bit is set to 0, the AHB master initiates transfers of unspecified length (INCR) or SINGLE transfers. Value After Reset: 0x0
RES_7_1	7:1	r	**Reserved for future use.**
RES_9_8	9:8	r	**Reserved** Value After Reset: 0x0
RES_11_10	11:10	r	**Reserved for future use.**
AAL	12	rw	**Address-Aligned Beats** When this bit is set to 1, the EQOS-AXI or EQOS-AHB master performs address-aligned burst transfers on Read and Write channels. Value After Reset: 0x0
RES_13	13	r	**Reserved for future use.**
MB	14	rw	**Mixed Burst** **When this bit is set high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE).** Value After Reset: 0x0
RB	15	rw	**Rebuild INCRx Burst** When this bit is set high and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst. Value After Reset: 0x0
RES_18_16	18:16	r	**Reserved for future use.**
RES_23_x	23:19	r	**Reserved** Value After Reset: 0x0
RES_26_24	26:24	r	**Reserved for future use.**
RES_28_x	29:27	r	**Reserved** Value After Reset: 0x0
RES_31_30	31:30	r	**Reserved for future use.**

DMA Interrupt Status Register

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

DMA_INTERRUPT_STATUS

DMA Interrupt Status Register **(1008_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
DC0IS	0	r	**DMA Channel 0 Interrupt Status** This bit indicates an interrupt event in DMA Channel 0. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 0 to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0
DC1IS	1	r	**DMA Channel 1 Interrupt Status** This bit indicates an interrupt event in DMA Channel 1. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 1 to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0
DC2IS	2	r	**DMA Channel 2 Interrupt Status** This bit indicates an interrupt event in DMA Channel 2. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 2 to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0
DC3IS	3	r	**DMA Channel 3 Interrupt Status** This bit indicates an interrupt event in DMA Channel 3. To reset this bit to 1'b0, the software must read the corresponding register in DMA Channel 3 to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0
RES_7_4	7:4	r	**Reserved for future use.**
RES_15_8	15:8	r	**Reserved** Value After Reset: 0x0
MTLIS	16	r	**MTL Interrupt Status** This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0
MACIS	17	r	**MAC Interrupt Status** This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source. Value After Reset: 0x0

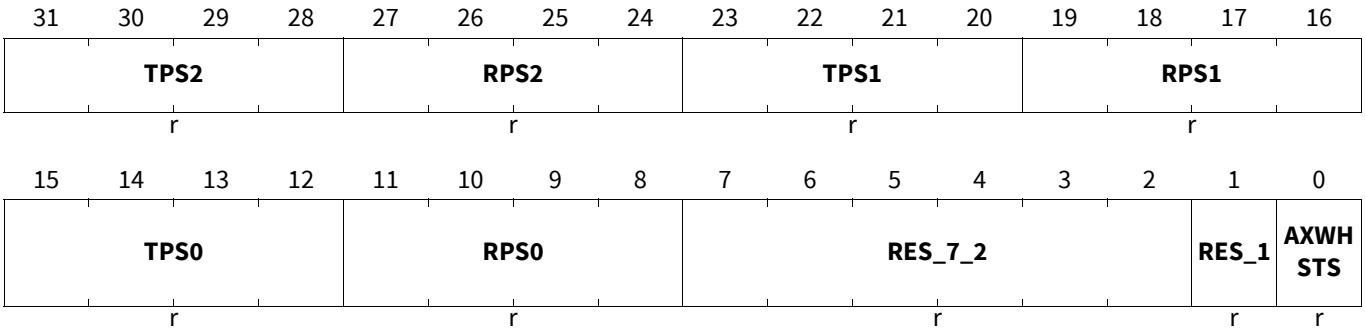
Field	Bits	Type	Description
RES_31_18	31:18	r	**Reserved** Value After Reset: 0x0

DMA Debug Status 0 Register

The Debug Status 0 register gives the Receive and Transmit process status for DMA Channel 0-Channel 2 for debugging purpose.

DMA_DEBUG_STATUS0

DMA Debug Status 0 Register **($100C_H$)** **Application Reset Value: $0000\ 0000_H$**



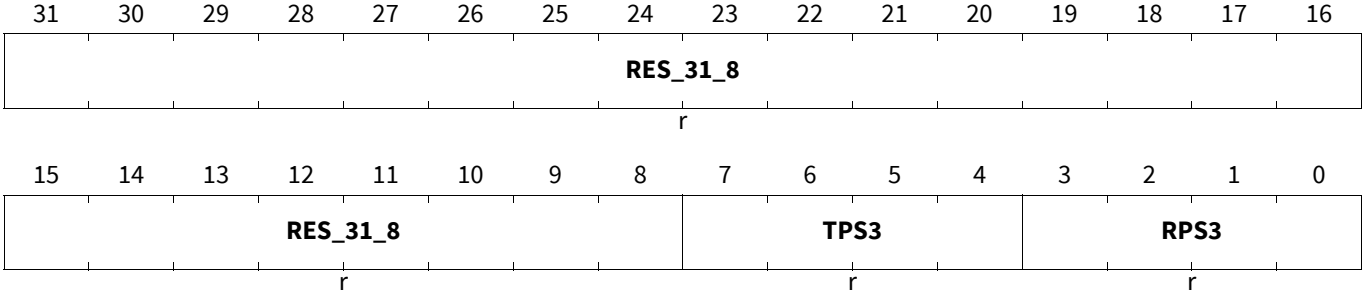
Field	Bits	Type	Description
AXWHSTS	0	r	**AHB Master Status** When high, this bit indicates that the AHB master FSMs are in the non-idle state. Value After Reset: 0x0
RES_1	1	r	**Reserved for future use.**
RES_7_2	7:2	r	**Reserved** Value After Reset: 0x0
RPS0	11:8	r	**DMA Channel 0 Receive Process State** This field indicates the Rx DMA FSM state for Channel 0: The MSB of this field always returns 0. This field does not generate an interrupt. Value After Reset: 0x0 0_H Stopped (Reset or Stop Receive Command issued) 1_H Running (Fetching Rx Transfer Descriptor) 2_H Reserved for future use 3_H Running (Waiting for Rx packet) 4_H Suspended (Rx Descriptor Unavailable) 5_H Running (Closing the Rx Descriptor) 6_H Timestamp write state 7_H Running (Transferring the received packet data from the Rx buffer to the system memory)

Field	Bits	Type	Description
TPS0	15:12	r	**DMA Channel 0 Transmit Process State** This field indicates the Tx DMA FSM state for Channel 0: The MSB of this field always returns 0. This field does not generate an interrupt. Value After Reset: 0x0 0_H Stopped (Reset or Stop Transmit Command issued) 1_H Running (Fetching Tx Transfer Descriptor) 2_H Running (Waiting for status) 3_H Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO)) 4_H Timestamp write state 5_H Reserved for future use 6_H Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow) 7_H Running (Closing Tx Descriptor)
RPS1	19:16	r	**DMA Channel 1 Receive Process State** This field indicates the Rx DMA FSM state for Channel 1. This field is similar to the RPS0 field. Value After Reset: 0x0
TPS1	23:20	r	**DMA Channel 1 Transmit Process State** This field indicates the Tx DMA FSM state for Channel 1. This field is similar to the TPS0 field. Value After Reset: 0x0
RPS2	27:24	r	**DMA Channel 2 Receive Process State** This field indicates the Rx DMA FSM state for Channel 2. This field is similar to the RPS0 field Value After Reset: 0x0
TPS2	31:28	r	**DMA Channel 2 Transmit Process State** This field indicates the Tx DMA FSM state for Channel 2. This field is similar to the TPS0 field. Value After Reset: 0x0

DMA Debug Status 1 Register

The Debug Status1 register gives the Receive and Transmit process status for DMA Channel 3-Channel 6.

DMA_DEBUG_STATUS1
DMA Debug Status 1 Register **(1010_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RPS3	3:0	r	**DMA Channel 3 Receive Process State** This field indicates the Rx DMA FSM state for Channel 3. This field is similar to the RPS6 field. Value After Reset: 0x0
TPS3	7:4	r	**DMA Channel 3 Transmit Process State** This field indicates the Tx DMA FSM state for Channel 3. This field is similar to the TPS6 field. Value After Reset: 0x0
RES_31_8	31:8	r	**Reserved for future use.**

DMA Channel i Control Register

The DMA Channel i Control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

DMA_CHi_CONTROL (i=0-3)
DMA Channel i Control Register **(1100_H+i*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_25							SPH	RES_23_21			DSL			RES_17	PBLx8
r							rw	r			rw			r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15_14		RES_13_0													
r		r													

Field	Bits	Type	Description
RES_13_0	13:0	r	**Reserved for future use.**
RES_15_14	15:14	r	**Reserved** Value After Reset: 0x0
PBLx8	16	rw	**8xPBL mode** When this bit is set, the PBL value programmed in Bits[21:16] in DMA_CH0_Tx_Control and Bits[21:16] in DMA_CH0_Rx_Control is multiplied by eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value. Value After Reset: 0x0
RES_17	17	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
DSL	20:18	rw	**Descriptor Skip Length** This bit specifies the Word, Dword, or Lword number (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor. When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous. Value After Reset: 0x0
RES_23_21	23:21	r	**Reserved** Value After Reset: 0x0
SPH	24	rw	**Split Headers** When this bit is set, the DMA splits the header and payload in the Receive path. The DMA writes the header to the Buffer Address1 of RDES0. The DMA writes the payload to the buffer to which the Buffer Address2 is pointing. The software must ensure that the header fits into the Receive buffers. If the header length exceeds the receive buffer size, the DMA does not split the header and payload. This bit is available only if Enable Split Header Structure option is selected. Value After Reset: 0x0
RES_31_25	31:25	r	**Reserved** Value After Reset: 0x0

DMA Channel i Transmit Control Register

The DMA Channel i Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

DMA_CHi_TX_CONTROL (i=0-3)
DMA Channel i Transmit Control Register **(1104_H+i*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_28				RES_27_24				RES_23_22		TxPBL					
r				r				r		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15	RES_14_13		RES_12	RES_11_5							OSF	TCW			ST
r	r		r	r							rw	rw			rw

Field	Bits	Type	Description
ST	0	rw	**Start or Stop Transmission Command** When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted. The DMA tries to acquire descriptor from either of the following positions: • The current position in the list This is the base address of the Transmit list set by the DMA_CH0_TxDesc_List_Address register. • The position at which the transmission was previously stopped If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the DMA_CH0_Status register is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the DMA_CH0_TxDesc_List_Address register, the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program DMA_CH0_TxDesc_List_Address register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state. Value After Reset: 0x0
TCW	3:1	rw	**Transmit Channel Weight** This field indicates the weight assigned to the corresponding Transmit channel. When reset is complete, this field is set to 0 for all channels by default, resulting in equal weights to all channels. Value After Reset: 0x0
OSF	4	rw	**Operate on Second Packet** When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained. Value After Reset: 0x0
RES_11_5	11:5	r	**Reserved** Value After Reset: 0x0
RES_12	12	r	**Reserved for future use.**
RES_14_13	14:13	r	**Reserved** Value After Reset: 0x0
RES_15	15	r	**Reserved for future use.**

Field	Bits	Type	Description
TxPBL	21:16	rw	**Transmit Programmable Burst Length** These bits indicate the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the 8xPBL mode in DMA_CH0_Control register. 2. Set the TxPBL. Note: The maximum value of TxPBL must be less than or equal to half the Tx Queue size (TQS field of MTL_TxQ[n]_Operation_Mode register) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. For example, in 64-bit data width configurations the total locations in Tx Queue of size 512 bytes is 64, TxPBL and 8xPBL needs to be programmed to less than or equal to 32. Value After Reset: 0x0
RES_23_22	23:22	r	**Reserved** Value After Reset: 0x0
RES_27_24	27:24	r	**Reserved for future use.**
RES_31_28	31:28	r	**Reserved** Value After Reset: 0x0

DMA Channel i Receive Control Register

The DMA Channel i Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

DMA_CHi_RX_CONTROL (i=0-3)

DMA Channel i Receive Control Register **(1108_H+i*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPF	**RES_30_28**			**RES_27_24**				**RES_23_22**		**RxPBL**					
rw	r			r				r		rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES_15	**RBSZ_13_y**												**RBSZ_x_0**		**SR**
r	rw												r		rw

Field	Bits	Type	Description
SR	0	rw	**Start or Stop Receive** When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets. The DMA tries to acquire descriptor from either of the following positions: • The current position in the list This is the address set by the DMA_CH0_RxDesc_List_Address register. • The position at which the Rx process was previously stopped If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the DMA_CH0_Status register is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the DMA_CH0_RxDesc_List_Address register, the DMA behavior is unpredictable. When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state. Value After Reset: 0x0
RBSZ_x_0	2:1	r	**Receive Buffer size Low** RBSZ[13:0] is split into two fields RBSZ_13_y and RBSZ_x_0. The RBSZ_x_0 is the lower field whose width is based on data bus width of the configuration. This field is of width 2, 3, or 4 bits for 32-bit, 64-bit, or 128-bit data bus width respectively. This field is read-only (RO). Value After Reset: 0x0
RBSZ_13_y	14:3	rw	**Receive Buffer size High** RBSZ[13:0] is split into two fields higher RBSZ_13_y and lower RBSZ_x_0. The RBSZ[13:0] field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16K bytes. The buffer size is applicable to payload buffers when split headers are enabled. **Note**: The buffer size must be a multiple of 4, 8, or 16 depending on the data bus widths (32-bit, 64-bit, or 128-bit respectively). This is required even if the value of buffer address pointer is not aligned to data bus width. Hence the lower RBSZ_x_0 bits are read-only and the value is considered as allzero. Thus the RBSZ_13_y indicates the buffer size in terms of locations (with the width same as bus-width). Value After Reset: 0x0
RES_15	15	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
RxPBL	21:16	rw	**Receive Programmable Burst Length** These bits indicate the maximum number of beats to be transferred in one DMA block data transfer. The DMA always attempts max burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior. To transfer more than 32 beats, perform the following steps: 1. Set the 8xPBL mode in the DMA_CH0_Control register. 2. Set the RxPBL. Note: The maximum value of RxPBL must be less than or equal to half the Rx Queue size (RQS field of MTL_RxQ[n]_Operation_Mode register) in terms of beats. This is required so that the Rx Queue has space to store at least another Rx PBL worth of data while the Rx DMA is transferring a block of data. For example, in 64-bit data width configurations the total locations in Rx Queue of size 512 bytes is 64, so RxPBL and 8xPBL needs to be programmed to less than or equal to 32. Value After Reset: 0x0
RES_23_22	23:22	r	**Reserved** Value After Reset: 0x0
RES_27_24	27:24	r	**Reserved for future use.**
RES_30_28	30:28	r	**Reserved** Value After Reset: 0x0
RPF	31	rw	**DMA Rx Channel0 Packet Flush** When this bit is set to 1, then DWC_ether_qos automatically flushes the packet from the Rx Queues destined to this DMA Rx Channel, when it is stopped. When this bit remains set and the DMA is re-started by the software driver, the packets residing in the Rx Queues that were received when this RxDMA was stopped, get flushed out. The packets that are received by the MAC after the RxDMA is re-started are routed to the RxDMA. The flushing happens on the Read side of the Rx Queue. When this bit is set to 0, the DWC_ether_qos not flush the packet in the Rx Queue destined to this RxDMA Channel when it is STOP state. This may in turn cause head-of-line blocking in the corresponding RxQueue. Value After Reset: 0x0

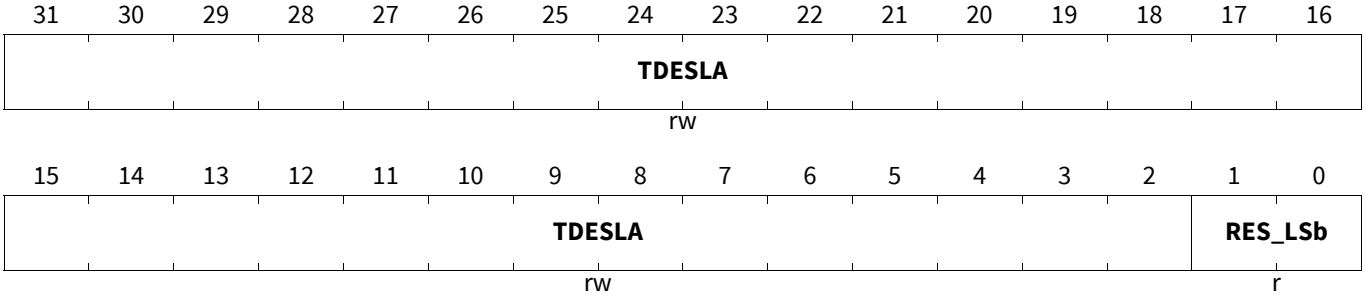
DMA Channel i Transmit Descriptor List Address Register

The DMA Channel i Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

You can write to this register only when the Tx DMA has stopped, that is, the ST bit is set to zero in DMA_CH0_Tx_Control register. When stopped, this register can be written with a new descriptor list address. When you set the ST bit to 1, the DMA takes the newly-programmed descriptor base address. If this register is not changed when the ST bit is set to 0, the DMA takes the descriptor address where it was stopped earlier.

DMA_CHi_TXDESC_LIST_ADDRESS (i=0-3)

DMA Channel i Transmit Descriptor List Address Register(1114_H+i*80_H)Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
RES_LSb	1:0	r	**Reserved** Value After Reset: 0x0
TDESLA	31:2	rw	**Start of Transmit List** This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO). The width of this field depends on the configuration: • 31:2 for 32-bit configuration • 31:3 for 64-bit configuration • 31:4 for 128-bit configuration Value After Reset: 0x0

DMA Channel i Receive Descriptor List Address Register

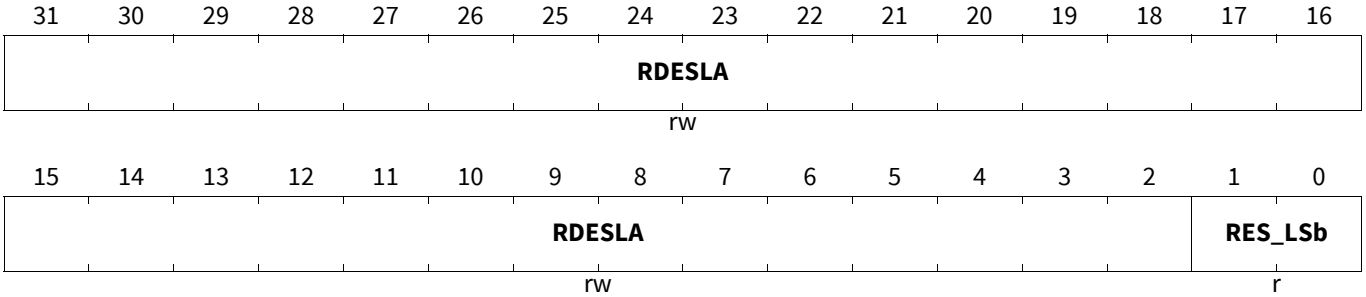
The DMA Channel i Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be Word, Dword, or Lword-aligned (for 32-bit, 64-bit, or 128-bit data bus). The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in DMA_CH0_Rx_Control register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

DMA_CHi_RXDESC_LIST_ADDRESS (i=0-3)

DMA Channel i Receive Descriptor List Address Register($111C_H$+i*80_H) Application Reset Value: 0000 0000_H



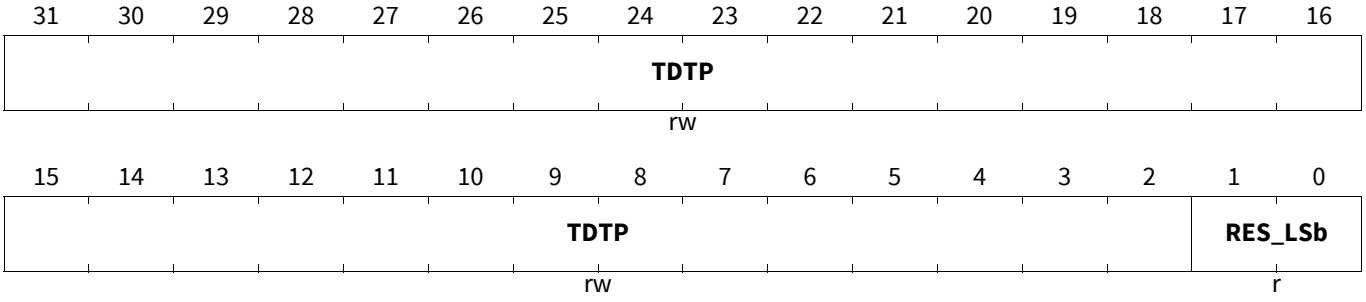
Field	Bits	Type	Description
RES_LSb	1:0	r	**Reserved** Value After Reset: 0x0
RDESLA	31:2	rw	**Start of Receive List** This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (1:0, 2:0, or 3:0) for 32-bit, 64-bit, or 128-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO). The width of this field depends on the configuration: • 31:2 for 32-bit configuration • 31:3 for 64-bit configuration • 31:4 for 128-bit configuration Value After Reset: 0x0

DMA Channel i Transmit Descriptor Tail Pointer Register

The DMA Channel i Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

DMA_CHi_TXDESC_TAIL_POINTER (i=0-3)

DMA Channel i Transmit Descriptor Tail Pointer Register(1120_H+i*80_H)Application Reset Value: 0000 0000_H



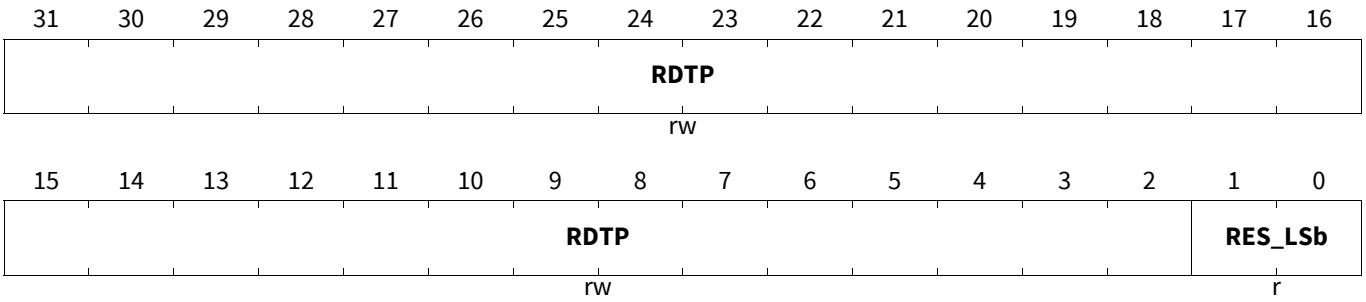
Field	Bits	Type	Description
RES_LSb	1:0	r	**Reserved** Value After Reset: 0x0
TDTP	31:2	rw	**Transmit Descriptor Tail Pointer** This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers. The width of this field depends on the configuration: • 31:2 for 32-bit configuration • 31:3 for 64-bit configuration • 31:4 for 128-bit configuration Value After Reset: 0x0

DMA Channel i Recieve Descriptor Tail Pointer Register

The DMA Channel i Rx Descriptor Tail Pointer Points to an offset from the base and indicates the location of the last valid descriptor.

DMA_CHi_RXDESC_TAIL_POINTER (i=0-3)

DMA Channel i Recieve Descriptor Tail Pointer Register(1128_H+i*80_H) Application Reset Value: 0000 0000_H



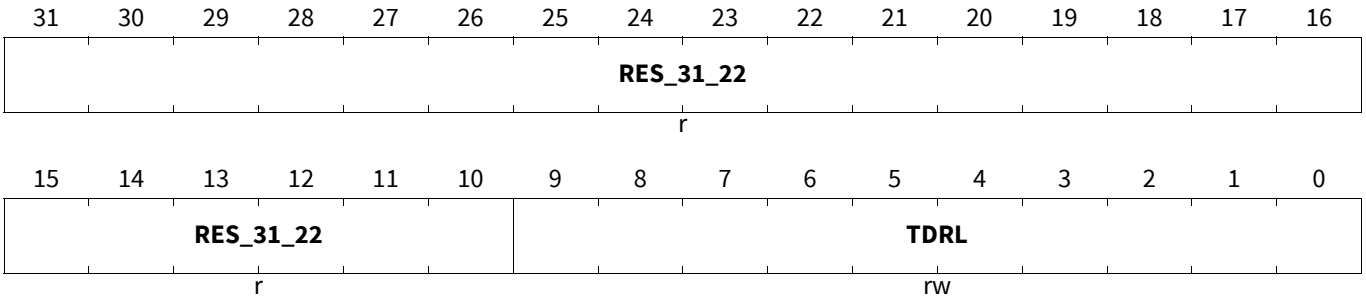
Field	Bits	Type	Description
RES_LSb	1:0	r	**Reserved** Value After Reset: 0x0
RDTP	31:2	rw	**Receive Descriptor Tail Pointer** This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers. The width of this field depends on the configuration: • 31:2 for 32-bit configuration • 31:3 for 64-bit configuration • 31:4 for 128-bit configuration Value After Reset: 0x0

DMA Channel i Transmit Descriptor Ring Length Register

The Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

DMA_CHi_TXDESC_RING_LENGTH (i=0-3)

DMA Channel i Transmit Descriptor Ring Length Register($112C_H$+i*80_H)Application Reset Value: 0000 0000_H



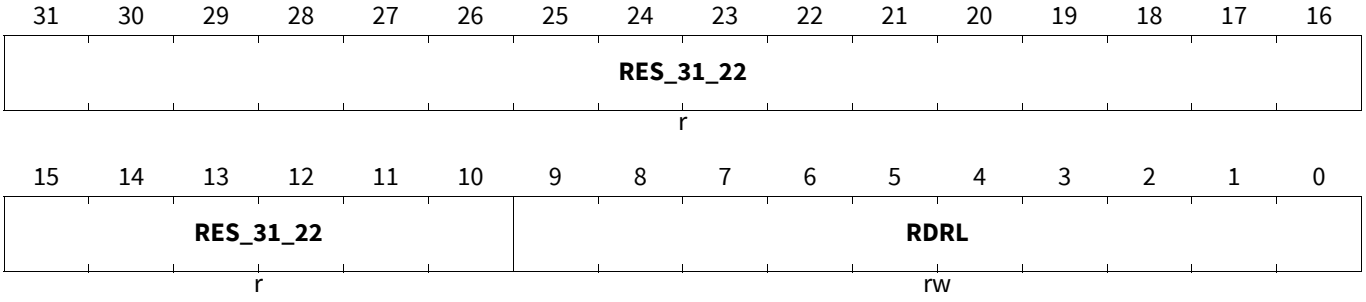
Field	Bits	Type	Description
TDRL	9:0	rw	**Transmit Descriptor Ring Length** This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. Synopsys recommends a minimum ring descriptor length of 4. For example, You can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9. Value After Reset: 0x0
RES_31_22	31:10	r	**Reserved** Value After Reset: 0x0

DMA Channel i Recieve Descriptor Ring Length Register

The DMA Channel i Rx Descriptor Ring Length register contains the length of the Receive descriptor circular ring.

DMA_CHi_RXDESC_RING_LENGTH (i=0-3)

DMA Channel i Recieve Descriptor Ring Length Register(1130_H+i*80_H) Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
RDRL	9:0	rw	**Receive Descriptor Ring Length** This register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. For example, You can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9. Value After Reset: 0x0
RES_31_22	31:10	r	**Reserved** Value After Reset: 0x0

DMA Channel i Interrupt Enable Register

The DMA Channel i Interrupt Enable register enables the interrupts reported by the Status register.

DMA_CHi_INTERRUPT_ENABLE (i=0-3)

DMA Channel i Interrupt Enable Register **(1134_H+i*80_H)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_16															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	RES_5_3			TBUE	TXSE	TIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r			rw	rw	rw

Field	Bits	Type	Description
TIE	0	rw	**Transmit Interrupt Enable** When this bit is set along with the NIE bit, the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled. Value After Reset: 0x0
TXSE	1	rw	**Transmit Stopped Enable** When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled. Value After Reset: 0x0
TBUE	2	rw	**Transmit Buffer Unavailable Enable** When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled. Value After Reset: 0x0
RES_5_3	5:3	r	**Reserved** Value After Reset: 0x0
RIE	6	rw	**Receive Interrupt Enable** When this bit is set along with the NIE bit, the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled. Value After Reset: 0x0
RBUE	7	rw	**Receive Buffer Unavailable Enable** When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled. Value After Reset: 0x0
RSE	8	rw	**Receive Stopped Enable** When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled. Value After Reset: 0x0

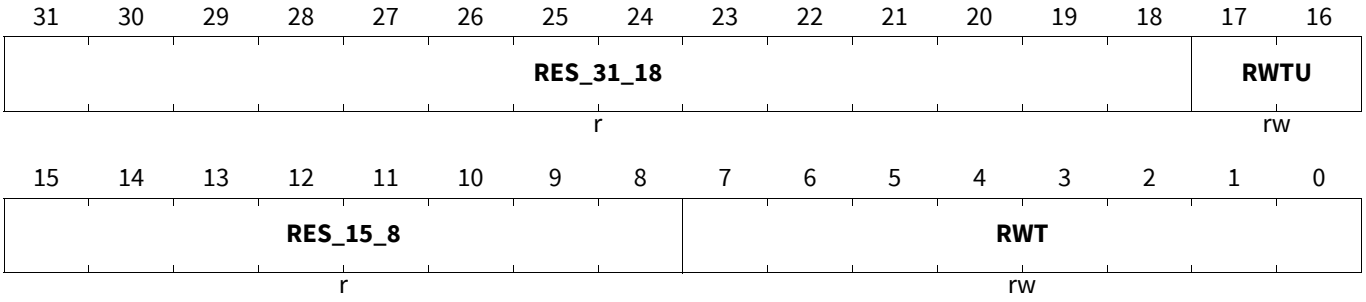
Field	Bits	Type	Description
RWTE	9	rw	**Receive Watchdog Timeout Enable** When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled. Value After Reset: 0x0
ETIE	10	rw	**Early Transmit Interrupt Enable** When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled. Value After Reset: 0x0
ERIE	11	rw	**Early Receive Interrupt Enable** When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled. Value After Reset: 0x0
FBEE	12	rw	**Fatal Bus Error Enable** When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled. Value After Reset: 0x0
CDEE	13	rw	**Context Descriptor Error Enable** When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled. Value After Reset: 0x0
AIE	14	rw	**Abnormal Interrupt Summary Enable** When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: • Bit 1: Transmit Process Stopped • Bit 7: Rx Buffer Unavailable • Bit 8: Receive Process Stopped • Bit 9: Receive Watchdog Timeout • Bit 10: Early Transmit Interrupt • Bit 12: Fatal Bus Error • Bit 13: Context Descriptor Error When this bit is reset, the abnormal interrupt summary is disabled. Value After Reset: 0x0
NIE	15	rw	**Normal Interrupt Summary Enable** When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the DMA_CH0_Status register: • Bit 0: Transmit Interrupt • Bit 2: Transmit Buffer Unavailable • Bit 6: Receive Interrupt • Bit 11: Early Receive Interrupt When this bit is reset, the normal interrupt summary is disabled. Value After Reset: 0x0

Field	Bits	Type	Description
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

DMA Channel i Recieve Interrupt Watchdog Timer Register

The Receive Interrupt Watchdog Timer register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the DMA_CHi_Status register.

DMA_CHi_RX_INTERRUPT_WATCHDOG_TIMER (i=0-3)
DMA Channel i Recieve Interrupt Watchdog Timer Register(1138_H+i*80_H) Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
RWT	7:0	rw	**Receive Interrupt Watchdog Timer Count** This field indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set. The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the DMA_CH0_Status register, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet. Value After Reset: 0x0
RES_15_8	15:8	r	**Reserved** Value After Reset: 0x0
RWTU	17:16	rw	**Receive Interrupt Watchdog Timer Count Units** This fields indicates the number of system clock cycles corresponding to one unit in RWT field. For example, when RWT=2 and RWTU=1, the watchdog timer is set for 2*512=1024 system clock cycles. Value After Reset: 0x0 00_B 256 01_B 512 10_B 1024 11_B 2048

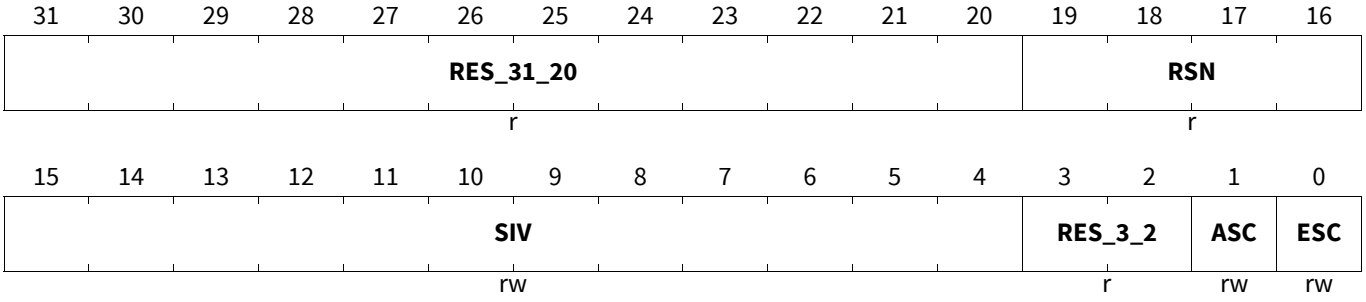
Field	Bits	Type	Description
RES_31_18	31:18	r	**Reserved** Value After Reset: 0x0

DMA Channel i Slot Function Control and Status Register

The Slot Function Control and Status register contains the control bits for slot function and the status for Transmit path.

DMA_CHi_SLOT_FUNCTION_CONTROL_STATUS (i=0-3)

DMA Channel i Slot Function Control and Status Register($113C_H$+i*80_H) **Application Reset Value: 0000 $07C0_H$**



Field	Bits	Type	Description
ESC	0	rw	**Enable Slot Comparison** When set, this bit enables the checking of the slot numbers programmed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is • equal to the reference slot number or • ahead of the reference slot number by one slot When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed. Value After Reset: 0x0
ASC	1	rw	**Advance Slot Check** When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the Tx descriptor is • equal to the reference slot number given in the RSN field or • ahead of the reference slot number by up to two slots This bit is applicable only when the ESC bit is set. Value After Reset: 0x0
RES_3_2	3:2	r	**Reserved** Value After Reset: 0x0

Field	Bits	Type	Description
SIV	15:4	rw	**Slot Interval Value** This field controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 us while the maximum value 4095 specifies the slot interval of 4096us. The default/ reset value is 0x07C which corresponds to slot interval of 125us. Value After Reset: 0x7c
RSN	19:16	r	**Reference Slot Number** This field gives the current value of the reference slot number in the DMA. It is used for slot comparison. Value After Reset: 0x0
RES_31_20	31:20	r	**Reserved** Value After Reset: 0x0

DMA Channel i Current Application Transmit Descriptor Register

The DMA Channel i Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

DMA_CHi_CURRENT_APP_TXDESC (i=0-3)
DMA Channel i Current Application Transmit Descriptor Register(1144_H+i*80_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTDESAPTR															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTDESAPTR															
r															

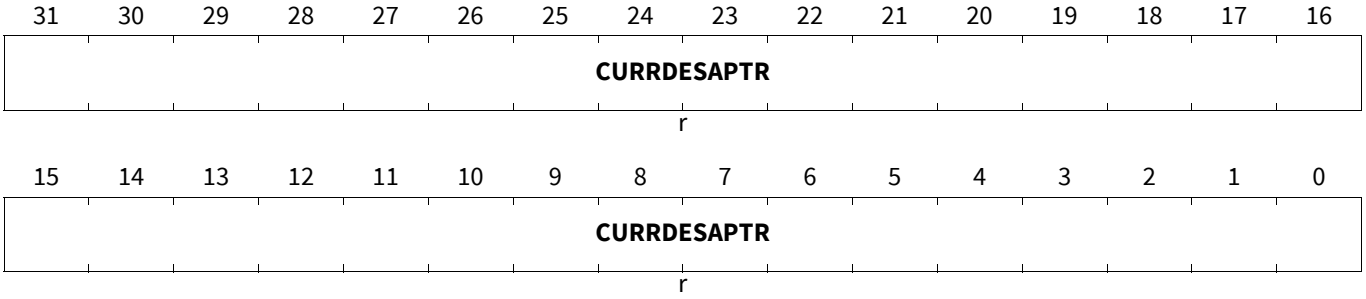
Field	Bits	Type	Description
CURTDESAPTR	31:0	r	**Application Transmit Descriptor Address Pointer** The DMA updates this pointer during Tx operation. This pointer is cleared on reset. Value After Reset: 0x0

DMA Channel i Current Application Receive Descriptor Register

The DMA Channel i Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

DMA_CHi_CURRENT_APP_RXDESC (i=0-3)

DMA Channel i Current Application Receive Descriptor Register($114C_H$+i*80_H) Application Reset Value: 0000 0000_H



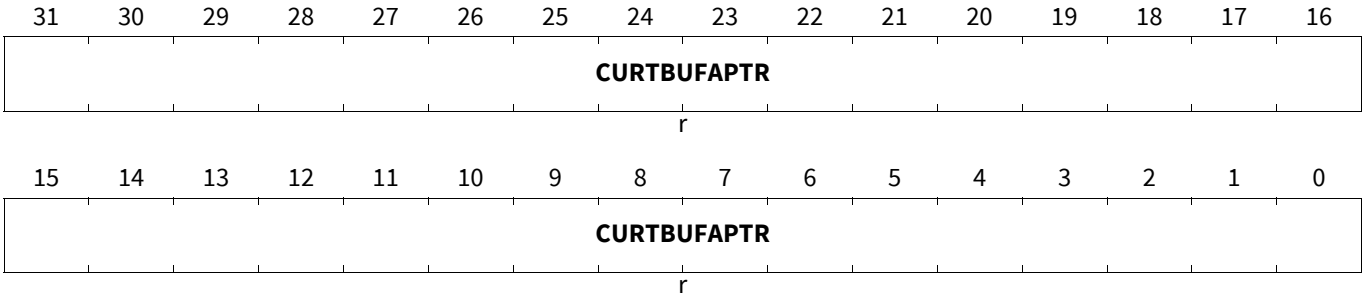
Field	Bits	Type	Description
CURRDESAPTR	31:0	r	**Application Receive Descriptor Address Pointer** The DMA updates this pointer during Rx operation. This pointer is cleared on reset. Value After Reset: 0x0

DMA Channel i Current Application Transmit Buffer Address Register

The DMA Channel i Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

DMA_CHi_CURRENT_APP_TXBUFFER (i=0-3)

DMA Channel i Current Application Transmit Buffer Address Register(1154_H+i*80_H) Application Reset Value: 0000 0000_H



Field	Bits	Type	Description
CURTBUFAPTR	31:0	r	**Application Transmit Buffer Address Pointer** The DMA updates this pointer during Tx operation. This pointer is cleared on reset. Value After Reset: 0x0

DMA Channel i Current Application Receive Buffer Address Register

The Channel 0 Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

DMA_CHi_CURRENT_APP_RXBUFFER (i=0-3)

DMA Channel i Current Application Receive Buffer Address Register($115C_H$+i*80_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRBUFAPTR															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRBUFAPTR															
r															

Field	Bits	Type	Description
CURRBUFAPTR	31:0	r	**Application Receive Buffer Address Pointer** The DMA updates this pointer during Rx operation. This pointer is cleared on reset. Value After Reset: 0x0

DMA Channel i Status Register

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

DMA_CHi_STATUS (i=0-3)

DMA Channel i Status Register (1160_H+i*80_H) Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES_31_22										REB			TEB		
r										r			r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	RES_5_3			TBU	TPS	TI
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r			rw	rw	rw

Field	Bits	Type	Description
TI	0	rw	**Transmit Interrupt** This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
TPS	1	rw	**Transmit Process Stopped** This bit is set when the transmission is stopped. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0

Field	Bits	Type	Description
TBU	2	rw	**Transmit Buffer Unavailable** This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPS0 field of the DMA_Debug_Status0 register explains the Transmit Process state transitions. To resume processing the Transmit descriptors, the application should do the following: 1. Change the ownership of the descriptor by setting Bit 31 of TDES3. 2. Issue a Transmit Poll Demand command. For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
RES_5_3	5:3	r	**Reserved** Value After Reset: 0x0
RI	6	rw	**Receive Interrupt** This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor. The reception remains in the Running state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
RBU	7	rw	**Receive Buffer Unavailable** This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
RPS	8	rw	**Receive Process Stopped** This bit is asserted when the Rx process enters the Stopped state. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
RWT	9	rw	**Receive Watchdog Timeout** This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received. Value After Reset: 0x0

Field	Bits	Type	Description
ETI	10	rw	**Early Transmit Interrupt** This bit indicates that the packet to be transmitted is fully transferred to the MTL Tx FIFO. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
ERI	11	rw	**Early Receive Interrupt** This bit indicates that the DMA filled the first data buffer of the packet. The RI bit of this register automatically clears this bit. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
FBE	12	rw	**Fatal Bus Error** This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
CDE	13	rw	**Context Descriptor Error** This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one's descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid.Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
AIS	14	rw	**Abnormal Interrupt Summary** Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the DMA_CH0_Interrupt_Enable register: • Bit 1: Transmit Process Stopped • Bit 7: Receive Buffer Unavailable • Bit 8: Receive Process Stopped • Bit 10: Early Transmit Interrupt • Bit 12: Fatal Bus Error • Bit 13: Context Descriptor Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0

Field	Bits	Type	Description
NIS	15	rw	**Normal Interrupt Summary** Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the DMA_CH0_Interrupt_Enable register: • Bit 0: Transmit Interrupt • Bit 2: Transmit Buffer Unavailable • Bit 6: Receive Interrupt • Bit 11: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in DMA_CH0_Interrupt_Enable register) affect the Normal Interrupt Summary bit. This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared. Access restriction applies. Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect. Value After Reset: 0x0
TEB	18:16	r	**Tx DMA Error Bits** This field indicates the type of error that caused a Bus Error. For example, error response on the AHB or AXI interface. Bit 18 1_B Error during data transfer by Tx DMA 0_B No Error during data transfer by Tx DMA Bit 17 1_B Error during descriptor access 0_B Error during data buffer access Bit 16 1_B Error during read transfer 0_B Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt. Value After Reset: 0x0
REB	21:19	r	**Rx DMA Error Bits** This field indicates the type of error that caused a Bus Error. For example, error response on the AHB or AXI interface. Bit 21 1_B Error during data transfer by Rx DMA 0_B No Error during data transfer by Rx DMA Bit 20 1_B Error during descriptor access 0_B Error during data buffer access Bit 19 1_B Error during read transfer 0_B Error during write transfer This field is valid only when the FBE bit is set. This field does not generate an interrupt. Value After Reset: 0x0

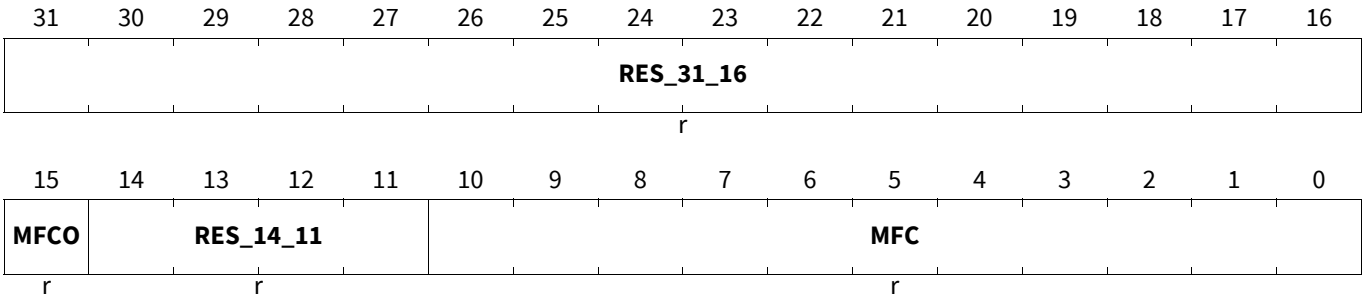
Field	Bits	Type	Description
RES_31_22	31:22	r	**Reserved** Value After Reset: 0x0

DMA Channel i Missed Frames Count Register

This register has the number of packet counter that got dropped by the DMA either due to Bus Error or due to programming RPF field in DMA_CHx_RX_CONTROL register.

DMA_CHi_MISS_FRAME_CNT (i=0-3)

DMA Channel i Missed Frames Count Register(1164_H+i*80_H) **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
MFC	10:0	r	**Dropped Packet Counters** This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in DMA_CHx_RX_CONTROL register. The counter gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_14_11	14:11	r	**Reserved** Value After Reset: 0x0
MFCO	15	r	**Overflow status of the MFC Counter** When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read. Access restriction applies. Clears on read. Self-set to 1 on internal event. Value After Reset: 0x0
RES_31_16	31:16	r	**Reserved** Value After Reset: 0x0

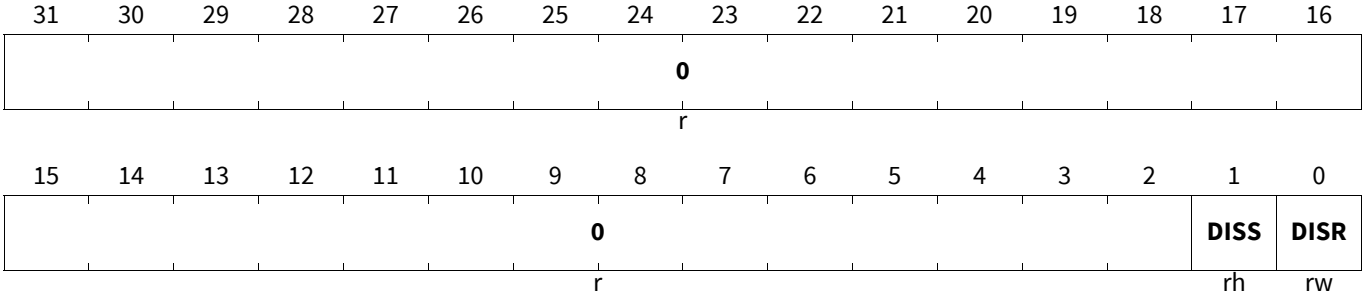
Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock for the module.

Note: After a power on or application reset, the input clocks are switched off inside the module and the GETHmodule is disabled (DISS set).

CLC

Clock Control Register **(2000_H)** **Application Reset Value: $0000\ 0003_H$**



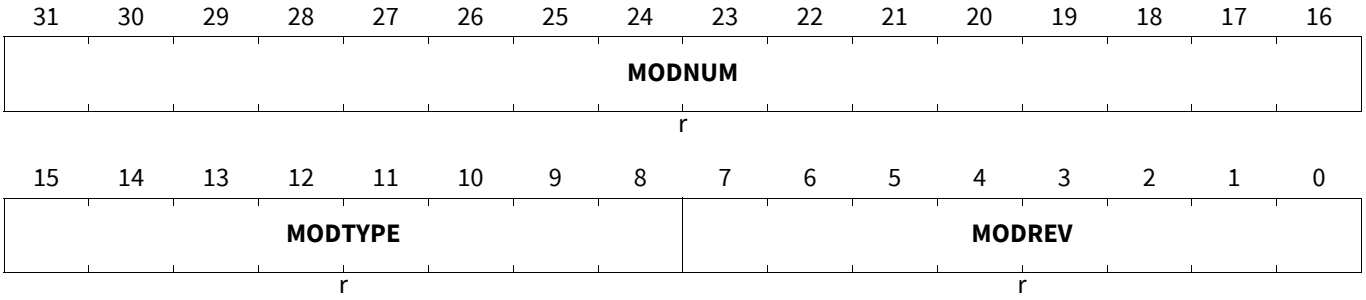
Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. The disable request is delayed internally until all pending transactions on the master (SRI) and slave (SPB) interface are completed.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Module Identification Register

The GETHModule Identification Register ID contains read-only information about the module version.

ID

Module Identification Register **(2004_H)** **Application Reset Value: $00E9\ C0XX_H$**



Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01_H (first revision). To be incremented eg. 2 for GMAC QoS
MODTYPE	15:8	r	**Module Type** This bit field defines the module as a 32-bit module: $C0_H$
MODNUM	31:16	r	**Module Number Value** This bit field defines the module identification number for the GETH: $00E9_H$

General Purpose Control Register

The Input and Output Control Register GPCTL determines which alternate input is to be used and selects the phy interface.

This register is not affected by the local module reset.

GPCTL

General Purpose Control Register (2008$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				**0**				**EPR**		**ALTI10**		**ALTI9**		**ALTI8**	
r				r				rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALTI7		**ALTI6**		**ALTI5**		**ALTI4**		**ALTI3**		**ALTI2**		**ALTI1**		**ALTI0**	
rw		rw		rw		rw		rw		rw		rw		rw	

Field	Bits	Type	Description
ALTIy (y=0-10)	2*y+1:2*y	rw	**Alternate Input Select** Selects the alternate input for channel y. If no signal is connected to an alternate input, it is connected to GND. Map of signals demultiplexed by ALTI bit fields: ALTI0: MDIO ALTI1: RXCLK, REFCLK ALTI2: CRS ALTI3: COL ALTI4: RXDV, CRSDV, RCTL ALTI5: RXER ALTI6: RXD0 ALTI7: RXD1 ALTI8: RXD2 ALTI9: RXD3 ALTI10: TXCLK 00_B Alternate Input 0 selected 01_B Alternate Input 1 selected 10_B Alternate Input 2 selected 11_B Alternate Input 3 selected

Field	Bits	Type	Description
EPR	24:22	rw	**External Phy Interface RMII Mode Bit** Used to select the phy interface during module reset of MAC triggered by setting SWR bit of DMA_Mode Register. Note that the status of this bit is latched in during module reset only. 000_B MII is selected and the divider (see 100B)is not active but the clock signals clk_tx_i and clk_rx_i are connected directly to the referring port pins 001_B RGMII is selected. The clock signals are connected directly to the referring port pins 010_B do not use! 011_B do not use! 100_B RMII is selected and an internal predivider divides the clock clk_rmii_i from pin ETHRXC by 2 or 20 and provides it to the internal signals clk_tx_i and clk_rx_i. The division factor depends from the signal mac_speed_o, controlled by bit FES in Register 0 (MAC Configuration Register) 101_B do not use! … 111_B do not use!
0	30:25, 31	r	**Reserved** Read as 0; should be written with 0.

Access Enable Register 0

The Access Enable Register 0 controls write access[1)] for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, … , EN31 -> TAG ID 011111B.

Attention: The DMA control registers are NOT protected by ACCEN0! They have separate (GETH_ACCEN0Dx) registers. See below.

ACCEN0
Access Enable Register 0 **($200C_H$)** **Application Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

The Access Enable Register 1 controls write access1) for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the device.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, … , EN31 -> TAG ID 111111B.

ACCEN1

Access Enable Register 1 **(2010_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

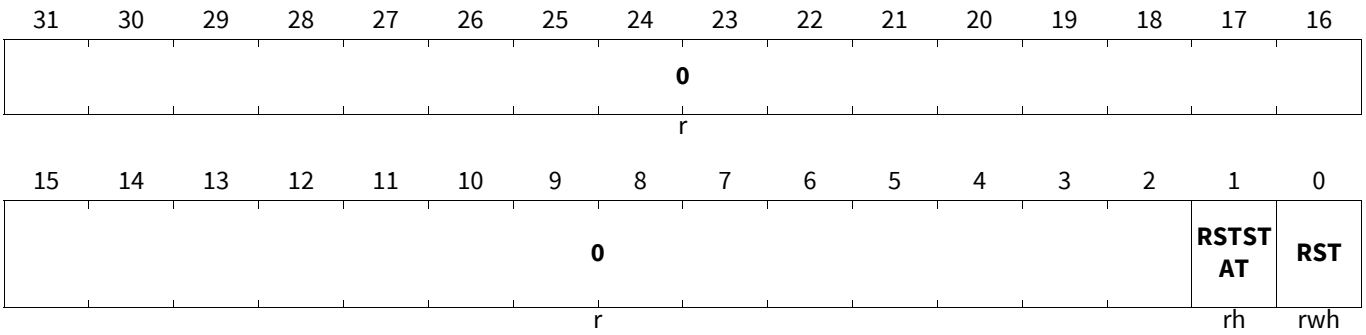
Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
Kernel Reset Register 0 **(2014_H)** **Application Reset Value: 0000 0000_H**

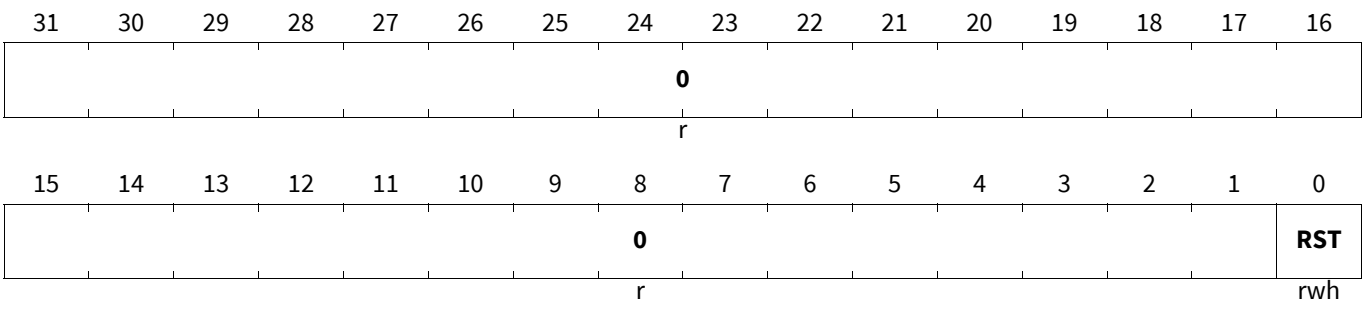


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set and both: master and slave interface are idle. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **(2018_H)** **Application Reset Value: 0000 0000_H**

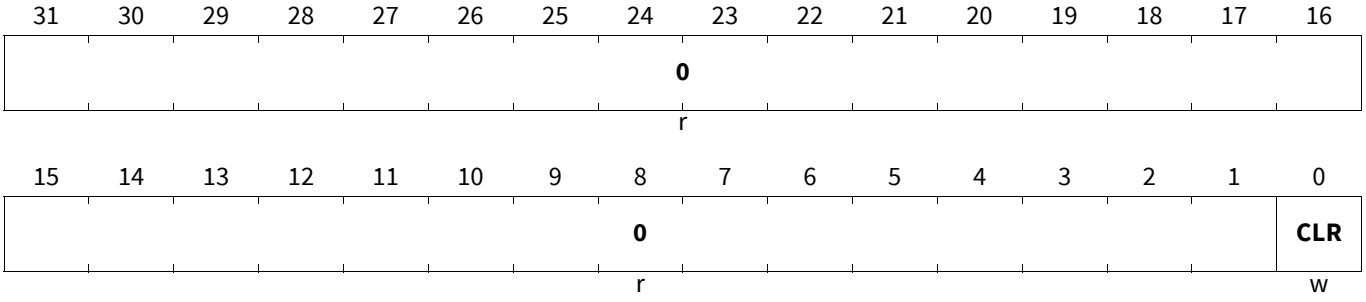


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set and both: master and slave interface are idle. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR
Kernel Reset Status Clear Register **($201C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Access Enable Register 0 for DMAx

The Access Enable Register 0 for DMA x controls write access[1)] for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0Dx / ACCEN1Dx are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0Dx.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, … , EN31 -> TAG ID 011111B.

1) The BPI_FPI Access Enable functionality controls only write transactions to the registers needed to configure DMAx. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in these registers

These registers (0 …x) protect the address space (size: 80H for each DMA) occupied by the control registers that are available for each DMA. E.g. DMA_CHx_Control, DMA_CH0_Tx_Control, DMA_CH0_Rx_Control, DMA_CH0_TxDesc_List_HAddress … and so on. (Offset 0x1100H + x * 80H … 0x117CH + x * 80H)

ACCEN0Dx (x=0-3)

Access Enable Register 0 for DMAx **(2020_H+x*8)** **Application Reset Value: FFFF $FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1 for DMAx

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1Dx is not implemented with register bits as the related TAG IDs are not used in the device.

Mapping of TAG IDs to ACCEN1Dx.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, … , EN31 -> TAG ID 111111B.

ACCEN1Dx (x=0-3)

Access Enable Register 1 for DMAx **(2024_H+x*8)** **Application Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

Skew Control Register

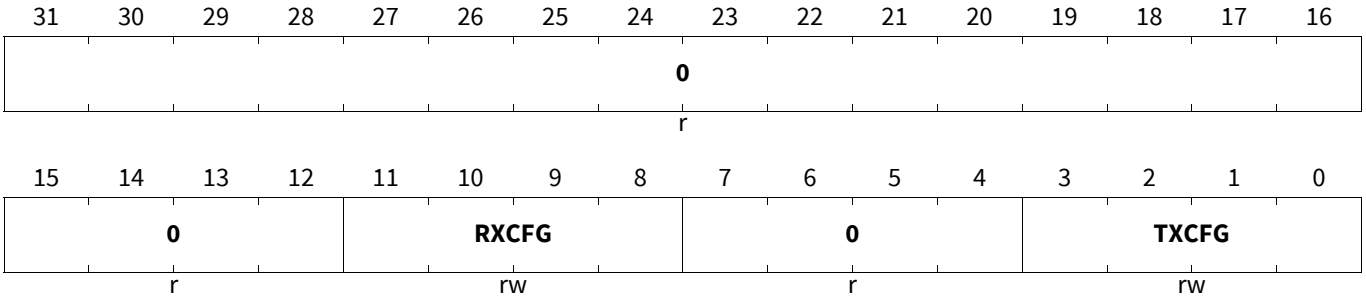
This register controlls the operation of the delay block.

This register is not affected by the local module reset

NOTE: Intended for configuration phase only. Do not change during frame transmission! Data might be lost.

SKEWCTL

Skew Control Register **(2040_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TXCFG	3:0	rw	**TX Clock delay control for RGMII Mode** Used to specify the receive timing skew in RGMII mode using integrated delay generation on TXCLK. In MII and RMII mode, this is not active. Skew is (TXCFG * 222,22) ps. If cleared, skew is 0 (default, skew generation switched off).
RXCFG	11:8	rw	**RX Clock delay control for RGMII Mode** Used to specify the receive timing skew in RGMII mode using integrated delay generation on RXCLK. In MII and RMII mode, this is not active. Skew is (RXCFG * 222,22) ps. If cleared, skew is 0 (default, skew generation switched off).
0	7:4, 31:12	r	**Reserved** Read as 0; should be written with 0.

44.6 IO Interfaces

The GETH is connected to general purpose I/O lines. Each line is connected to up to four different general purpose I/O lines that can be chosen alternatively. The selection from one of these alternatives is done by application SW by programming GETH_GPCTL.ALTIx respectively.

For MDIO, ALTIx selects the direction control signal (“HW_OUT control”) gmii_mdo_o_e_[x] as well. The signals gmii_mdo_o_e_[n] that are not selected are forced passive.

The following port control operations and selections must be executed for these I/O lines:

- GETH_GPCTL.ALTIx to select alternate inputs. If RGMII is selcted ALTI is not needed to be programmed. The input ports are selected automatically. For RGMII there is only one alternative input per signal available.
- Pad driver characteristics selection for the outputs (Port PDR registers)
- Port Pin Controller Select Register PCSR (Set bit SELx for TXER, TXCLK, TXD[3:0], TXEN, TCTL if used for GETH)
- Input/output function selection (Port IOCR registers)

Some PHYs provide an interrupt output, signalling e.g. line activity. It might be useful to connect this to an interrupt input on system level. This signal is not connected to the GETH itself!

If for one product no signal is connected to an alternate input, it is connected to GND internally at module entity level. This allows to leave some signals unconnected in the application (i.e. RXER, CRS, COL) and save pins and external connection to GND.

Table 503 List of GETH Interface Signals

Interface Signals	I/O	Description
TRIGO(9:0)	out	**Ethernet TRIGO Service Request**
TXCLKA	in	**Transmit Clock Input for MII**
TXCLKB		
TXCLKC		
TXCLKD		
REFCLKA	in	**Reference Clock input for RMII (50 MHz)**
REFCLK(3:1)		
MDC	out	**MDIO clock**
MDIOA	in	**MDIO Input**
MDIOB		
MDIOC		
MDIOD		
MDIO	out	**MDIO Output**
TXEN	out	**Transmit Enable MII and RMII**
TXER	out	**Transmit Error MII**
TXD(3:0)	out	**Transmit Data**
CRSA	in	**Carrier Sense MII**
CRSB		
COLA	in	**Collision MII**
COLB		
COLC		
COLD		
RXDVA	in	**Receive Data Valid MII**
RXDVB		
RXDVC		
RXDVD		
RXERA	in	**Receive Error MII**
RXERB		
RXERC		
RXERD		
RXD0A	in	**Receive Data 0 MII, RMII and RGMII (RGMII can use RXD0A only)**
RXD0B		
RXD0C		
RXD0D		

Table 503 List of GETH Interface Signals (cont'd)

Interface Signals	I/O	Description
RXD1A	in	**Receive Data 1 MII, RMII and RGMII (RGMII can use RXD1A only)**
RXD1B		
RXD1C		
RXD1D		
RXD2A	in	**Receive Data 2 MII and RGMII (RGMII can use RXD2A only)**
RXD2B		
RXD2C		
RXD2D		
RXD3A	in	**Receive Data 3 MII and RGMII (RGMII can use RXD3A only)**
RXD3B		
RXD3C		
RXD3D		
PPS	out	**Pulse Per Second**
RXCLKA	in	**Receive Clock MII and RGMII**
RXCLKB		
RXCLKC		
RXCLKD		
TCTL	out	**Transmit Control for RGMII**
CRSDVA	in	**Carrier Sense / Data Valid combi-signal for RMII**
CRSDVB		
CRSDVC		
CRSDVD		
RCTLA	in	**Receive Control for RGMII**
TXCLK	out	**Transmit Clock Output for MII and RGMII**
GREFCLK	in	**Gigabit Reference Clock input for RGMII (125 MHz high precission)**

44.7 Revision History

Table 504 Revision History

Reference	Change to Previous Version	Comment
V1.3.10		
Page 415	Previous versions removed from revision history.	
V1.3.11		
–	No functional change.	–
V1.3.12		
Page 414	Updated transmit/receive for MII and RGMII in **IO Interfaces**.	
Page 415	Updated Revision History.	
V1.3.13		

Table 504 Revision History (cont'd)

Reference	Change to Previous Version	Comment
Page 16	Corrected: Bit 10 changed to Bit 1.	
Page 413	Corrected: 'receive timing skew' changed to 'transmit timing skew' in SKEWCTL.TXCFG.	
Page 6	Removed 'Optional' two times in feature list.	
Page 143	Removed last sentence of Note (set to internal only).	
Page 143	Explanation of write delay in Note rewritten (changed from 4 to 6 cycles).	
Page 166	Explanation of write delay in last paragraph of general notes rewritten (changed from 4 to 6 cycles).	
Page 220	Changed long description for MAC_RWK_PACKET_FILTER (explanation of write delay, changed from 4 to 6 cycles).	
V1.3.14		
Page 51, Page 84, Page 85, Page 90, Page 92, Page 146, Page 179	Removed information regarding MAC_RxQ_Ctrl3 register.	
Page 92	Corrected wrong equation and updated bullet list.	
Page 128	Corrected missing sentence part.	
Page 11, Page 108	Added notes.	
Page 6	Updated bullet list items.	
Page 55, Page 68, Page 71, Page 77, Page 79, Page 84, Page 105	Removed duplicate sentences.	
Page 49, Page 55, Page 71, Page 77, Page 98, Page 102, Page 104, Page 107, Page 152	Optimized numbering of notes.	
Page 71, Page 72, Page 68	Added missing "GETH" and "internal reference time".	
Page 179	Optimized MAC_RWK_PACKET_FILTER long description.	
Page 145	Replaced Tx by Rx register.	
V1.3.15		
Page 92	Equation reduced and datawidth(32) and N(7) removed.	
Page 2	Formal update of section "Delta to AURIX™ TC2xx".	

45 SRI External Bus Unit (EBU)

The External Bus Unit (EBU), controls the transactions between external memories or peripheral units, and the internal memories and peripheral units. The basic interfaces of the EBU are shown in **Figure 724**.

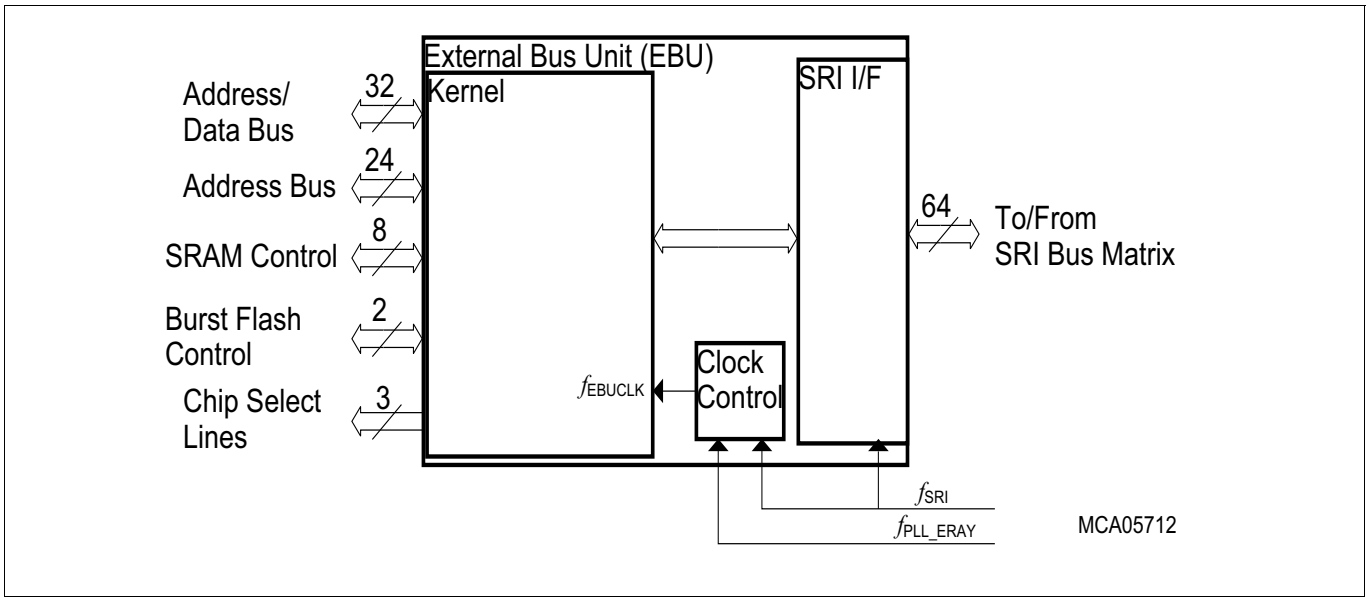


Figure 724 EBU Interface Diagram

The EBU is internally connected to the SRI by a single slave interface.

45.1 Feature List

Features supported in the Memory Controller:-

- Highly programmable access parameters.
- Intel-style peripheral/device support.
- Burst FLASH support (see **Section 45.3.14** for specific device types).
- Synchronous SRAM support (see **Section 45.3.14** for specific device types).
- NAND flash
- Multiplexed access (address & data on the same bus)
- Data Buffering: Two read buffers. Single write buffer in the SRI interface.
- Three programmable address regions.
- External bus frequency: Module frequency:flash clock = 1:1, 1:2, 1:3 or 1:4.

45.2 Overview

The Memory Controller module for SRI-based systems connects on-chip controller cores (e.g. Tricore CPU, DMA Controller) to external resources such as memories and peripherals. **Figure 725** shows Memory Controller within a typical system.

Any SRI master can (in conjunction with an SRI Matrix) access external memories through the Memory Controller.

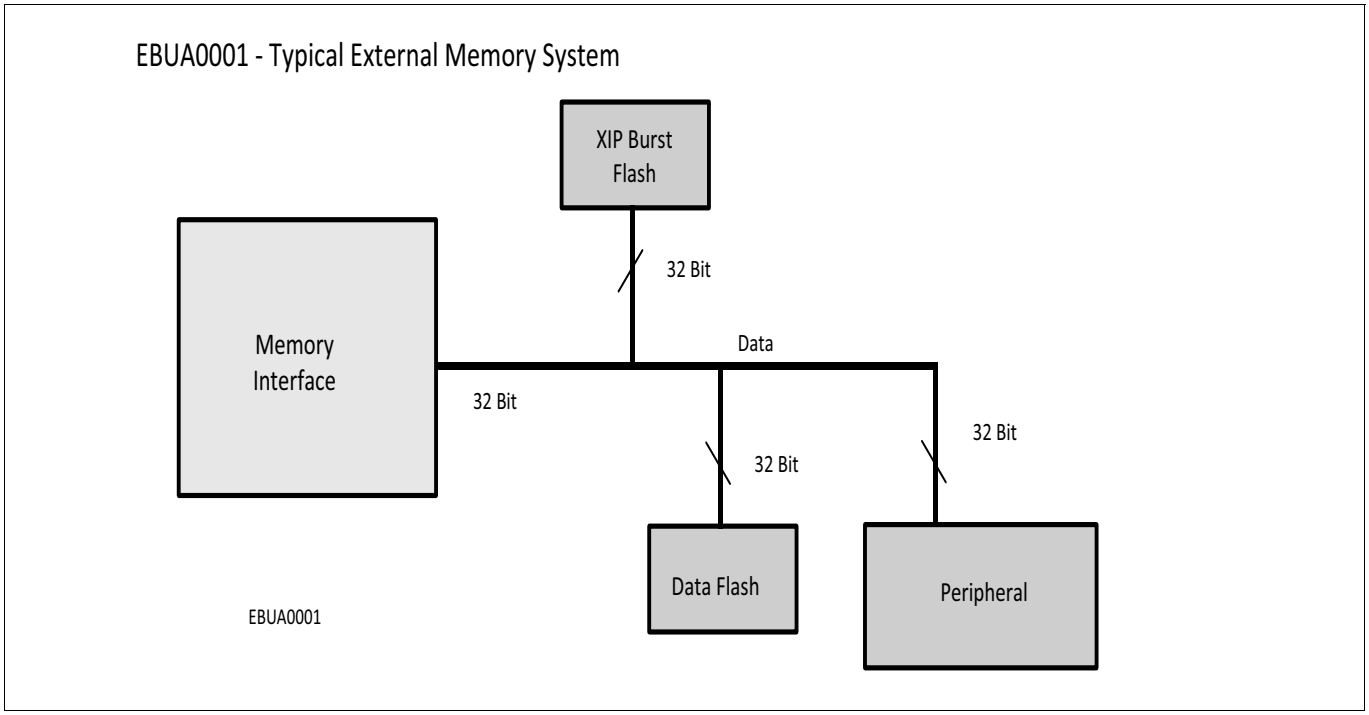


Figure 725 Typical External Memory System

45.3 Functional Description

45.3.1 References

The EBU complies with the requirements of Open NAND Flash Interface Specification (Revision 2.0, 27-February-2008) when generating accesses to external memories.

45.3.2 Product Specific Core Customisation

A pin multiplexing scheme has been implemented to allow the use of low power, 5 volt compatible pads for the 32 bit databus. This has the effect of imposing the following limitations on EBU configuration

- Some programming of the Ports logic is needed to ensure that the correct pads are available for EBU to operate
- SDRAM is not supported
- Use of byte control lines limits the available external address bus to A(19:0).
- Region 2 cannot be used if external bus arbitration is enabled, External bus arbitration is not supported so no advantage is gained by enabling it.
- The use of slower pads for the main databus means that extra setup time may be needed for asynchronous writes and that the clock frequency for synchronous writes may have to be reduced.
- The combined CS signal described in **Section 45.3.10.7** is not available as a device I/O for the TC39x.

It is possible to programme the EBU ignoring these restrictions but doing so will result in unpredictable operation and possible lockups of the system.

45.3.3 Allocation of Unused Signals as GPIO

The EBU will allow pins not required for its programmed configuration to be allocated for use as GPIO.

For some signals, the EBU is unable to determine automatically whether or not the signals are required for a particular configuration. These signals can be manually made available for GPIO or alternate EBU functions by writing to the **USERCON** register.

Additionally some EBU signals are routed through the normal output multiplexer of the Ports logic. These signals have to be configured for EBU operation using the control registers in the relevant Ports instance.

45.3.4 Memory Controller Structure

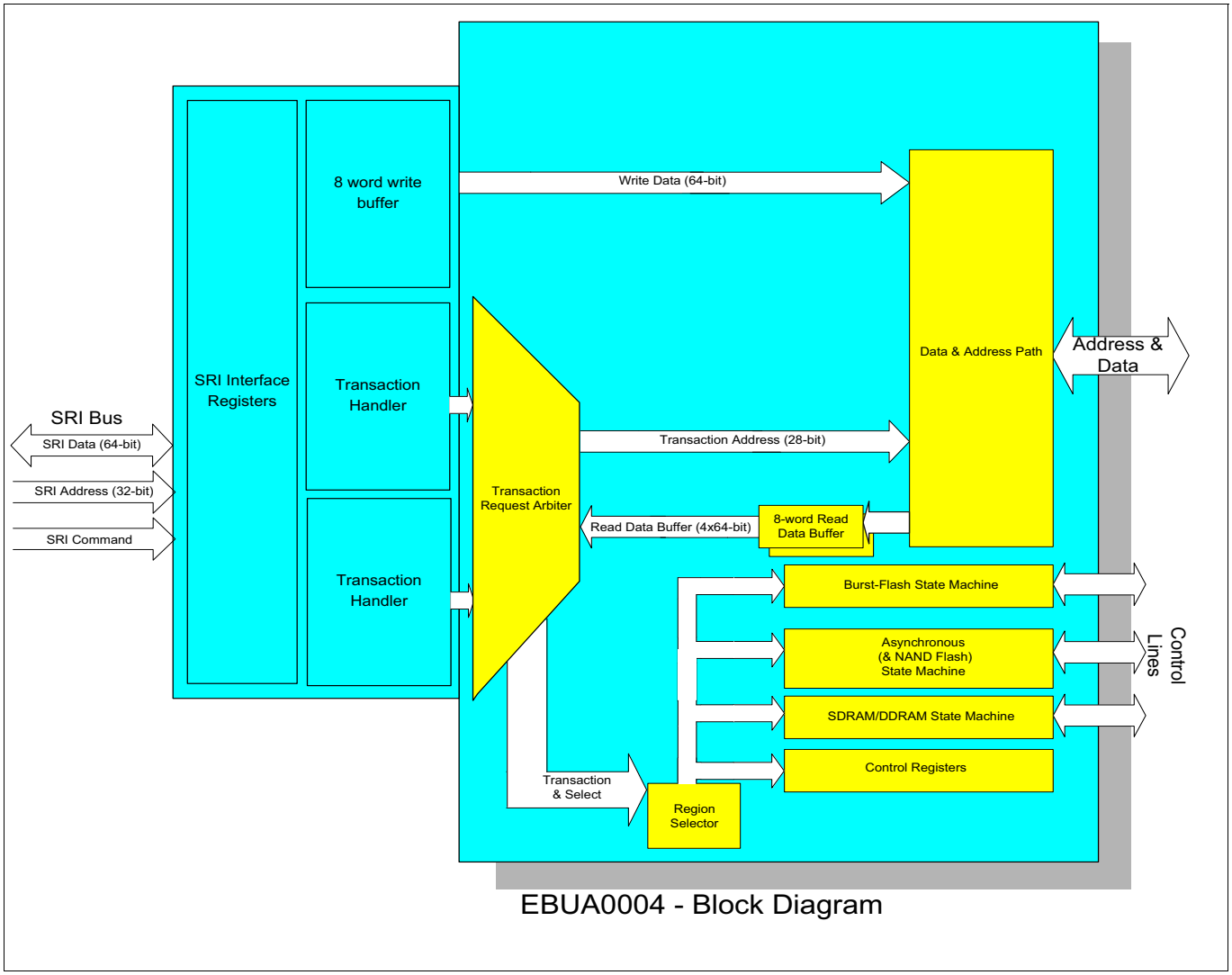


Figure 726 Memory Controller Block Diagram

The SRI interface translates SRI transactions from the ports into appropriate transaction requests which can be transferred to the arbiter. The interface can handle two transactions simultaneously as required by the SRI protocol which has a two deep, command pipeline. The interface will also generate an ECC code for all read data returned and will check ECC codes received along with each SRI transaction. In the event of a transaction failing the ECC check, an error will be flagged to the system for further processing and the transaction will be terminated with an SRI error.

The arbiter looks at the transaction requests and schedules corresponding requests to the relevant state machine. Only one state machine can be active at any time and the arbiter is designed to prevent out-of-order

execution. The dedicated state machines are used to sequence control signals and to co-ordinate accesses to each of the different external memory/device types and also the internal registers.

If two consecutive accesses are to the same state machine, the second access can be queued before the first has completed. This allows accesses to run “back-to-back” on the external bus and maximises bandwidth utilisation.

45.3.5 Memory Controller Read Architecture

The memory controller contains two, identical read buffers each with a capacity of 4, 64 bit words. A read access will be allowed to proceed if one of the buffers is flagged as available. The data read from the external memory will be stored in the read buffer and the outputs from the read buffer will be multiplexed to the SRI port. Once the SRI port signals that all data has been returned to the requesting SRI master, the read buffer will again be flagged as available.

This architecture allows reads to be in progress simultaneously, as a second read can be running while the first read is still waiting for data to be returned to the SRI master.

45.3.6 Access Arbitration

Arbitration of SRI accesses is handled by the SRI itself. The EBU will only ensure that accesses are processed in the order that they arrive at the SRI interface.

45.3.6.1 Programming Sequence Locking

Programming sequences for some Burst Flash devices require that the source of the programming transactions has exclusive access to the external memory device for the duration of the write sequence. If such devices are used and can be accessed by multiple transaction sources then the “Locked Programming Sequence” feature can be enabled via register bit EBU_BUSWCONx.LOCKCS.

The fail-safe timeout timer defaults to a count of 255 and is decremented at a frequency of SRI_CLK/16. It commences counting from the preload value every time a write completes. The default value can be changed by writing to the EBU_MODCON.LOCKTIMEOUT register field.

45.3.6.2 Access Enable

Write accesses from any specific bus master can be temporarily denied access to the EBU registers by setting the relevant bit in the ACCEN register. See the register description in **“Access Enable Registers, ACCEN0 and ACCEN1” on Page 85** for a full description of the operation of this feature.

45.3.7 Clocking Strategy and Local Clock Generation

The Memory Controller can be configured to operate from several possible clock sources. The clock generation logic is used to select between these clock sources and generate the internal clock used for the memory controller, EBU_CLK.

45.3.7.1 Clocking Modes

The bridges can be operated in one of two modes using one of two clocks:-

- Asynchronous mode: The SRI clock and Memory Controller internal clock (EBU_CLK) are assumed to be asynchronous. EBU_CLK is derived from the Flexray PLL output.
- Synchronous mode: The is derived from the controller (SRI) clock. The EBU_CLK and SRI interface clocks have aligned edges (although pulse swallowing can be used on the EBU_CLK clock, so that the EBU core can run at a lower frequency than the SRI bus matrix).

- Synchronous or Asynchronous mode at half of SRI clock: The EBU clock is running at half the frequency of the processor clock. The EBU clock is edge aligned with the processor and SRI interface clocks.

The clock for the SRI interface of the memory controller, is always be derived from the same synchronous source (the processor clock, .

The mode of operation of the bridge is controlled by the **CLC.SYNC** register field. The **CLC.SYNCACK** field will be updated to report the current operating mode.

The **CLC.SYNC** register field can be used (dynamically) to switch between these two modes.

If **CLC.DIV2** is set to 0_B, then **CLC.SYNC** also controls the clock input and switches it between the processor clock and the Flexray PLL output.

However, if **CLC.DIV2** is set to 1_B, then EBU clock source is forced to be half the frequency of the processor clock and **CLC.SYNC** serves no useful purpose. The value of **DIV2** in use by the memory controller is stored in the **CLC.DIV2ACK** field.

45.3.7.2 Standby Mode

The Memory Controller can be configured to disable its internal clock and enter standby mode by writing a logic '1' to the **EBU_CLC.DISR** register field.

45.3.7.3 External Bus Clock Generation

The memory controller can generate two external bus clocks.

One of these is intended for use with SDRAM type devices controlled by the SDRAM state machine[1)] and is output on SDCLKO.

The other is intended for use with other synchronous memories such as burst flash and is output on BFCLKO. See **"Burst Flash Clock" on Page 46**

Clock feedback is used to correct for pad output and PCB track delays and each clock output therefore has an equivalent clock input. These are SDCLKI and BFCLKI respectively.

The clocking mode will be determined by the **AGEN** fields of the configured memory regions.

Setting the **CLK_COMB** bit in the **MODCON** register will cause the same clock to be generated on both the SDCLKO and BFCLKO pins. It is recommended that only one region is configured for a continuously running clock if this option is selected and that all synchronous regions have the same clock divide ratio programmed.

45.3.8 External Bus Arbitration

External bus arbitration is provided to allow the EBU to share its external bus with other master devices. This capability allows other external master devices to obtain ownership of the external bus, and to use the bus to access external devices connected to this bus. The scheme provided by the EBU is compatible with other TriCore and XC2XXX devices and therefore allows the use of such devices as (external bus) masters together with the .

*Note: In this section, the term "external master" is used to denote a device which is located on the **external** bus and is capable of generating accesses across the external bus (i.e. is capable of driving the external bus). An external master is not able to access units that are located inside the .*

45.3.8.1 External Bus Modes

The EBU can be in one of two bus states on the external bus:

- Owner State
- Hold State

When in Owner State, the EBU operates as the master of the external bus. In other words, the EBU drives the external bus as required in order to access devices located on the external bus. While the EBU is in Owner State it is not possible for any other master to perform any accesses on the external bus. When the EBU is in Owner State, **MODCON.BUSSTATE** is set to 1_B.

In Hold State, the EBU tri-states the appropriate signal on the external bus in order to allow another external bus master to perform accesses on the external bus (i.e. to allow another master to drive the various external bus signals without contention with EBU). When the EBU is in Hold State, **MODCON.BUSSTATE** is set to 0_B.

45.3.8.2 Arbitration Signals and Parameters

The arbitration scheme consists of an external bus master that is responsible for controlling the allocation of the external bus. This master is referred to as the "Arbiter" within this document. The other external bus master

1) See **"Programmable Device Types" on Page 21** for the device types supported by the EBU and the state machines used to control them

(termed Participant within this document) requests ownership of the bus, and when necessary, from the Arbiter. The EBU can be programmed to operate either as an Arbiter or as a Participant (see **Page 8**). The following three lines are used by the EBU to arbitrate the external bus.

Table 505 EBU External Bus Arbitration Signals

Signal	Direction	Function
$\overline{\text{HOLD}}$	In	$\overline{\text{HOLD}}$ is asserted (low) by an external bus master when the external bus master requests to obtain ownership of the external bus from the EBU.
$\overline{\text{HLDA}}$	In/Out[1)]	$\overline{\text{HLDA}}$ is asserted (low) by the Arbiter to signal that the external bus is available for use by the Participant (i.e. the bus is not being used by the Arbiter). $\overline{\text{HLDA}}$ is sampled by the Participant to detect when it may use the external bus.
$\overline{\text{BREQ}}$	Out	$\overline{\text{BREQ}}$ is asserted (low) by the EBU when the EBU requests to obtain ownership of the external bus.

1) The direction of this signal depends upon the mode in which the EBU is operating (see **Page 8**).

Two components that are equipped with the EBU arbitration protocol can be directly connected together (without additional external logic) as shown below:

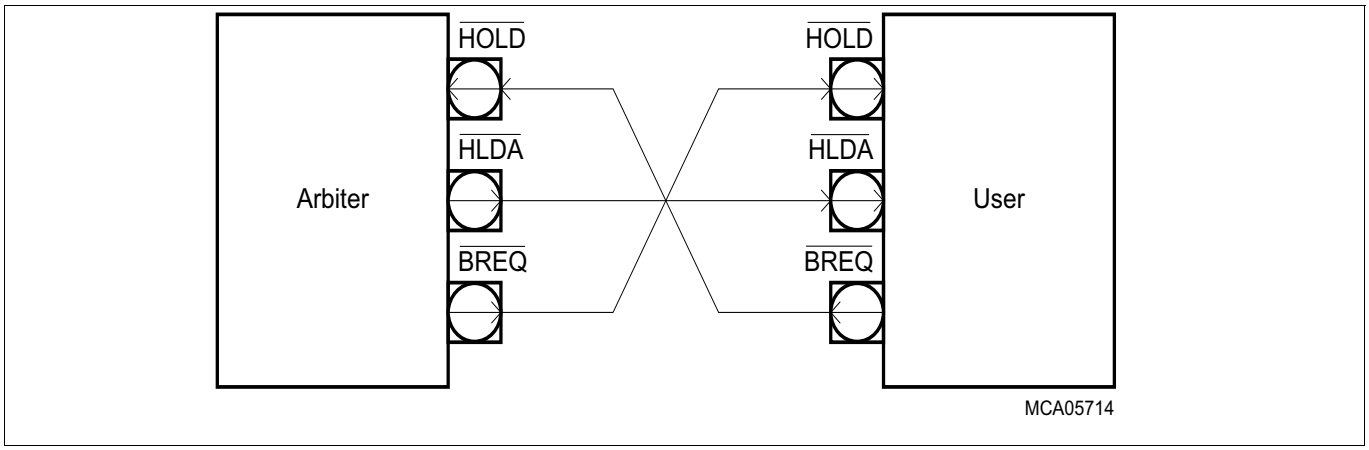


Figure 727 Connection of Bus Arbitration Signals

*Note: In the example of **Figure 727**, it is possible for the EBU to perform the function or either Arbiter or Participant (or indeed both the Arbiter and Participant may be the EBU).*

Table 506 lists the programmable parameters for the external bus arbitration.

Table 506 External Bus Arbitration Programmable Parameters

Parameter	Function	Description See
EBU_MODCON.ARBMODE	Arbitration mode selection	**Page 8**
EBU_MODCON.ARBSYNC	Arbitration input signal sampling control	
EBU_MODCON.EXTLOCK	External bus ownership locking control	
EBU_MODCON.TIMEOUTC	External bus time-out control	

45.3.8.3 Arbitration Modes

The arbitration mode of the EBU can be selected through configuration pins during reset or by programming the EBU_MODCON.ARBMODE bit field (see **Page 8**) after reset. Four different modes are available:

- No Bus Mode
- Sole Master Mode
- Arbiter Mode
- Participant Mode

45.3.8.3.1 No Bus Arbitration Mode

All accesses of the EBU to devices on the external bus are prohibited and will generate an SRI bus error. The EBU is in hold state all the time. The $\overline{\text{HOLD}}$ and $\overline{\text{HLDA}}$ arbitration inputs are ignored and $\overline{\text{BREQ}}$ arbitration output remains at high (inactive) level.

No Bus Mode is selected by EBU_MODCON.ARBMODE = 00_B.

45.3.8.3.2 Sole Master Arbitration Mode

The EBU is the only master on the external bus; therefore no arbitration is necessary and the EBU has access to the external bus at any time. The EBU is in owner state all the time. The $\overline{\text{HOLD}}$ arbitration input is ignored, and the $\overline{\text{HLDA}}$ and $\overline{\text{BREQ}}$ arbitration outputs remain at high (inactive) level.

Sole Master Mode is selected by EBU_MODCON.ARBMODE = 11_B.

45.3.8.3.3 Arbiter Mode Arbitration Mode

The EBU is the default owner of the external bus (e.g. applicable when operating from external memory). Arbitration is performed if an external master (e.g. second TriCore) needs to access the external bus.

The EBU is cooperative in relinquishing ownership of the external bus while operating in Arbiter Mode. When the $\overline{\text{HOLD}}$ input is active, the EBU will generate delay any attempt to access the external bus from the internal SRI. However, the EBU is aggressive in regaining ownership of the external bus while operating in Arbiter Mode. The EBU, having yielded ownership of the bus, will always request return of ownership even if there is no EBU external bus access pending.

Arbiter Mode is selected by EBU_MODCON.ARBMODE = 01_B.

Table 507 and **Figure 728** show the functionality of the arbitration signals in Arbiter Mode.

Table 507 Function of Arbitration Pins in Arbiter Mode

Pin	Type	Pin Function in Arbiter Mode
$\overline{\text{HOLD}}$	In	In owner state (EBU is the owner of the external bus), a low level at $\overline{\text{HOLD}}$ indicates a request for bus ownership from the external master. In hold state (EBU is not the owner of the external bus), a high level at $\overline{\text{HOLD}}$ indicates that the external master has relinquished bus ownership, which causes the EBU to exit hold state.
$\overline{\text{HLDA}}$	Out	While $\overline{\text{HLDA}}$ is high, the EBU is operating in owner state. A high-to-low transition indicates that the EBU has entered hold state and that the external bus is available to the external master. While $\overline{\text{HLDA}}$ is low, the EBU is operating in hold state. A low-to-high transition indicates that the EBU has exited hold state, and has retaken ownership of the external bus.
$\overline{\text{BREQ}}$	Out	$\overline{\text{BREQ}}$ is high during normal operation. The EBU drives $\overline{\text{BREQ}}$ low two EBU clock cycles after entering hold state (after asserting $\overline{\text{HLDA}}$ low). $\overline{\text{BREQ}}$ returns high one clock cycle after the EBU has exited hold state (after driving $\overline{\text{HLDA}}$ high).

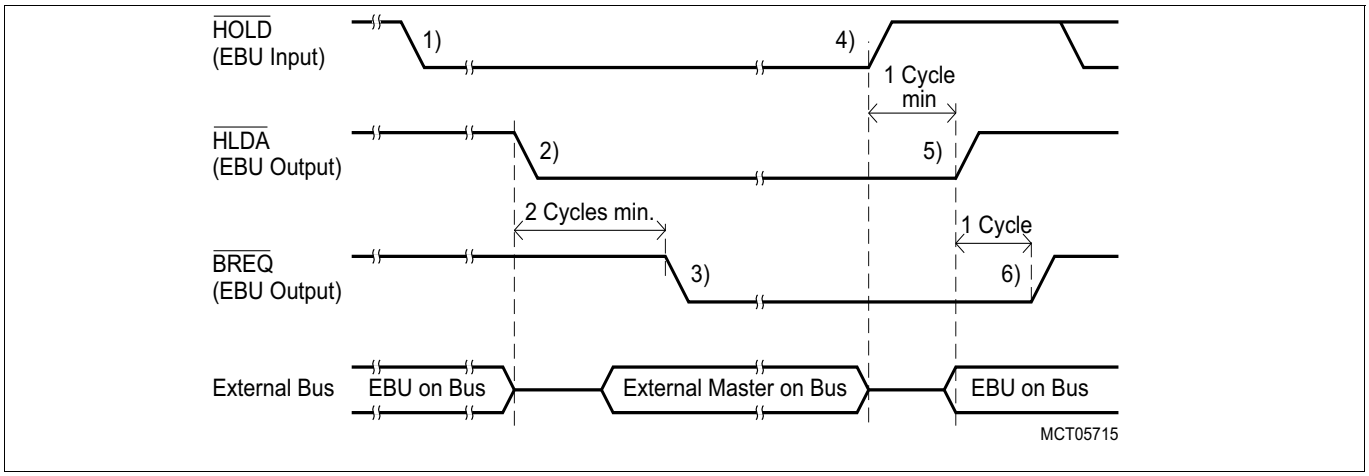


Figure 728 Arbitration Sequence with the EBU in Arbiter Mode

In Arbiter Mode, the arbitration sequence starts with the EBU as owner of the external bus.

1. The external master wants to perform an external bus access by asserting a low signal on the $\overline{\text{HOLD}}$ input.
2. When the EBU is able to release bus ownership, it enters hold state by tri-stating its bus interface lines and drives $\overline{\text{HLDA}}$ = 0 to indicate that it has released the bus. At this point, the external master ia allowed to drive the bus.
3. Two clock (EBU_CLK) cycles minimum after issuing $\overline{\text{HLDA}}$ low, the EBU drives $\overline{\text{BREQ}}$ low in order to regain bus ownership. This bus request is issued whether or not the EBU has a pending external bus access. However, the external master will ignore this signal until it has finished its bus access. This scheme assures that the external master can perform at least one complete external bus access.
4. When the external master has completed its access, it tri-states its bus interface and sets $\overline{\text{HOLD}}$ to inactive (high) level to signal that it has released the bus back to the EBU.
5. When the EBU detects that the bus has been released, it returns $\overline{\text{HLDA}}$ to high level and returns to owner state by actively driving the bus interface lines. There is always at least one clock (EBU_CLK) cycle delay from the release of the $\overline{\text{HOLD}}$ input to the EBU driving the bus.

6. Finally, the EBU deactivates the $\overline{\text{BREQ}}$ signal a minimum of one clock (EBU_CLK) cycle after deactivation of $\overline{\text{HLDA}}$. Now (and not earlier) the external master can generate a new hold request to the EBU.

This sequence assures that the EBU can perform at least one complete bus cycle before it re-enters hold state as a result of a request from the external master.

The conditions that cause change of bus ownership when the EBU is operating in Arbiter Mode are shown in **Figure 729**.

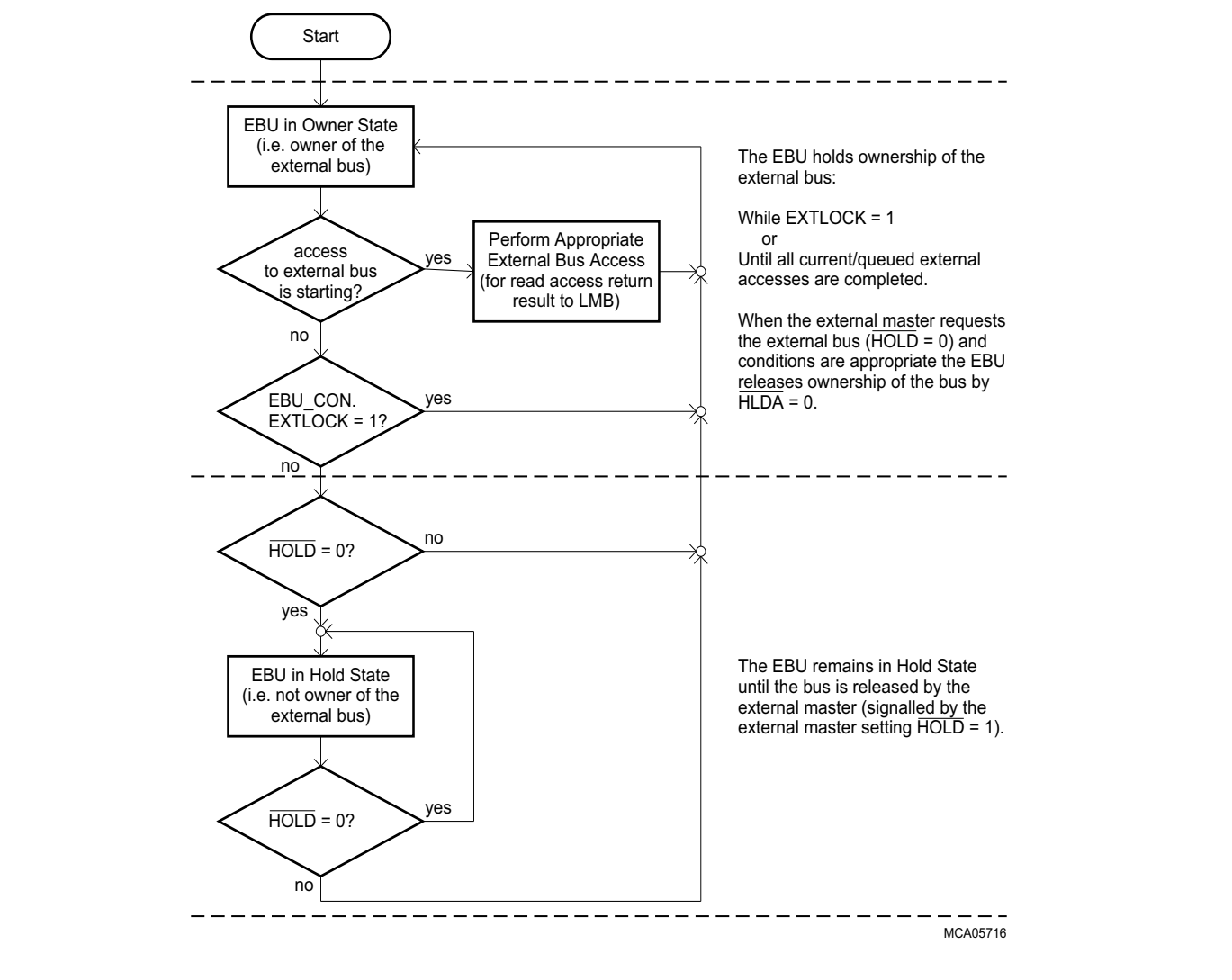


Figure 729 Bus Ownership Control in Arbiter Mode

45.3.8.3.4 “Participant Mode” Arbitration Mode

The EBU tries to gain bus ownership only in case of pending transfers (e.g. when operating from internal memory and performing stores to external memory). While the EBU is not the owner of the external bus (default state), any SRI access to the external bus will be delayed by the EBU. Any such access will cause the EBU to arbitrate for ownership of the external bus.

Once the access has been completed, the EBU will continue to accept requests from the SRI bus until the external master asserts $\overline{HOLD}$ = 0. After the external master has asserted $\overline{HOLD}$ = 0, the EBU will respond by delaying subsequent SRI accesses to external memory and will return ownership of the bus to the external master once any ongoing transaction is complete.

The use of the arbitration signals in Participant Mode is:

Table 508 Function of Arbitration Pins in Participant Mode

Pin	Type	Pin Function in Participant Mode
$\overline{HOLD}$	In	When the EBU is not in hold state ($\overline{HLDA}$ = 0) and has completely taken over control of the external bus, a low level at $\overline{HOLD}$ requests the EBU to return to hold state.
$\overline{HLDA}$	In	When the $\overline{HLDA}$ signal is high, the EBU is in hold state. When the EBU has requested ownership of the bus, the EBU is released from hold state by a high-to-low transition at $\overline{HLDA}$.
$\overline{BREQ}$	Out	$\overline{BREQ}$ remains high as long as the EBU does not need to access the external bus. When the EBU detects that an external access is required, it sets $\overline{BREQ}$ = 0 and waits for signal $\overline{HLDA}$ to become low. When the EBU has completed the external bus access (and has re-entered hold state), it will set $\overline{BREQ}$ = 1 to signal that it has relinquished ownership of the external bus.

Participant Mode arbitration mode is selected by EBU_CON.ARBMODE = 10_B.

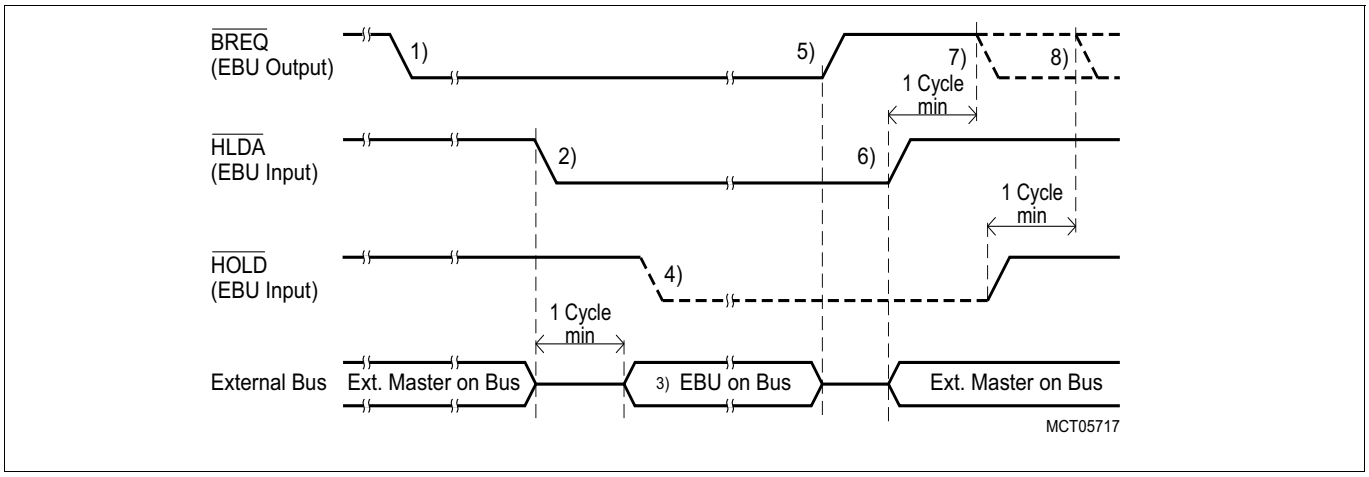


Figure 730 Arbitration Sequence with the EBU in Participant Mode

In Participant Mode, the arbitration sequence starts with the EBU in hold state.

1. The EBU detects that it has to perform an external bus access by asserting a low signal on the $\overline{BREQ}$ output.
2. When the external master is able to release bus ownership, the external master releases the external bus by tri-stating its bus interface lines and drives the $\overline{HLDA}$ = 0.
3. At least one clock (EBU_CLK) cycle after detecting $\overline{HLDA}$ = 0, the EBU will start to drive the external bus.

4. When the EBU is in owner state, the external master may optionally drive $\overline{HOLD}$ = 0 to signal that it wants to regain ownership of the external bus.
5. When the criterias are met for the EBU to release the bus ownership, the EBU enters hold state and drives $\overline{BREQ}$ = 1 output high to signal that it has released the bus.
6. When the external master detects that the EBU has released the bus ($\overline{BREQ}$ = 1), it returns $\overline{HLDA}$ to high level and takes ownership of the external bus.
7. The EBU will not request ownership of the external bus again ($\overline{BREQ}$ = 0) at least one clock (EBU_CLK) cycle after $\overline{HLDA}$ has been driven high).
8. In owner state, the EBU will not request ownership of the external bus $\overline{BREQ}$ = 0) for at least one clock (EBU_CLK) cycle after its $\overline{HOLD}$ input has been driven high.

The conditions that cause change of bus ownership when the EBU is operating in Participant Mode is shown in **Figure 729**.

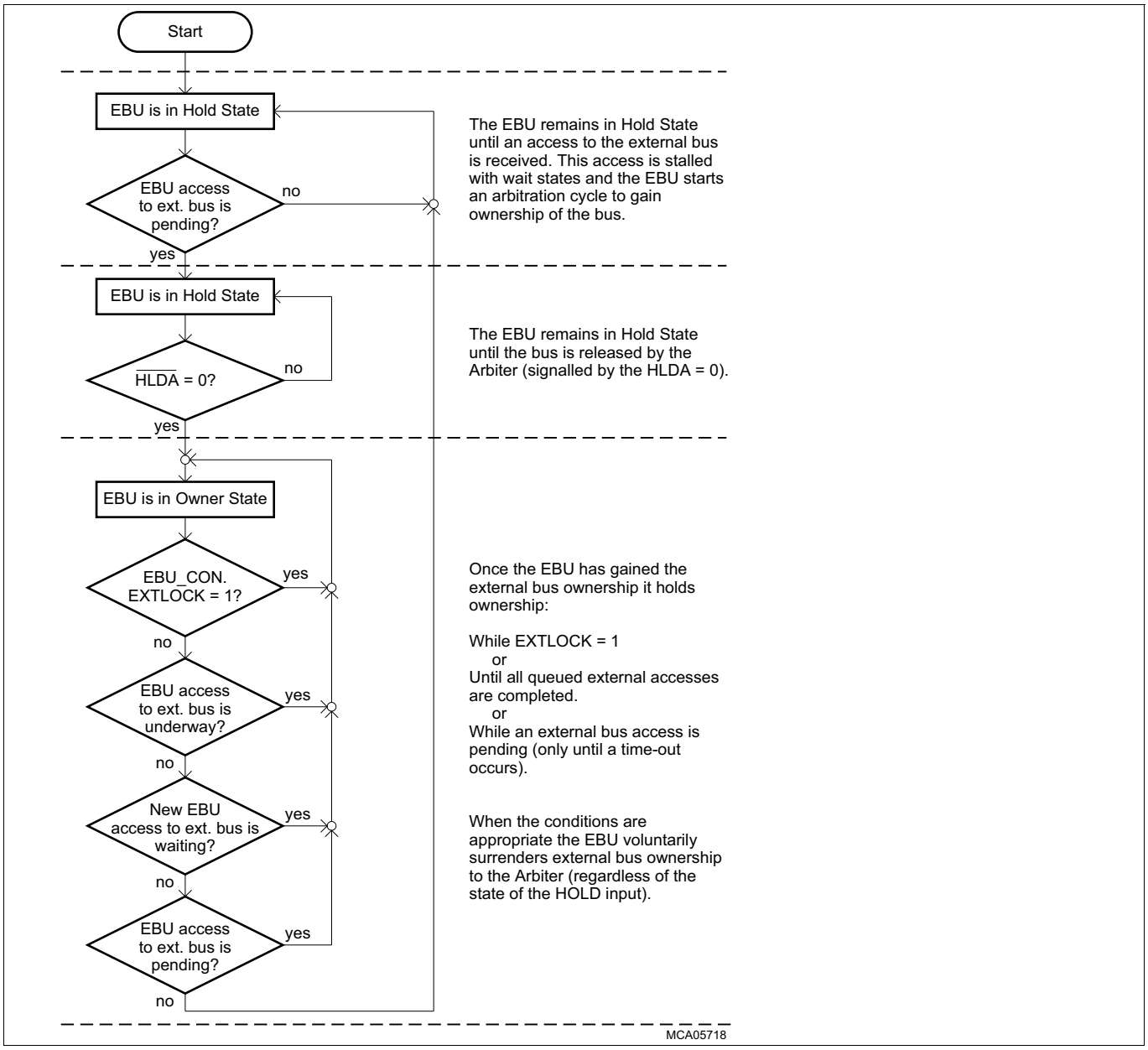


Figure 731 Bus Ownership Control with the EBU in Participant Mode

45.3.8.4 Switching Arbitration Modes

The arbitration logic will allow arbitration modes to be switched “on-the-fly” and will switch between modes as described in the following sections. However it is recommended that the arbitration mode is set as soon as possible after power on and that it is then not modified.

If this is not possible, then changes in arbitration mode should use “no bus” mode as an intermediate state with the arbiter in participant mode being placed in no bus mode first. Once both arbiters are in no bus mode, then the new modes can be set.

Note: It should be remembered that the arbitration logic drives the HLDA pin in arbiter or sole master modes and that having both sets of logic in one of these modes should be avoided to prevent contention on the signal.

45.3.8.5 Arbitration Input Signal Sampling

The sampling of the arbitration inputs can be programmed for two modes:

- Synchronous Arbitration
- Asynchronous Arbitration

When synchronous arbitration signal sampling is selected (ARBSYNC = 0), the arbitration input signals are sampled and evaluated in the same clock cycle. This mode provides the least overhead during arbitration (i.e. when changing bus ownership). The disadvantage is that the input signals must adhere to setup and hold times with respect to EBU_CLK to prevent the propagation of meta-stable signals in the EBU.

When asynchronous arbitration signal sampling is selected (ARBSYNC = 1), the arbitration signals are sampled and then fed to an additional register to be evaluated in the cycle following that in which they were sampled. This provides the EBU with good immunity to signals changing state at or around the time at which they are sampled. The disadvantage is the introduction of additional latency during arbitration (i.e. when changing bus ownership).

45.3.8.6 Locking the External Bus

The external bus can be locked to allow the EBU to perform uninterrupted sequences of external bus accesses. The EBU allows two methods of locking the external bus:

- Locked SRI accesses (i.e Read-Modify-Write)
- Lock bit EXTLOCK

When the EBU has ownership of the external bus and is performing external bus accesses in response to a locked SRI access sequence, the ownership of the external bus will not be relinquished until the locked SRI access sequence has been completed.

When lock bit EXTLOCK = 1, the EBU will hold the ownership of the external bus until EXTLOCK is subsequently cleared. If EXTLOCK is written to 1 while the EBU is the owner of the external bus, the EBU is immediately prevented from responding to requests for the external bus until EXTLOCK is cleared[1)]. If EXTLOCK is written to 1 while the EBU is not the owner of the external bus, the EBU will immediately attempt to gain ownership. When the EBU gains the ownership of the external bus the next time, the external master is prevented from regaining ownership of the external bus until EXTLOCK is again cleared.

Note: There is no time-out mechanism available for the EXTLOCK bit. When the EBU is owner of the external bus with EXTLOCK bit set, the external master will remain locked off the bus until the EXTLOCK bit is cleared by software.

1) Requests for the external bus already pending when EXTLOCK is set will not be cancelled so the EBU can give up control of the external bus after EXTLOCK is set provided that the request occurs before the EXTLOCK bit is set.

45.3.8.7 Interaction with Debug System

The EBU monitors the OCDS suspend signal used to freeze peripheral state when the system receives a break command. If the EBU detects a break it will request the external bus (if it is not the current owner). Once the EBU owns the bus, it will refuse any further arbitration requests until the suspend is released. This allows the external memories to be read by the attached debugger.

If this behaviour is not required, setting the **MODCON.OCDS_SUSP_DIS** bitfield will cause external bus arbitration to continue to operate normally during a break.

45.3.8.8 Arbitrating SDRAM control signals

Normally, the memory controller will not surrender control of a connected SDRAM type device[1)] when arbitrating the external bus. This is because the memory controller needs to keep track of which pages are open in the SDRAM and also because of restrictions on the SDRAM clock.

However, the memory controller can be programmed to tri-state the SDRAM control signals SDCLKO, CKE, $\overline{RAS}$ and $\overline{CAS}$ by setting the EBU_MODCON.SDTRI bit to 1_B. When this bit is set, SDRAM can be shared with another controller provided certain conditions are met by both memory controllers.

- The SDRAM must be in self refresh mode with the clock safely stopped before ownership of the external bus is transferred. This ensures that all pages in the SDRAM are closed and that the CKE signal is at logic zero. This can be achieved for the memory controller by setting the **SDRMREF.AUTOSELFR** to 1_B.
- The SDRAM CKE input must have a pull down sufficient to ensure that there is a guaranteed logic zero on the input while bus ownership is being transferred.

45.3.9 Start-Up/Boot Process

During reset, the EBU will be in “nobus” mode and all pins will be controlled by the Port I/O logic. After reset is removed, the EXTBOOT.EBU_CFG register filed can be written to request an external boot. If external boot is required, then the EBU will set the BUSCONx.PORTW and BUSRCONx.AGEN fields to the appropriate values and enable the necessary pins for use by the EBU. If external boot is not requested then the EBU will remain in “nobus” mode until configured by software.

45.3.9.1 Disabled (arbitration mode is “nobus”)

The EBU will come up with access to the external bus disabled after reset (i.e. no access from SRI to external memory is possible without EBU re-configuration).

45.3.9.2 External Boot Mode

The External Boot Mode of the EBU allows the EBU to boot (i.e. run all start-up code) from external memory. Immediately after reset a system may have no knowledge as to the type of memory connected to the external bus. When external boot mode is selected, the EBU will exit system firmware with the registers configures to allow slow (safe) accesses to both muxed and non-muxed asynchronous memories.

When External Boot Mode is selected, the EBU can be configured to automatically read a 32-bit Boot Configuration Value from an external memory (connected to $\overline{CS0}$, chip select region 0). The Boot Configuration Value in the external boot memory makes it possible to initialize the EBU with more appropriate configuration values for the external boot memory. These configuration values will, in turn, be used for the subsequent read accesses from the external boot memory (i.e. instruction fetches).

1) SDRAM, DDRAM & LPDDR-NVM devices

The boot configuration fetch is triggered by writing 1_B to the EBU_EXTBOOT.EBUCFG bit before the end of system boot[1]. In this case the EBU_EXTBOOT.CFGEND bit should be polled to determine when the fetch has ended and the registers updated.

The EBU_EXTBOOT.CFGERR bit will be set if the configuration word fetch fails to retrieve valid data. See **"Configuration Word Fetch Process" on Page 15** for a full description of the configuration process.

Note: External boot cannot be used if an SDRAM is connected. This is because these devices do not have an external reset input and will therefore be in an indeterminate state after a reset of the EBU. This state could cause the memory device to drive the databus and, as the SDCLKO output is disabled after reset, the memory device will remain in this state during the external boot process and interfere with the reads from the boot memory.

45.3.9.2.1 Configuration Word Fetch Process

If an External Boot configuration fetch is enabled, the EBU will perform one external bus read access to external bus address 000004_H of the memory device attached to chip select line $\overline{CS0}$. As it is assumed by the that the attached memory device has a 32 bit data bus, this is equivalent to a SRI or byte address of 000010_H because of the address translation implemented in the EBU (see **Chapter 45.3.10.4 on Page 20**).

The data read by this read access is used to configure the EBU with parameters (see **Page 17**). The boot read access itself is performed as an asynchronous access cycle with all timing parameters set to their maximum values. This access scheme supports ROMs, EPROMs or Flash memories with both separate address and data connections and also multiplexed address and data connections. **Figure 732** shows a timing example of booting from a standard demultiplexed asynchronous memory device.

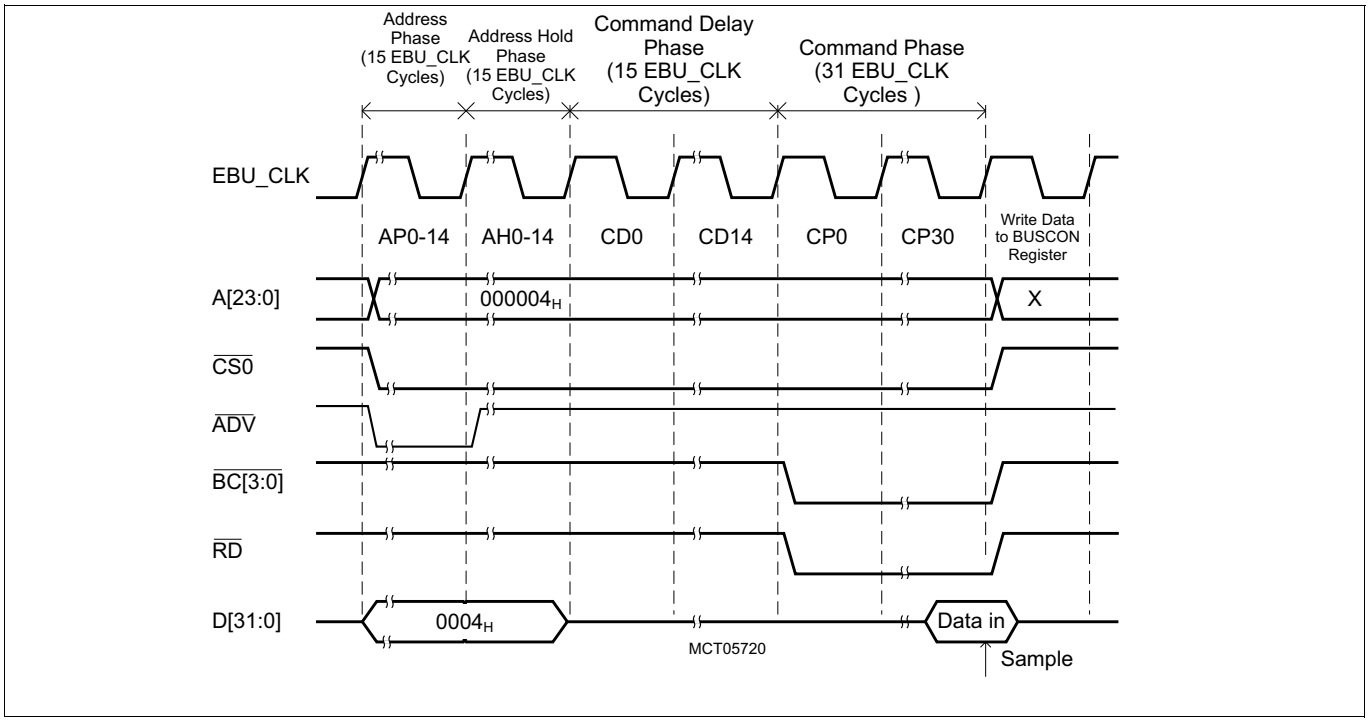


Figure 732 Boot Read Access Cycle

When a configuration fetch is triggered, the EBU waits 256 EBU_CLK clock cycles until the access is initiated, as shown in **Figure 732**. This gap is inserted to fulfill the recovery times needed by external synchronous devices

1) This is determined by the state of the BOOT_ACTIVE flag generated by the SCU.

such as Flash ROMs. During the read access, the maximum number of programmable EBU_CLK cycles is inserted (WAITRDC = 31), and the evaluation of the $\overline{WAIT}$ signal is inhibited.

Note: The boot memory must be connected to chip select line $\overline{CS0}$. The EBU assumes that the boot memory is 32-bits wide and, therefore, always reads a 32-bit configuration word at the "Data in" point shown above.

*Note: If $FFFF_H$ is returned as on bits 15 downto 0 of the configuration word during the boot read cycle (e.g. by reading the configuration word from an erased external boot memory device), the arbitration mode is set to No Bus Mode (ARBMODE = 00_B, see **"No Bus Arbitration Mode" on Page 8**). However since a value for the BCGEN field of 11_B is not legal for boot, at least one bit of the fetched word will be 0_B for valid data*

45.3.9.2.2 Boot Configuration Value

The EBU supports boot operation from 32-bit wide memories. The format of the Boot Configuration Value is as follows (Bits 31 to 16 are reserved for future expansion):

Boot Configuration Value **Reset Value:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCGEN		WAIT		WAITINV	ADDRC				WAITRDC					PORTW	

Field	Bits	Type	Description
PORTW	1:0		**Port Width** 00_B external memory is 8 bit 01_B external memory is 16 bit 10_B external memory is two 16 bit memories in parallel to make a 32 bit memory 11_B external memory is 32 bit
WAITRDC	[6:2]		**Number of Wait States for Read Accesses** Loaded into EBU_BUSRAP0.WAITRDC.
ADDRC	[10:7]		**Number of Cycles in the Address Phase** Loaded into EBU_BUSRAP0.ADDRC.
WAITINV	11		**WAIT Input Polarity Control** Loaded into EBU_BUSRCON0.WAITINV.
WAIT	[13:12]		**External Wait State Control** Loaded into EBU_BUSRCON0.WAIT.
BCGEN	[15:14]		**Byte Control Signal Control** Loaded into EBU_BUSRCON0.BCGEN.
RES	[31:16]		**Reserved** This bit field is reserved for future use. In the , RES should always be set to 0_H.

45.3.10 Accessing the External Bus

Each internal SRI master can access external devices via the EBU. The SRI address range occupied by the EBU is defined in the system address map. In addition, the EBU provides three user-programmable external memory regions which can be mapped into the SRI address space using the EBU_ADDRSELx registers. The EBU_ADDRSELs register allows the base address and size of the region to be defined.

Each of these regions is provided with a set of registers that determine the parameters of the external bus transaction as defined in **Section 45.3.10.1**. As can be seen from the table **Section 45.3.10.1**, each register set corresponds directly to a chip select signal for an external memory device. in An SRI transaction that matches one of these user-programmable external memory regions is translated by the EBU to the appropriate external access(es).

45.3.10.1 External Memory Regions

Each of the external memory regions has its own associated chip select output $\overline{CS}[2:0]$ and a set of control registers to specify the type of memory/peripheral device and the access parameters.

The access parameters for each of the regions can be programmed individually to accommodate different types of external devices. Separate control registers are available to control read and write accesses. This allows optimal access types, speeds and parameters to be chosen. Access type is configured via BUSRCONx and BUSWCONx. Access parameters are configured via BUSRAPx and BUWAPx.

Throughout this document the generic term EBU_BUSCONx is used when either of BUSRCONx or BUSWCONx is applicable and EBU_BUSAPx is used when either of BUSRAPx or BUSWAPx is applicable.

Table 509 EBU Address Regions, Registers and Chip Selects

Region	Associated Chip Select	Address Select Registers	Bus Configuration Registers	Bus Access Parameters Registers
Region 0	$\overline{CS0}$	EBU_ADDRSEL0	EBU_BUSRCON0 EBU_BUSWCON0	EBU_BUSRAP0 EBU_BUSWAP0
Region 1	$\overline{CS1}$	EBU_ADDRSEL1	EBU_BUSRCON1 EBU_BUSWCON1	EBU_BUSRAP1 EBU_BUSWAP1
Region 2	$\overline{CS2}$	EBU_ADDRSEL2	EBU_BUSRCON2 EBU_BUSWCON2	EBU_BUSRAP2 EBU_BUSWAP2

Table 510 lists the programmable parameters that are available for the three external regions (regions 0 to 2) independent of the attached memory device.

Table 510 Programmable Parameters of Regions

Register	Parameter (Bit/Bit field)	Function
EBU_ADDRSELx	ALTSEG	Alternate segment of region to be compared to SRI address bits [31:28].
	BASE	Region base address to be compared with SRI address in conjunction with the MASK parameter.
	MASK	Address mask for each external region. Specifies the number of right-most bits in the base address starting from bit 26.
	WPROT	Write Protect bit for each region.
	ALTENAB	Alternate segment enable of a region. Determines whether or not parameter ALTSEG is always compared to SRI address.
	REGENAB	Enable bit for each region. A disabled region will always generate a miss during address comparison.
EBU_BUSCONx	AGEN	Region access type: See **Section 45.3.11.1**

45.3.10.2 Address Comparison

Address Comparison is implemented in addition to the address decoding in the SRI bus matrix and is used to map SRI addresses to the EBU regions. The address comparison operation will, by default, take two clock cycles. If the frequency of the SRI bus is set to be less than or equal to 180 MHz, then the SRI interface of the EBU can be set to use single cycle decode. This is controlled by the **MODCON.FAST_SRI** bitfield. When set to 1_B, the SRI interface will use a single cycle for address decode.

Note: As the FAST_SRI bit modifies the SRI interface behaviour dynamically, some care is needed when changing the mode of operation. When setting or clearing the bit, time should be allowed after writing the register for the setting to take effect before attempting a read from the EBU. To ensure the new setting has taken effect, it is suggested that the EBU_MODCON register is written twice with the new setting. This is because internal architecture constraints mean that the data from the first write must have been transferred to the register before the second write can proceed.

45.3.10.2.1 Operation Address Comparison

Each of the three EBU regions can be programmed for independent base addresses and lengths by bits and bit fields in registers EBU_ADDRSELx.

- Bit REGEN is the enable control of a region. If the region is disabled (REGEN = 0), no address comparison will take place for the region.
- Bit field BASE specifies address bits A[31:12] of region x, where A[31:28] must only point to segments 8, 10, 13, and 14. [1)]
- Bit field MASK determines the length of a region. It specifies how many bits of anSRI SRI address must match the contents of the BASE(x) bit field (to a maximum of 15, starting with A[26]). Note that address bits A[31:27] must always match.
- Bit WPROT write protects a region. If the region is protected (WPROT = 1), no address comparison will take place for that region on a write access[2)]

1) There is no hardware lockout preventing other values being written to A[31:28], but in most cases this will result in an inaccessible memory. Enabling a memory region at segment $F8_H$ will result in the memory region conflicting with the registers. The EBU will not work correctly if this is done.

- Bit field ALTSEG determines the number of an alternate segment that can be used for address comparison with A[31:28] (if enabled by ALTENAB = 1).
- Bit ALTENAB determines whether an additional alternate segment number as defined by ALTSEG is used for address comparison.

When defining mirrored segments, the user is responsible for ensuring that there is no collision. There is no checking mechanism in hardware that ensures that each segment defined (either in BASE[31:28] or ALTSEG[11:8] or both) is exclusive. Therefore, the user must ensure that each mapping from region 0 to 3 does not interfere with any other; otherwise, only the mapping with the highest priority will take effect.

45.3.10.3 SRI Bus Width Translation

If the SRI access size is wider than the external bus width specified for the selected external region, the internal access is split in the EBU into several external accesses. For example, if the SRI requests to read a 64-bit word and the external device is only 16-bit wide, the EBU will automatically perform four external 16-bit accesses. When multiple accesses are generated in this way, external bus arbitration is blocked until the multiple access is complete. This means that the EBU remains the owner of the external bus for the duration of the access sequence. The external accesses are performed in ascending SRI address order.

45.3.10.4 Address Alignment During Bus Accesses

During an external bus access, the EBU will align the internal byte address to generate the appropriate external word or half-word address aligned to the external address pins. The address alignment will be done as follows:

- For 8 bit memory accesses
  - AD[15:0] will be driven with the value from A_{SRI}[15:0] (mutiplexed accesses only)
  - A[23:0] will be driven with the value from A_{SRI}[23:0]
- For 16 bit memory accesses
  - AD[15:0] will be driven with the value from A_{SRI}[16:1] (mutiplexed accesses only)
  - A[23:0] will be driven with the value from A_{SRI}[24:1]
- For 32 bit memory accesses
  - AD[15:0] and AD[31:16] will both be driven with the value from A_{SRI}[17:2] (accesses to paired 16-bit multiplexed devices only)
  - AD[31:0] will be driven with the right-justified and zero-padded value from A_{SRI}[31:2] (accesses to paired 32-bit multiplexed devices only)
  - A[23:0] will be driven with the value from A_{SRI}[25:2]

45.3.10.5 SRI Data Buffering

The data for all SRI writes are “posted” into a buffer in the SRI interface before the access is passed to the state machine blocks for execution. This means that all the write data is available to the state machine before the access starts independent of the relative clock frequencies of the internal and external buses and that the write completes at the initiating master before the data is written to the external memory.

2) The WPROT bit also appiles to the read phase of a read modify write, SRI transaction. This is to prevent two possible error conditions. The first would be where the read phase was accepted and the write phase errored. This is an SRI protocol violation. The second would be where the read and write accesses occurred to different memory addresses (if the read took place to a write protected region and there was an unprotected, lower priority, region mapped to the same address space).

45.3.10.6 Chip Select Control

The EBU generates three chip select signals, $\overline{CSx}$, which are all available at dedicated chip select outputs. Each chip select is associated exclusively with an EBU region. See **"External Memory Regions" on Page 18** for a complete description.

45.3.10.7 Combined Chip Select ($\overline{CSCOMB}$)

The EBU can also generate a combined, global chip select. This is controlled using the EBU_ADDRSELx.GLOBALCS register fields. If this is set to 1_B in a register, then the $\overline{CSCOMB}$ chip select is asserted whenever the associated normal chip select is asserted. Multiple GLOBALCS fields can be set simultaneously.

45.3.11 Connecting External Memories

The EBU supports interconnection to a wide variety of memory/peripheral devices with flexible programming of the access parameters. In the following sections, the basic features for these access modes are described. The types of external access cycles provided by the EBU are:

- Asynchronous accesses with multiplexed or demultiplexed address and data bus
  - ROMs, EPROMs
  - NOR flash devices
  - NAND flash devices
  - Static RAMs and PSRAMs
- Synchronous accesses with multiplexed or demultiplexed address and data bus
  - NOR flash devices (operating in burst mode)
  - PSRAMs
  - Page mode flash[1)]
- Accesses using SDRAM Type Command Protocol
  - Mobile SDRAM

Note: Not all memory types supported by the memory controller are known to be available in all quality grades and temperature ranges.

Each type of access is controlled by a state machine in the EBU kernel. See **Table 511** for a list of the access types supported and the controlling state machine.

SDRAM type accesses comply with the relevant protocol and have limited programmability. Asynchronous and Synchronous accesses use the concept of access phases of programmable duration and are more fully described in **"Phases for Asynchronous and Synchronous Accesses" on Page 24**

45.3.11.1 Programmable Device Types

Each CS region (0 to) can be individually configured using the BUSCONx.AGEN register field, to be connected to one of the following external memory/device types:

1) Page mode flash devices are not synchronous but have in common with synchronous devices the ability to return multiple data words for each supplied address. They are therefore handled by the synchronous state machine of the EBU

Table 511 agen description

agen value	Device Type	State Machine
0	Muxed Asynchronous Type (default after reset)	Asynchronous
1	Muxed Burst Type	Synchronous
2	NAND flash (page optimised)	Asynchronous
3	Muxed Cellular RAM	Synchronous
4	Demuxed Asynchronous Type	Asynchronous
5	Demuxed Burst Type	Synchronous
6	Demuxed Page Mode	Synchronous
7	Demuxed Cellular RAM	Synchronous
8	SDRAM	SDRAM
9	Reserved	Synchronous
10	Reserved	SDRAM
11	Reserved	-
12	Reserved	SDRAM
13	Reserved	Synchronous
14	Reserved	-
15	Reserved	-

The AGEN fields for the BUSRCONx and BUSWCONx can, in most cases, be set independently. This allows devices such as burst flash, which require synchronous reads and asynchronous writes to be supported efficiently.

There are device types where this capability does not confer any advantage. In these cases, writing to the BUSRCONx.AGEN field will also update the BUSWCONx.AGEN field.

The AGEN values which trigger the automatic BUSWCONx.AGEN update are:

- SDRAM: 8_D

While the BUSRCONx.AGEN field is set to one of these values, any attempt to set the value of the related BUSWCONx.AGEN field will be ignored.

45.3.11.2 Support for Multiplexed Device Configurations

Memory Controller supports a number of configurations of Multiplexed memory/peripheral devices using different values of the EBU_BUSRCONx.PORTW bit-field. The EBU_BUSWCONx registers also contain the PORTW field but in this case the field is read only and reflects the value set in the related one of the EBU_BUSRCONx registers. The values set in the EBU_BUSRCONx registers are used for both read and write accesses.

Note: When using multiplexed devices a non-zero recovery phase is mandatory for all devices to prevent read data from one access conflicting with the address for the next access to the multiplexed memory.

Table 512 Pins used to connect Multiplexed Devices to Memory Controller

Memory Device Configuration	Memory Controller Pins		
	A(23:0)[1)]	**AD(31:16)**	**AD(15:0)**
8-bit MUX	A(23:0)	-	A(7:0)/ D(7:0)
16-bit MUX	A(23:0)	-	A(15:0)/ D(15:0)
Twin 16-bit MUX	A(23:0)	A(15:0)/ D(31:16)	A(15:0)/ D(15:0)
32-bit MUX	-	A(31:16)/ D(31:16)	A(15:0)/ D(15:0)

1) These pins are always outputs which are connected to address pins on the Multiplexed device(s)

Table 513 Selection of Multiplexed Device Configuration

PORTW value	
00_B	8-bit multiplexed (deprecated)[1)]
01_B	16-bit multiplexed[2)]
10_B	Twin, 16-bit Multiplexed[3)]
11_B	32 bit multiplexed[4)]

1) 8-bit port width is only intended for NAND flash and ONFI devices which do not require particular values on the address bus. Correct operation of multiplexed addressing is not verified or guaranteed.
2) Address will only be driven onto AD(15:0) during the address and address hold phases. A(15:0) will be driven with address for duration of access
3) Lower 16 bits of address will be driven onto both A(15:0) and AD(15:0) during the address and address hold phases
4) Full address will be driven onto A(15:0) and AD(15:0) during the address and address hold phases

45.3.11.3 Support for Non-Multiplexed Device Configurations

The Memory Controller supports 16-bit and 32-bit non-multiplexed memory devices.

Table 514 Pins used to connect non-multiplexed Devices to Memory Controller

Memory Device Configuration	Memory Controller Pins		
	A(23:0)	**AD(31:16)**	**AD(15:0)**
8-bit non-MUX	A(23:0)	-	D(7:0)
16-bit non-MUX	A(23:0)	-	D(15:0)

Table 514 Pins used to connect non-multiplexed Devices to Memory Controller (cont'd)

Memory Device Configuration	Memory Controller Pins		
	A(23:0)	**AD(31:16)**	**AD(15:0)**
twin, 16-bit non-MUX	A(23:0)	D(31:16)	D(15:0)
32-bit non-MUX	A(23:0)	D(31:16)	D(15:0)

Table 515 Selection of non-Multiplexed Device Configuration

PORTW value	
00_B	8-bit[1)]
01_B	16-bit
10_B	Twin 16-bit
11_B	32-bit

1) 8-bit port width is only intended for NAND flash and ONFI devices which do not require particular values on the address bus. Correct operation of addressing for other device types is not verified or guaranteed

45.3.12 Phases for Asynchronous and Synchronous Accesses

Accesses to asynchronous and synchronous devices are composed of a number of standard access phases (according to the type of device and the type of access). Each output signal for the EBU is active in defined phases. The length of the phases can be programmed to adjust the pulse width and timing relationships of the output signals

There are six access phases defined:

- Address Phase AP (mandatory for read and write cycles of both device types)
- Command Delay Phase CD (optional)
- Command Phase CP (mandatory)
- Burst Phase BP (Synchronous Accesses only)
- Data Hold Phase DH (optional, only applies to write cycles)
- Recovery Phase RP (optional)

Throughout the remainder of this document, a short-hand notation is adopted to represent any clock cycle in any phase. This notation consists of two or three letters followed by a number. The letters identify the access phase within which the clock cycle is located (e.g. AP for Address Phase). The number denotes the number of EBU_CLK clock cycles within the phase (i.e. 1 = first, etc.). In the case of delays that can be extended by external control inputs the lower case letters "e" and "i" are inserted following the two letter phase identifier to differentiate between internally ("i") and externally ("e") generated delays. For example, AP2 identifies the second clock in the Address Phase. CPe3 identifies the third clock in the Command Phase which is being extended by external wait-states.

45.3.12.1 Address Phase (AP)

The Address Phase is mandatory. It always consists of at least one or more EBU_CLK cycles. The phase can be optionally extended to accommodate slower devices.

At the start of the Address Phase, the EBU:

- Selects the device to be accessed by asserting the appropriate $\overline{CSx}$ signal,
- Issues the address which is to be accessed on the address bus,
- For multiplexed devices, drives the address onto the muliplexed address/data bus,

- Asserts the $\overline{ADV}$ signal low,[1)]
- Asserts the appropriate $\overline{BCx}$ signals if these are programmed to be asserted with the $\overline{CSx}$ signal,
- At the end of the Address Phase the EBU returns the $\overline{ADV}$ signal to high.

The length (number of EBU_CLK cycles) of the Address Phase is programmed via the EBU_BUSAPx.ADDRC bit field parameter.

45.3.12.2 Address Hold Phase (AH)

The Address Hold Phase is optional. It consists of zero or more EBU_CLK cycles. It is intended to provide hold time for the multiplexed address bits after the $\overline{ADV}$ signal has returned to the inactive state.

At the end of the address hold phase, the multiplexed address will be removed from the bus:

- During a read access, the multiplexed address/data bus will return to the high impedance condition to allow the read data to be driven by the external memory
- During a write access, the write data will be driven onto the multiplexed address/data bus

45.3.12.3 Command Delay Phase (CD)

The Command Delay phase is optional. This means that it can also be programmed for a length of zero EBU_CLK clock cycles. The CD phase allows for the insertion of a delay between Address Phase (or optional Address Hold phase) and Command Phase(s). This phase accommodates devices that are not fast enough to receive commands immediately after getting the address or multiplexed devices which require a bus turnaround delay on reads.

The length (number of EBU_CLK cycles) of the Command Delay phase is programmed via the EBU_BUSAPx.CMDDELAY bit field. This parameter makes it possible to select between zero to seven Command Delay phases.

1) If an active high, ALE, signal is required, the polarity of the $\overline{ADV}$ output can be inverted by setting the ALE field of the EBU_MODCON register.

45.3.12.4 Command Phase (CP)

The Command Phase is mandatory for asynchronous devices. It always consists of at least one or more EBU_CLK cycles. The phase can optionally be extended to accommodate slower devices.

The length (number of EBU_CLK cycles) of the Command Phase is separately programmable for read and write accesses. Bit field EBU_BUSAPx.WAITRDC determines the basic length of Command Phases during read cycles and bit field EBU_BUSAPx.WAITWRC determines the basic length of Command Phases during write cycles.

Additionally, when accessing asynchronous devices, a Command Phase can also be extended externally using the $\overline{\text{WAIT}}$ signal when the region being accessed is programmed for external command delay control via bit EBU_BUSCONx.WAIT.

The Command Phase is further subdivided into:

- CPi (= internally-programmed Command Phase)
- CPe (= externally-prolonged Command Phase, i.e. prolonged by the assertion of the $\overline{\text{WAIT}}$ signal).

At the start of the Command Phase, the EBU:

- Asserts the appropriate control signal $\overline{\text{RD}}$ or RD/$\overline{\text{WR}}$ low according to the access type (read or write),
- Issues the data to be written on the data bus AD[15:0] (in the case of a write cycle),
- Asserts the appropriate $\overline{\text{BCx}}$ low (in the case where $\overline{\text{BCx}}$ is programmed to be asserted with the $\overline{\text{RD}}$ or RD/$\overline{\text{WR}}$ signals).

At the end of the Command Phase during an asynchronous access, the EBU:

- Returns the appropriate control signal $\overline{\text{RD}}$ or RD/$\overline{\text{WR}}$ high according to the type of access type (read or write),
- Latches the data from the data bus AD[15:0] (in the case of a read cycle),
- Returns the appropriate $\overline{\text{BCx}}$ high (in the case where $\overline{\text{BCx}}$ is programmed to be asserted with the $\overline{\text{RD}}$ or RD/$\overline{\text{WR}}$ signals).

At the end of the Command Phase during a synchronous access, the $\overline{\text{RD}}$ or RD/$\overline{\text{WR}}$ signals hold their state and the state machine transfers to the Burst Phase. Transition to Burst Phase will always occur synchrnously with a rising edge of BFCLKO. If necessary, the Command Phase will be extended to ensure that this happens.

45.3.12.5 Data Hold Phase (DH)

The Data Hold phase is optional. This means that it can also be programmed for a length of zero EBU_CLK clock cycles. Furthermore, it is only available for asynchronous write accesses (with two exceptions, see below). The Data Hold phase extends the amount of time for which data is still held on the bus after the rising edge of the RD/$\overline{\text{WR}}$ signal occurred. The Data Hold phase is used to accommodate external devices that require a data hold time after the rising edge of the RD/$\overline{\text{WR}}$ signal. The length (number of EBU_CLK cycles) of the Data Hold phase is programmed via the EBU_BUSAPx.DATAC bit field.

45.3.12.5.1 Exceptional use of Data Hold

In two cases, signals other than the data bus may need to have a hold time maintained relative to RD/$\overline{\text{WR}}$ or $\overline{\text{RD}}$. These cases occur when $\overline{\text{ADV}}$ and $\overline{\text{BAA}}$ are being used as NAND flash control signals which happens for AGEN settings of 2_D or 13_D. For these two settings, the Data Hold Phase will apply to both reads and writes.

45.3.12.6 Burst Phase (BP)

The Burst Phase is mandatory during burst accesses. At the end of the Burst Phase the EBU reads data from the data bus or updates the write data. During a burst access, Burst Phases are repeated as many times as required in order to read or write the required amount of data from or to the external memory device.

At the start of the first Burst Phase during a burst read access, the EBU:

- Drives the $\overline{BAA}$ signal low to cause the Burst Flash device to advance the address with each subsequent BFCLKO positive edge.

The first burst phase of an access will always start on arising edge of BFCLKO. If necessary, the length of the previous phase will be extended to ensure that this happens.

At the end of the last Burst Phase during a burst access, the EBU:

- Returns the $\overline{BAA}$ signal high,
- Returns the $\overline{CSx}$ signal high,
- Returns the $\overline{RD}$ signal high.
- Returns the RD/$\overline{WR}$ signal high

provided that a Control Hold phase has not been programmed.

During accesses to Burst Flash devices the length of the Burst Phase will be programmed such that the end of the Burst Phase always coincides with a positive edge of the appropriate BFCLKO (Burst Flash Clock) signal.

A Burst Phase is always at least one clock cycle in length. The length of each Burst Phase (i.e. the number of cycles) is derived from the value of the EXTCLOCK and EXTDATA fields in the EBU_BUSAPx register. The length of the burst phase will be either be:

- one period of BFCLKO if EXTDATA is 00_B,
- two periods of BFCLKO if EXTDATA is 01_B.
- four periods of BFCLKO if EXTDATA is 10_B.
- eight periods of BFCLKO if EXTDATA is 11_B.

45.3.12.7 Recovery Phase (RP)

The Recovery Phase is optional (although for access types which would cause a bus contention a single cycle of recovery is normally forced by the memory controller logic). This means that it can also be programmed for a length of zero EBU_CLK clock cycles. This phase allows the insertion of a delay following an external bus access that delays the start of the Address Phase for the next external bus access. This permits flexible adjustment of the delay between accesses to the various external devices. The following individually programmable delays are provided on a region by region basis for the following conditions:

- Bit fields EBU_BUSAPx.RDRECOVC determine the basic length of the Recovery Phase after a read access.
- Bit fields EBU_BUSAPx.WRRECOVC determine the basic length of the Recovery Phase after a write access.
- Bit fields EBU_BUSAPx.DTACS determine the length (basic number of EBU_CLK clock cycles) of the Recovery Phase after a read/write access of one region that is followed by a read/write access of another region or a read to one region is followed by a write to the same region (BUSRAPx.DTACS) or a write to one region is followed by a read to the same region (BUSWAPx.DTACS).

The EBU implements a “highest wins” algorithm to ensure that the longest applicable recovery delay is always used between consecutive accesses to the external bus. **Table 516** shows the scheme for determining this delay for all possible circumstances. For example, if a read access to a region associated with $\overline{CS1}$ is followed by a write to a region associated with $\overline{CS2}$, the delay will be the highest of BUSRAP1.DTACS and BUSRAP1.RDRECOVC. In this case, if BUSRAP1.DTACS is greater than BUSRAP1.RDRECOVC, then the number of recovery cycles between the two accesses is BUSRAP1.DTACS clock cycles (minimum).

Table 516 Parameters for Recovery Phase

Case			Parameter(s) used to calculate "Highest Wins" Recovery Phase
Region	Current Access	Next Access	
Same $\overline{CSn}$	Read	Read	RDRECOVC
	Write	Write	WRRECOVC
	Read	Write	BUSRAPx.DTACS
	Write	Read	BUSWAPx.DTACS
Different $\overline{CSn}$	Read	Read	BUSRAPx.DTACS, RDRECOVC
	Write	Write	BUSWAPx.DTACS, WRRECOVC
	Read	Write	BUSRAPx.DTACS, RDRECOVC
	Write	Read	BUSWAPx.DTACS, WRRECOVC

45.3.13 Asynchronous Read/Write Accesses

Asynchronous read/write access of the EBU support the following features:

- EBU_CLK clock-synchronous signal generation
- Support for 8-bit, 16-bit and 32-bit bus width
  Performing an SRI access with a data size greater than that of the external device automatically triggers a sequence of the appropriate number of external accesses to match the SRI access width.
- Demultiplexed address/data lines
- Programmable access parameters
  - Internal control of command delay cycles
  - External and/or internal control of wait states
  - Variable data hold cycles for write operation (to allow flexible hold time adjustment)
  - Variable inactive/recovery cycles when:
    Switching between different memory regions (CS),
    Switching between read and write operations,
    After each read cycle,
    After each write cycle.

Software driver routines are required in order to support Nand Flash devices using asynchronous device accesses. A single Nand Flash access sequence is performed by generating the appropriate sequence of discrete asynchronous device accesses in software.

45.3.13.1 Signal List

The following signals of the EBU are used for asynchronous accesses:

Table 517 Asynchronous Mode Signal List

Signal/Pin	Type	Function
AD[31:0]	O	Address/Data bus lines 0-31
A[23:0]	O	Address bus lines 0-23
$\overline{\text{CS}}$[2:0]	O	Chip select 0-3
$\overline{\text{RD}}$	O	Read control line
RD/$\overline{\text{WR}}$	O	Write control line
$\overline{\text{BC}}$[3:0]	O	Byte control lines 0-3
$\overline{\text{WAIT}}$	I	Wait input

45.3.13.2 Standard Asynchronous Access Phases

Accesses to asynchronous devices are composed of a subset of the standard access phases which are detailed in **Section 45.3.12**. The standard access phases for asynchronous devices are:

- AP: Address Phase (compulsory - see **Page 24**)
- AH: Address Hold Phase (Optional - see **Page 25**)
- CD: Command Delay Phase (optional - see **Page 25**)
- CP: Command Phase (compulsory - see **Page 26**)
- DH: Data Hold Phase (optional - see **Page 26**)
- RP: Recovery Phase (optional - see **Page 27**)

Figure 733 above shows an example of an access to a non-multiplexed device and **Figure 734** an example of an access to a multiplexed device.

45.3.13.3 Example Waveforms

The following figures show example waveforms for asynchronous accesses

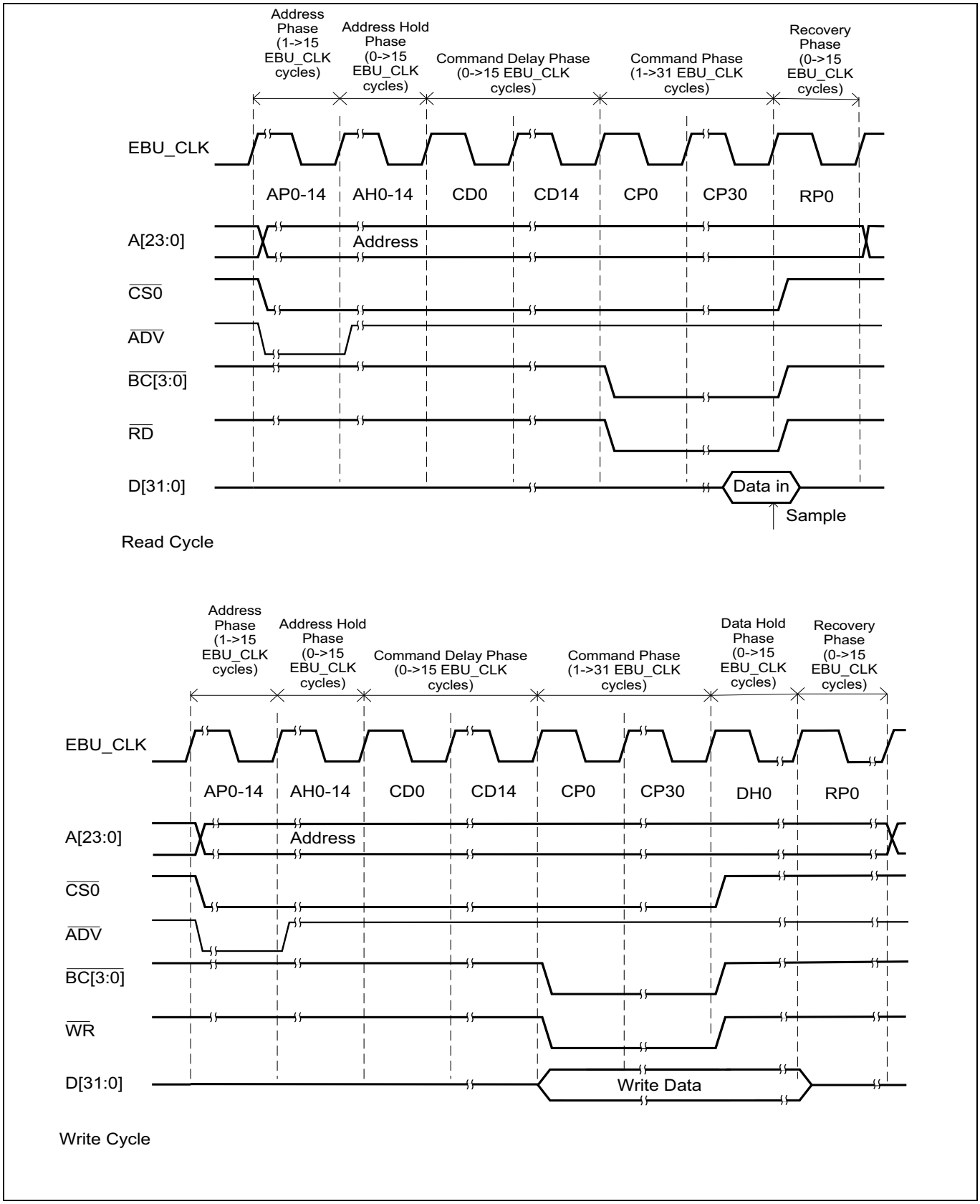


Figure 733 Asynchronous non-muxed Access

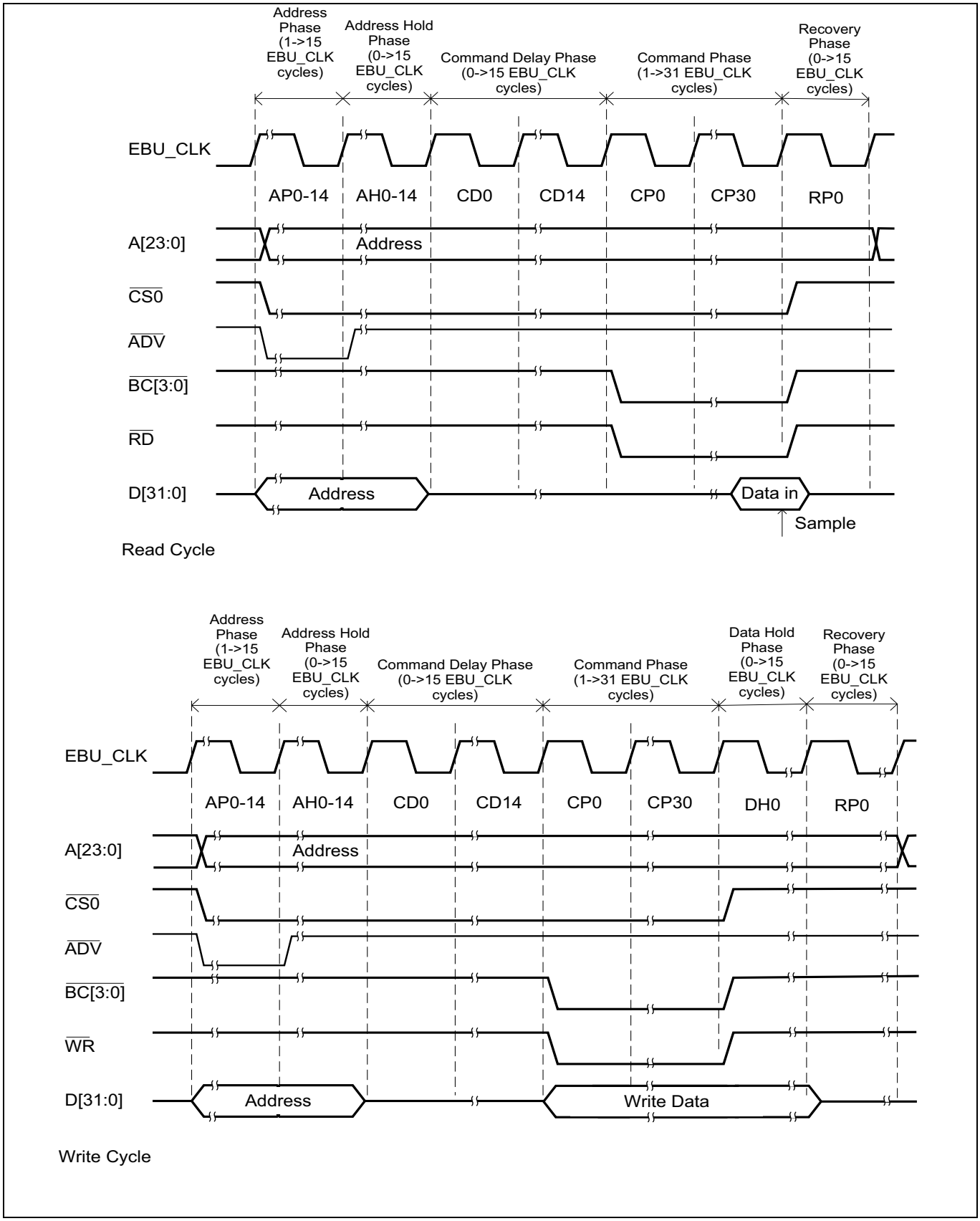


Figure 734 Asynchronous muxed Access

45.3.13.4 Control of $\overline{ADV}$ & Other Signal Delays During Asynchronous Accesses

For asynchronous accesses, the Memory Controller output signals: $\overline{ADV}$, $\overline{CS}$, $\overline{RD}$, RD/$\overline{WR}$, and $\overline{BC}$ signals can be delayed with respect to the start of the access phases they are asserted in. The amount by which the signal is delayed depends on the setting of the EBU_BUSRAPx.EXTCLOCK field of the region being addressed for read accesses and the EBU_BUSWAPx.EXTCLOCK field of the region being addressed for writes:-

- When EXTCLOCK is set to 00_B, control signals are asserted on the negative edge of . i.e. they are in effect delayed by an EBU_CLK high pulse width (T_{PH})with respect to the other signals.
- When EXTCLOCK is not set to 00_B, control signals are asserted on the next positive edge of . i.e. they are in effect delayed by an cycle (T_{CLK}) with respect to the other signals.

The Memory Controller allows these delays to be enabled and disabled independently via the register bits EBU_BUSCONx.EBSE for $\overline{ADV}$ and EBU_BUSCONCx.ECSE for the other control signals. The default setting after reset has the delay disabled.

Table 518 $\overline{ADV}$ Signal Timing

EXTCLOCK is set to	ADV Falling Edge Position		ADV Rising Edge Position	
	Delay Disabled[1]	Delay Enabled	Delay Disabled[1]	Delay Enabled
00_B	Start of AP1	Start of AP1 + T_{PH}	Start of AP1	Start of AP1 + T_{PH}
01_B, 10_B, 11_B	Start of AP1	End of AP1	Start of AP1	End of AP1

1) See **Figure 733** for details of this signal positioning.

Table 519 $\overline{RD}$ and RD/$\overline{WR}$ Signal Timing

EXTCLOCK is set to	Set at:		Cleared at:	
	Delay Disabled[1]	Delay Enabled	Delay Disabled[1]	Delay Enabled
00_B	Start of CP1	Start of CP1 + T_{PH}	End of CPn[2]	End of CPn + T_{PH}
01_B, 10_B, 11_B	Start of CP1	End of CP1	End of CPn	End of CPn + T_{CLK}

1) See **Figure 733** for details of this signal positioning.

2) CPn indicates the final Command Phase.

Table 520 $\overline{CS}$ Signal Timing

EXTCLOCK is set to	Set at:		Cleared at:	
	Delay Disabled[1]	Delay Enabled	Delay Disabled[1]	Delay Enabled
00_B	Start of AP1	Start of AP1 + T_{PH}	End of DHn[2]	End of DHn + T_{PH}
01_B, 10_B, 11_B	Start of AP1	End of AP1	End of DHn	End of DHn + T_{CLK}

1) See **Figure 733** for details of this signal positioning.

2) DHn indicates the final Data Hold Phase. This is replaced by CPn, the final Command Phase, if the programmed Data Hold phase length is zero clocks.

The byte control signals, $\overline{BC}$x, can either use the timing in **Table 519** if EBU_BUSCONx.BCGEN is set to 01_B or 10_B or the timing in **Table 520** if EBU_BUSCONx.BCGEN is set to 00_B. A EBU_BUSCONx.BCGEN value of 11_B is not valid for asynchronous accesses.

Write data is handled differently to the other signals. Delays are never applied to the write data.

Note: If the control signals are delayed a recovery phase must be used to prevent conflicts between accesses as the rising edge of the control signals will be delayed past the end of the command phase. If a

multiplexed access is used without a recovery phase, the address for the next access will be delayed by one clock cycle to enforce a bus turnaround time on the data bus, resulting in the valid address being driven one clock after ADV is asserted.

45.3.13.5 Programmable Parameters

Table 521 lists the programmable parameters for asynchronous accesses.

Table 521 Asynchronous Access Programmable Parameters

Register	Parameter (Bit/Bit field)	Function
EBU_BUSAPx	ADDRC	Number of cycles in address phase
	CMDDELAY	Number of programmed command delay cycles[1].
	AHOLD	Number of Cycles in Address Hold Phase[1]
	WAITRDC	Number of programmed wait states for read accesses.
	WAITWRC	Number of programmed wait states for write accesses.
	DATAC	Number of Data Hold cycles.
EBU_BUSAPx	RDRECOVC	Number of minimum recovery cycles after a read access.
	WRRECOVC	Number of minimum recovery cycles after a write access.
	DTARDWR	Number of minimum recovery cycles between a read access and a write access.
	DTACS	Number of minimum recovery cycles when the next access going to a different memory region.
EBU_BUSCONx	WAIT	External Wait State control (OFF, asynchronous, synchronous)
	WAITINV	Reversed polarity at WAIT: active low or active high

1) This phase can be programmed for devices without multiplexed address and data busses but serves no purpose other than extending the access

45.3.13.6 Asynchronous Access Control

In general, there are only two critical phase during accesses to asynchronous devices with separate address and data buses. These phases are:

- **Command Phase** (see Page 26) which is used to control the widths of the Output Enable and Write Enable pulses. The length of the Command Phase is set by the bit fields EBU_BUSRAPx.WAITRDC and EBU_BUSWAPx.WAITWRC.
- **Data Hold Phase** (see Page 26) which is used to ensure that the write data is stable long enough to allow it to be successfully latched into the device. The length of the Command Phase is set by the bit field EBU_BUSWAPx.DATAC

As the “address to data valid” and the “chip select to data” valid parameters of asynchronous devices are usually greater than “output enable to data valid” parameter, the lengths of the “Address”, “Address Hold” and “Command Delay” phases can be used to extend the time between the start of the access when the address and chip select are driven and the end of the Command Phase when the read or write event occurs.

For devices with a multiplexed address and data buses two additional phases control the latching of the address into the device and also critical:

- **Address Phase** (see Page 24) which is used to control the width of the ADV pulse. The length of the Address Phase is available via bit fields EBU_BUSAPx.**ADDRC**

- **Address Hold Phase** (see **Page 25**) which is used to ensure that the address is stable long enough to allow it to be successfully latched into the device. The length of the Address Phase is available via bit fields EBU_BUSAPx.**AHOLDC**

45.3.13.6.1 External Extension of the Command Phase by WAIT

The $\overline{\text{WAIT}}$ input can be used to cause the EBU to extend the Command Phase by inserting additional cycles prior to deactivation of the $\overline{\text{RD}}$ and RD/$\overline{\text{WR}}$ lines. This signal can be programmed separately for each region to be ignored or sampled either synchronously or asynchronously (selected via the EBU_BUSCONx.WAIT bit field). Additionally, the polarity of $\overline{\text{WAIT}}$ can be programmed for active low (default after reset) or active high function via bit EBU_BUSCONx.WAITINV. The signal will only take effect after the programmed number of Command Phase cycles has passed. This means that the signal can only be used to extend the phase, not to shorten it.

Table 522 Operation of WAIT input

Value of BUSCONx.WAIT	Mode of the $\overline{\text{WAIT}}$ input
0_D	OFF (default after reset).
1_D	Asynchronous input at WAIT.
2_D	Synchronous input at WAIT.
3_D	reserved

When programmed for synchronous operation, $\overline{\text{WAIT}}$ is sampled on every rising edge of EBU_CLK during the Command Phase. The sampled value is then used on the next rising edge of EBU_CLK to decide whether to prolong the Command Phase or to start the next phase. **Figure 735** shows an example of $\overline{\text{WAIT}}$ used in Synchronous Mode.

Note: Due to the one-cycle delay in Synchronous Mode between the sampling of the $\overline{\text{WAIT}}$ input and its evaluation by the EBU, the Command Phase must always be programmed to be at least one EBU_CLK cycle (via EBU_BUSAPx.WAITRDC or EBU_BUSAPx.WAITWRC) in this mode.

When programmed for asynchronous operation, $\overline{\text{WAIT}}$ is also sampled at each rising edge of EBU_CLK during the Command Phase. However, an extra synchronization cycle is inserted prior to the use of the sampled value. This means that the sampled value is not used until the second following rising edge of EBU_CLK. **Figure 736** shows an example of $\overline{\text{WAIT}}$ used in Asynchronous Mode.

Note: Due to the two-cycle delay in Asynchronous Mode between the sampling of the $\overline{\text{WAIT}}$ input and its evaluation by the EBU, the Command Phase must always be programmed to be at least two EBU_CLK cycles (via EBU_BUSAPx.WAITRDC or EBU_BUSAP.WAITWRC) in this mode.

Figure 735 shows an example of the extension of the Command Phase through the $\overline{\text{WAIT}}$ input in synchronous mode:

- At EBU_CLK edge 1 (at the end of the Address Phase), the EBU samples the $\overline{\text{WAIT}}$ input as low and starts the first cycle of the Command Phase (CPi1 - internally programmed).
- At EBU_CLK edge 2, the EBU samples the $\overline{\text{WAIT}}$ input as low and starts an additional Command Phase cycle (CPe2 - externally generated) as a result of the $\overline{\text{WAIT}}$ input sampled as low at EBU_CLK edge 1.
- At EBU_CLK edge 3, the EBU samples the $\overline{\text{WAIT}}$ input as high and starts an additional Command Phase cycle (CPe3 - externally generated) as a result of the $\overline{\text{WAIT}}$ input sampled as low at EBU_CLK edge 2.
- Finally at EBU_CLK edge 4, as a result of the $\overline{\text{WAIT}}$ input sampled as high at point 3, the EBU terminates the Command Phase, reads the input data from D[31:0] and starts the Recovery Phase.

Note: Synchronous operation means that even though access to the device may be asynchronous, the control logic generating the control signals must meet setup and hold time requirements with respect to EBU_CLK.

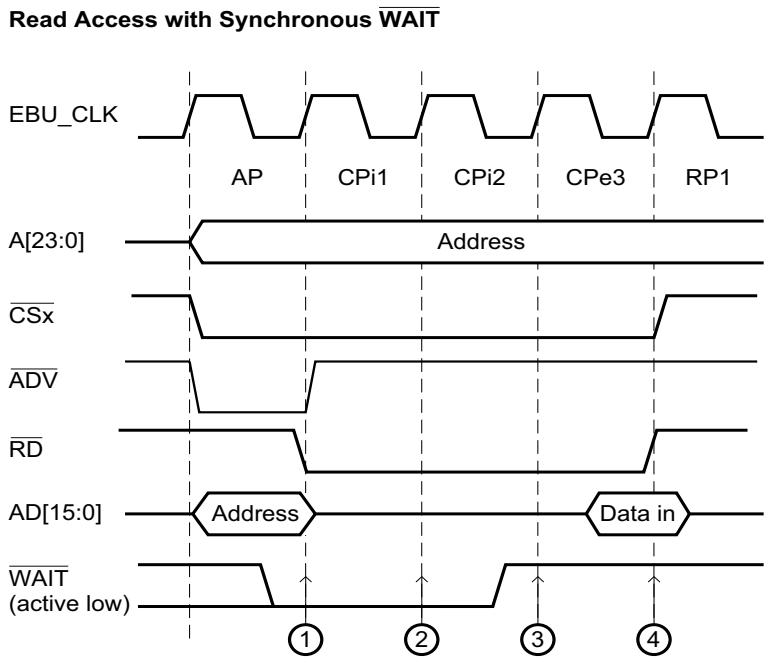


Figure 735 External Wait Insertion (Synchronous Mode)

Figure 736 shows an example of the extension of the Command Phase through the WAIT input in asynchronous mode:

- At EBU_CLK edge 1 (at the end of the Address Phase), the EBU samples the WAIT input as low and starts the first cycle of the Command Phase (CPi1 - internally programmed).
- At EBU_CLK edge 2, the EBU samples the WAIT input as low and starts the second cycle of the Command Phase (CPi2 - internally programmed).
- At EBU_CLK edge 3, the EBU samples the WAIT input as high and starts an additional Command Phase cycle (CPe3 - externally generated) as a result of the WAIT input sampled as low at EBU_CLK edge 1.
- At EBU_CLK edge 4, the EBU starts an additional Command Phase cycle (CPe4 - externally generated) as a result of the WAIT input sampled as low at EBU_CLK edge 2.
- Finally at EBU_CLK edge 5, as a result of the WAIT input sampled as high at EBU_CLK edge 3, the EBU terminates the Command Phase, reads the input data from AD[15:0],and starts the Recovery Phase.

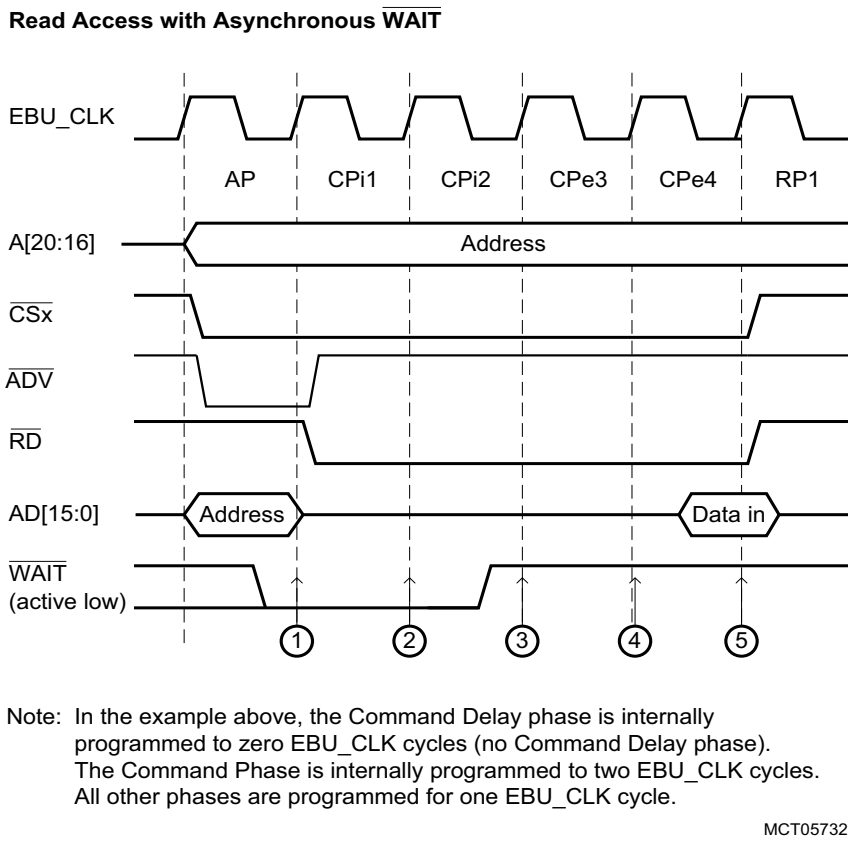


Figure 736 External Wait Insertion (Asynchronous Mode)

45.3.13.7 Interfacing to Asynchronous Nand Flash Devices

The memory controller provides support for specific Nand Flash devices. The required access sequences (read or write) are generated by connecting the Nand Flash device as an Asynchronous Device and using appropriate processor generated access sequences to emulate the NAND flash commands. **Figure 737** Shows an example of Memory Controller connected to a Nand Flash device:-

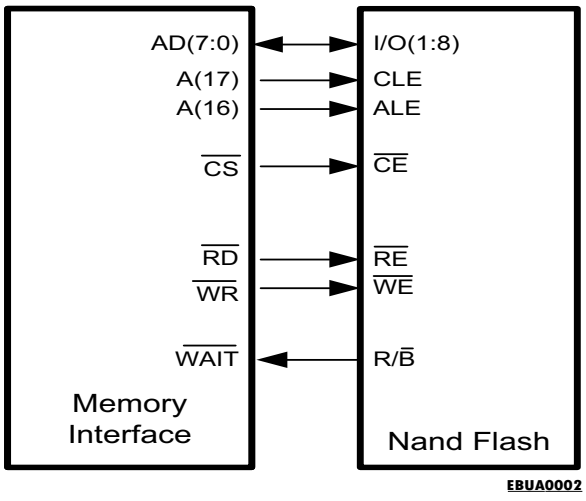


Figure 737 Example of interfacing a Nand Flash device to the Memory Controller

The R/$\overline{B}$ input from the NAND flash is connected to the memory controller $\overline{WAIT}$ input and is available as the EBU_MODCON.STS. This enables a NAND flash to be driven by software from the processor.

In the instance shown above two address lines are connected to the Nand Flash, and rather than being connected to address inputs, they are connected to control inputs. This allows access to three “registers” in the Nand Flash as follows:-

Table 523 Nand Flash “Registers” (8 bit device)

SRI Address	“Register”	Comment
Base + 00000_H	Data Register	Read/Write: Used to read data from and write data to the device.
Base + 10000_H	Address Register	Write only: Used to write the required access address to the device.
Base + 20000_H	Command Register	Write only: Used to write the required command to the device.

Note: SRI addresses are byte addresses and addresses on the external bus are word addresses. The SRI address of the virtual register will depend on the port width. of the addressed memory region. See ***“SRI Bus Width Translation” on Page 20***

45.3.13.7.1 NAND flash page mode

NAND flash memories are page oriented devices capable of extended read operations with a single setup phase for command signals at the beginning of the access. The asynchronous controller of the Memory Controller will split a large transfer into multiple accesses to external memory but each of these accesses will have the overhead of the initial setup phase. Enabling page mode, by setting the agen field in EBU_BUSCONx to 2_D, will cause the standard flow of the controller to be modified as follows:

- For a read, if data remains to be fetched at the end of a command phase, the controller will start a new command delay phase, instead of a new address phase or recovery phase and the address will not be incremented. If EBU_BUSRAPx.cmddelay is set to zero, the command delay phase will have a duration of one clock cycle but in this case the command delay phase is mandatory to ensure that the $\overline{RD}$ and RD/$\overline{WR}$ signals return to the high state.
- For a write, if data remains to be written at the end of a data hold phase (or command phase if the length of data hold is zero), the controller will start a new command phase, instead of a new address phase or recovery phase and the address will not be incremented. If EBU_BUSWAPx.datac is set to zero, the data hold phase will have a duration of one clock cycle as in this case the data hold phase is mandatory to ensure that the $\overline{RD}$ and RD/$\overline{WR}$ signals return to the high state. The command phase will be forced to have a minimum length of two clocks.

Enabling NAND flash page mode will also reconfigure the EBU signals used for ALE and CLE. While A(17:16) can still be used directly, the same values will also be output on $\overline{ADV}$ and $\overline{BAA}$.

- $\overline{ADV}$ will be used as ALE and will output the value of A(16).
- $\overline{BAA}$ will be used as CRE and will output the value of A(17).

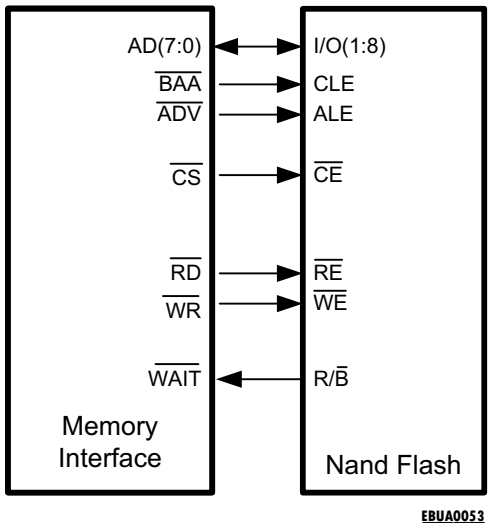


Figure 738 NAND Flash Connection for AGEN=3_D

To allow accesses to run consecutively without violating $\overline{\text{ADV}}$ and $\overline{\text{BAA}}$ timing restrictions, both signals will be set inactive (1_B) at the beginning of the recovery phase of the access.

See **Figure 739** for example waveforms.

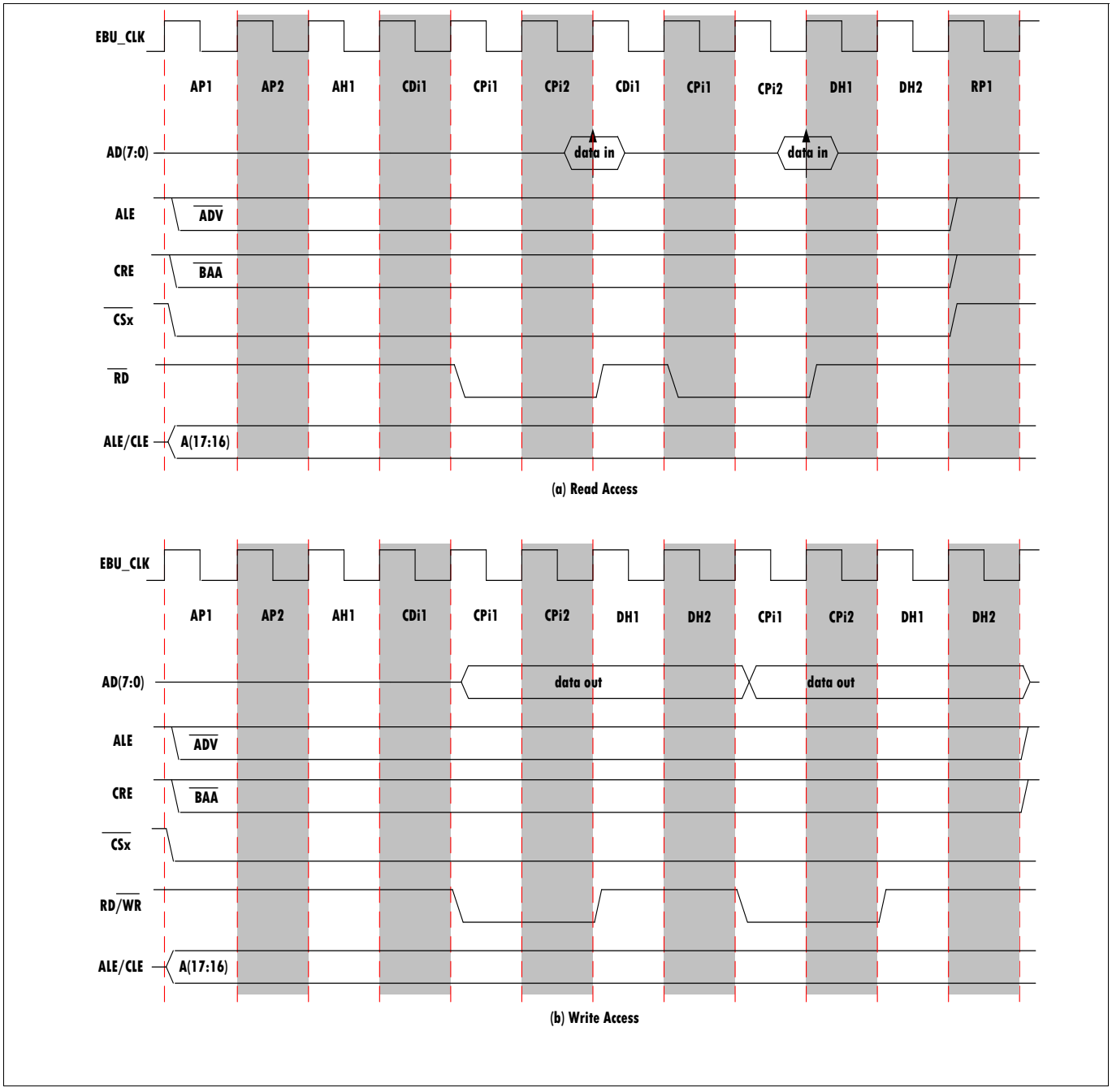


Figure 739 NAND Flash Page Mode Accesses

Example Nand Flash Read Sequence

Figure 740 shows an example of how the processor can generate a Nand Flash read access sequence given this configuration:-

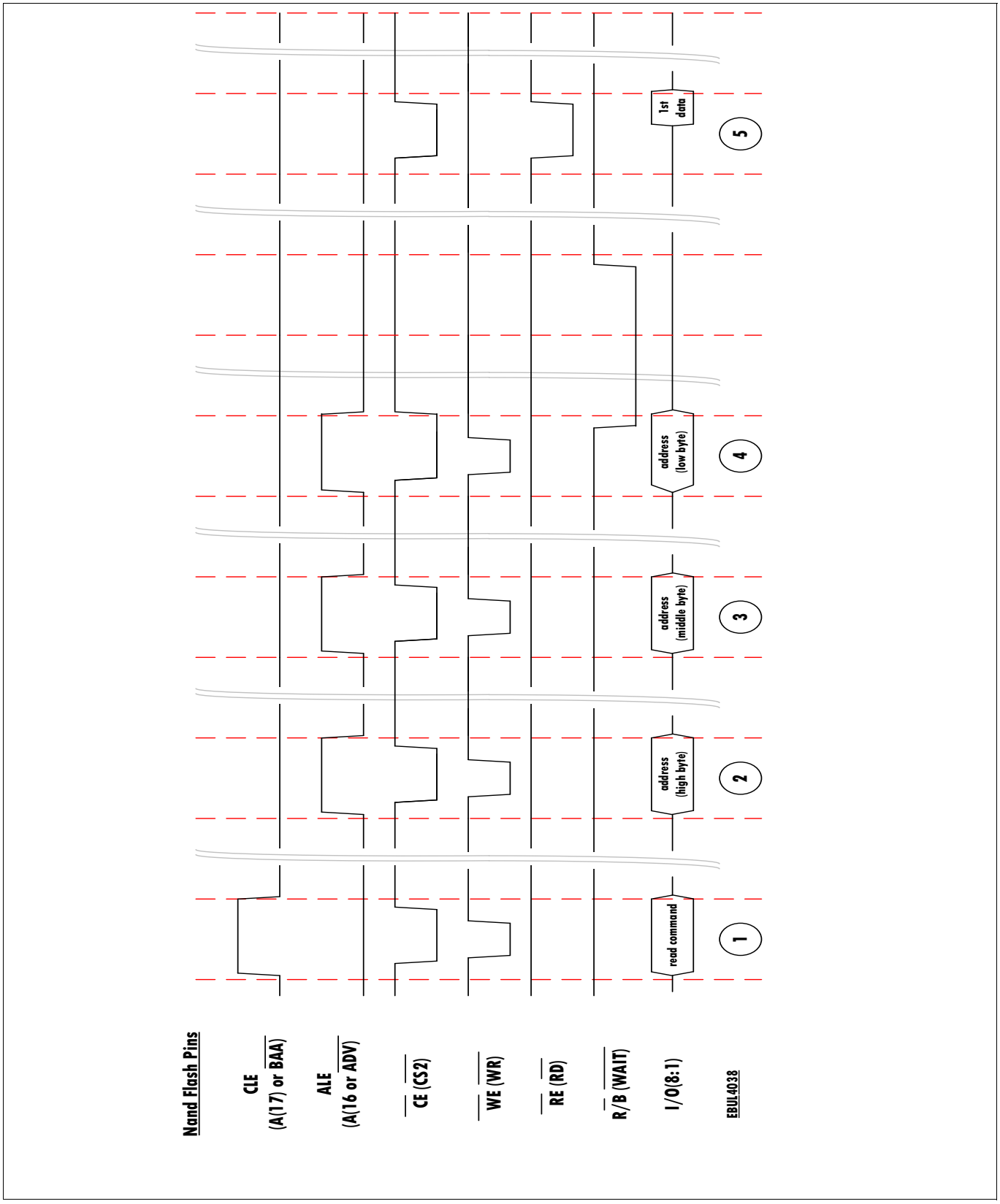


Figure 740 Example of an Memory Controller Nand Flash access sequence (read)

1. In the cycle marked ‘1’ in **Figure 740** the processor initiates a read sequence by writing the “Read Command” value to address “NAND_FLASH_BASE + 0x20000”. This generates a write sequence with CLE (A(17)) driven high and ALE (A(16)) driven low.
2. In the cycle marked ‘2’ the processor loads the most significant byte of the read address by writing to address “NAND_FLASH_BASE + 0x10000”. This generates a write sequence with CLE (A(17)) driven low and ALE (A(16)) driven high.
3. In the cycle marked ‘3’ the processor loads the middle significant byte of the read address by repeating the access specified in ‘2’ above.
4. In the cycle marked ‘4’ the processor loads the least significant byte of the read address by repeating the access specified in ‘2’ above. The Nand Flash responds to this final address byte by driving it’s R/$\overline{B}$ output low. The processor monitors this pin (using the EBU_MODCON.sts bit) until the Nand Flash has completed it’s internal data fetch.
5. In the cycle marked ‘5’ the processor reads the first byte of data by reading address “NAND_FLASH_BASE + 0x00000”. The processor can subsequently read any additionally required (sequential) data bytes by repeating cycle ‘5’.

Note: A similar scheme can be used to generate write access sequences.

45.3.14 Synchronous Read/Write Accesses

The Memory Controller is designed to generate waveforms compatible with the burst modes of:

1. INTEL and compatible burst flash devices
2. SPANSION and compatible burst flash devices
3. INFINEON and MICRON cellular RAM
4. Fujitsu and Compatible FCRAM/uTRAM/CosmoRAM
5. Samsung OneNAND burst capable NAND flash and compatible devices
6. M-Systems DiskOnchipG3 and compatible devices
7. GSI/ISSI/IDT SSRAM

Note: Not all of the supported synchronous memory types are known to be available in automotive grade or with I/O voltages compatible with the TC29x I/O pads.

Features

The Synchronous Access Controller is primarily designed to perform burst mode read and write cycles for an external instruction memories, external Cellular RAM and FCRAM data memories. In general, the features are:-

- Fully synchronous timing with flexible programmable timing parameters (address cycles, read wait cycles, data cycles).
- Programmable $\overline{WAIT}$ function.
- Programmable burst (mode and length)
- 8-bit device width.
- 16-bit device width.
- 32-bit device width
- Page mode read accesses.
- Resynchronisation of read data to a feedback clock to maximise the frequency of operation (optional).

45.3.14.1 Signals

The following signals are used for the Burst FLASH interface:-

Table 524 Burst Flash Signal List

Signal	Type	Function
	I/O	Multiplexed Address/Data bus
$\overline{RD}$	O	Read control
$\overline{WR}$	O	Write control
A(:0)	O	Address bus
$\overline{ADV}$	O	Address valid strobe
$\overline{WAIT}$	I	Wait/terminate burst control
$\overline{CS}$	O	Chip select
BFCLKO	O	Burst FLASH Clock, running equal to, 1/2, 1/3 or 1/4 of the frequency of the EBU core logic.
BFCLKI	I	Burst FLASH Clock Feedback.
$\overline{BAA}$	O	Advance Burst Address
STS	I	Burst FLASH Status Input (Optional, value of $\overline{WAIT}$ pin available through status register)

45.3.14.2 Support for Burst FLASH device types

Support is provided for a maximum of different Burst FLASH configurations on the external bus - i.e. one on each external chip select.

Bit-fields EBU_BUSCONx.EBSE, EBU_BUSCONx.ECSE, EBU_BUSCONx.wait, EBU_BUSCONx.FBBMSEL, EBU_BUSCONx.BFCMSEL and EBU_BUSCONx.FETBLEN are used to configure specific characteristics for burst access cycles.

45.3.14.3 Typical Burst Flash Connection

The figure below shows a typical burst flash connection.

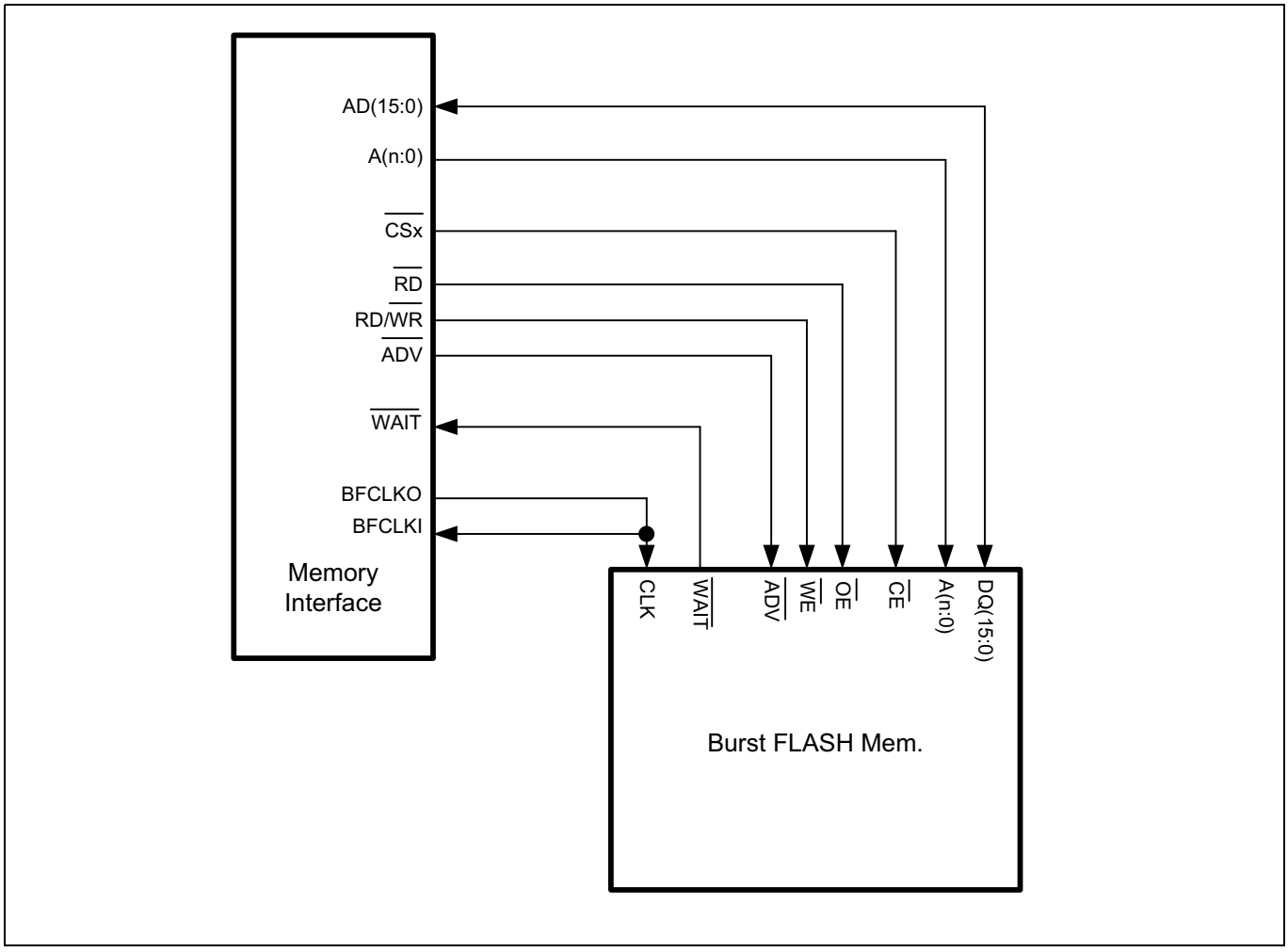


Figure 741 Typical Burst Flash Connection

45.3.14.4 Standard Access Phases

Accesses to burst FLASH devices are composed of a number of “Standard Access Phases” (which are detailed in **Section 45.3.12**). The Standard Access Phases for Burst FLASH devices are:-

- AP: Address Phase (compulsory - see **“Address Phase (AP)” on Page 24**).
- AH: Address Hold Phase (optional see **“Address Hold Phase (AH)” on Page 25**).
- CD: Command Delay Phase (optional - see **“Command Delay Phase (CD)” on Page 25**).
- CP: Command Phase (optional - see **“Command Phase (CP)” on Page 26**).
- BP: Burst Phase (compulsory - see **“Burst Phase (BP)” on Page 26**).
- RP: Recovery Phase (optional - see **“Recovery Phase (RP)” on Page 27**).

Note: During a burst access the Burst Phase (BP) is repeated the required number of times to complete the burst length.

See **Figure 742** for an example synchronous access to a non-multiplexed device and **Figure 743** for an example synchronous access to a multiplexed device.

45.3.14.5 Example Waveforms

The following figures show example waveforms for synchronous accesses

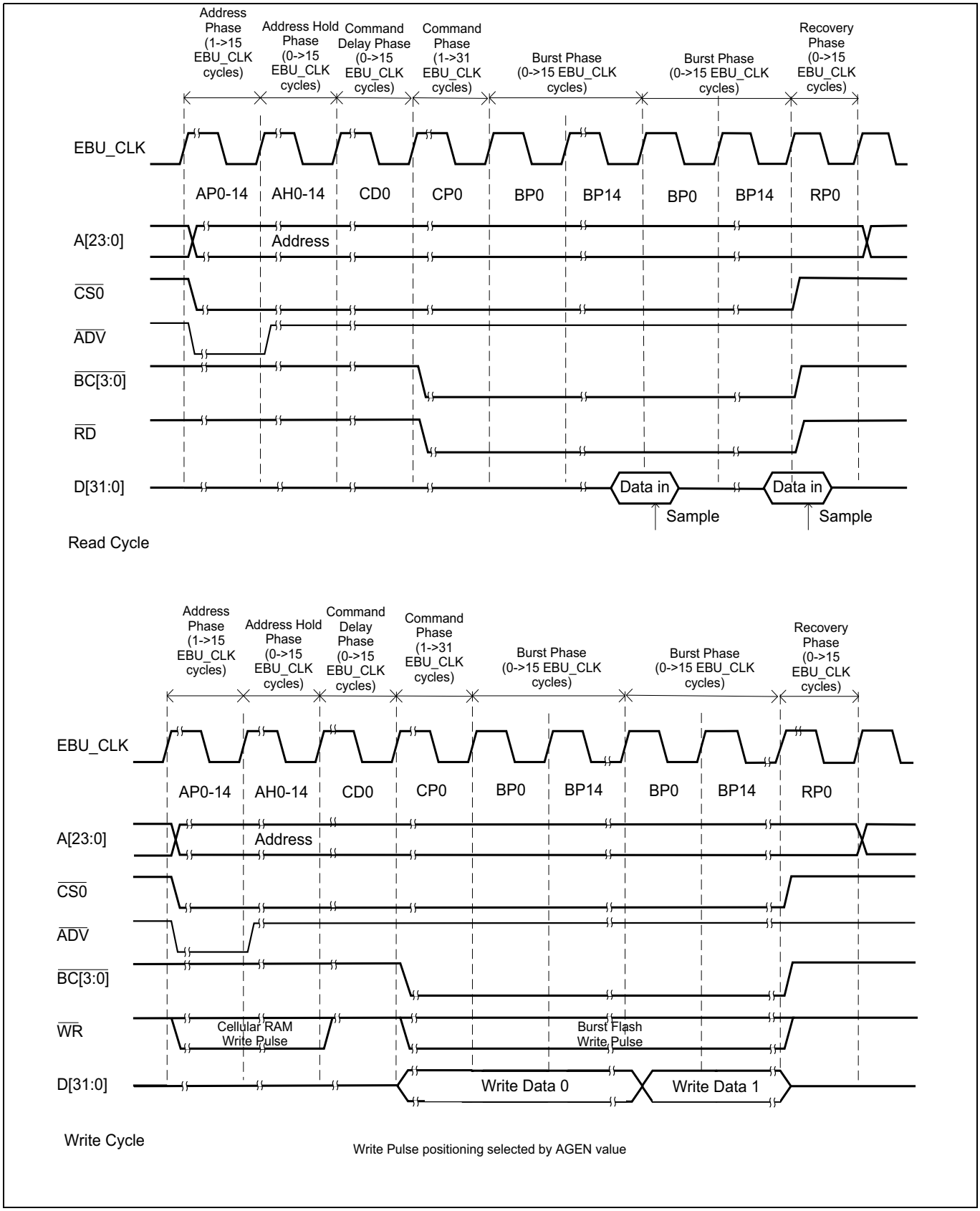


Figure 742 Synchronous non-muxed Access

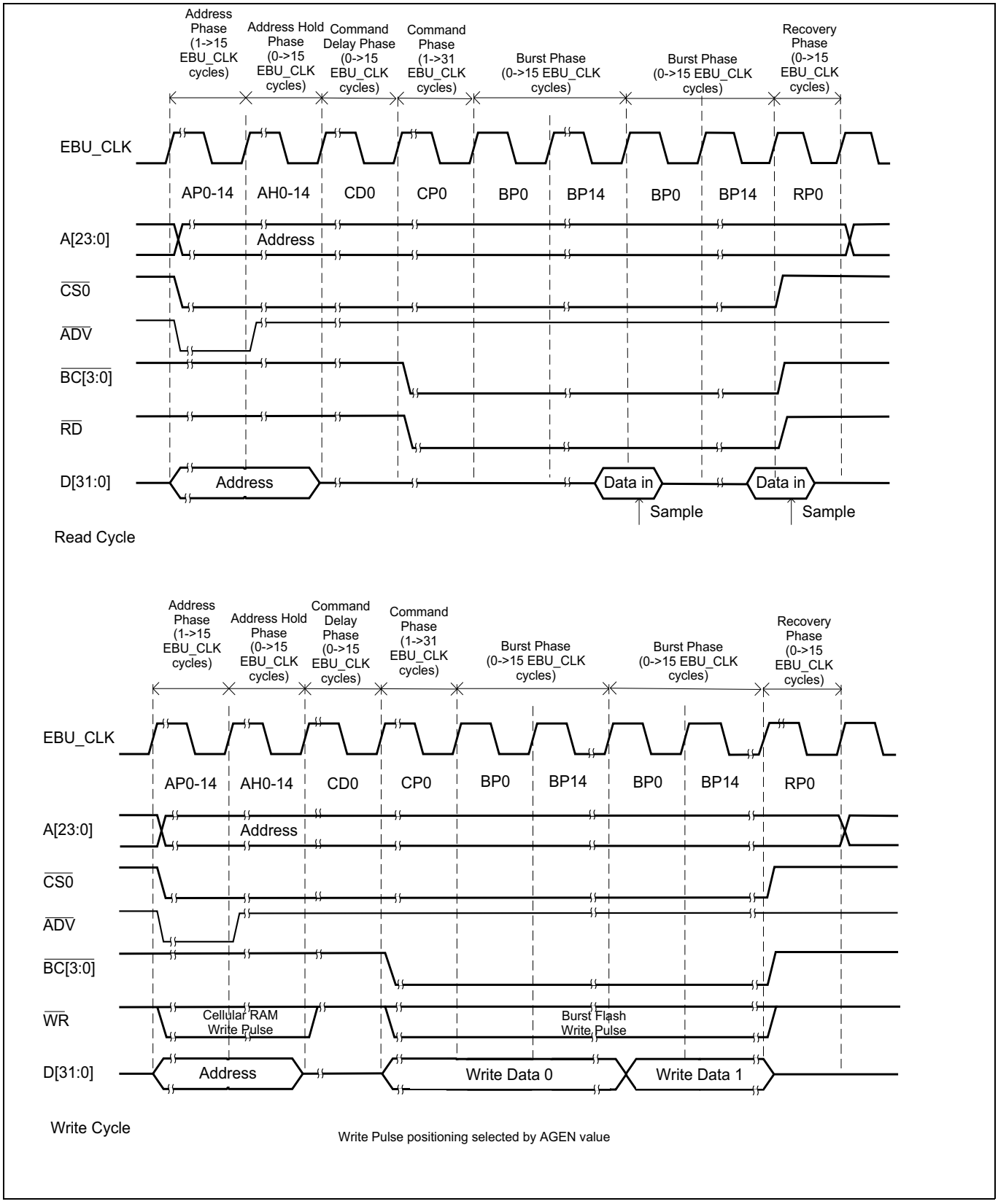


Figure 743 Synchronous muxed Access

45.3.14.6 Burst Length Control

The maximum number of valid data samples that can be generated by a flash device in a single read access is set by the EBU_BUSCONx.FBBMSEL bit and the EBU_BUSCONx.FETBLEN bit field.

The EBU_BUSCONx.FBBMSEL bit is used to select Continuous Burst Mode where there is no limit to the number of data samples in a burst read access.

The EBU_BUSCONx.FBBMSEL and EBU_BUSCONx.FETBLEN bit-fields are used to select the maximum number of data samples in a single access. Where an SRI request exceeds the amount of data that can be fetched or stored by the programmed number of data samples, the EBU will automatically generate the appropriate number of burst accesses to transfer the required amount of data.

Note: Selection of Continuous Burst Mode (by use of the 'FBBMSEL' bit) overrides the maximum burst setting (specified by the FETBLEN bit-field).

45.3.14.7 Burst Flash Clock

Since the can run too fast for clocking Burst FLASH devices, the Memory Controller provides an additional clock source (BFCLKO). This signal is generated by a programmable clock divider driven by and allows to BFCLKO ratios of 1:1, 2:1, 3:1 and 4:1 to be selected. The frequency of the signal is determined by bit-field EBU_BUSRP.EXTCLOCK. Note that it is possible to set a different clock rate for synchronous writes to the same device by programming EBU_BUSWP.EXTCLOCK to a different value.

If a continuously running BFCLKO is required, then the BUSRCONx.BFCMSEL field can be used to enable an ungated flash clock. This bit is normally set to 1_B in all the BUSRCONx registers after reset. If cleared, the related BUSRAPx.EXTCLOCK field will be used to generate a stable BFCLKO. If multiple BUSRCONx.BFCMSEL fields are set to 0_B, then the highest priority (lowest index) BUSRAPx.EXTCLOCK field will be used.

During a burst access to a synchronous device, BFCLKO will generate correctly aligned clock edges as shown in **Figure 744**. The BFCLKO signal is gated to ensure that it is low (zero) at all other times (including asynchronous read/writes of/to synchronous devices). This provides power savings and ensures correct asynchronous accesses to Burst FLASH device(s).

The start of the address and burst phases are synchronised, by hardware, to the rising edge of BFCLKO. Exiting from a phase extended by the WAIT input will also be synchronised to the rising edge of BFCLKO.

Note: The length of the standard accesses phases during Burst FLASH accesses are programmed as a multiple of independent of the BFCLKO frequency. It is the users responsibility to program the access phases to ensure that the sampling of data by Memory Controller guarantees valid sampling of the data from the Burst FLASH device.

The EBU uses the clock to generate all external bus access sequences.

Table 525 EXTCLOCK to clock ratio mapping

EXTCLOCK value	BFCLKO divide ratio
00	1:1
01	1:2
10	1:3
11	1:4

Unless documented elsewhere, all outputs to the external bus are generated of the rising edge of .

The BFCLKO phase is controlled so that control signal changes will normally occur at the rising edge of BFCLKO unless configured otherwise by register settings.

45.3.14.8 Control of $\overline{ADV}$ & Control Signal Delays During Synchronous Accesses

The Memory Controller output signals: $\overline{ADV}$, $\overline{CS}$, $\overline{RD}$, RD/$\overline{WR}$, $\overline{BC}$ and AD signals can be delayed with respect to the BFCLKO clock signal.

The delays can be enabled and disabled by the register bits EBU_BUSCONx.EBSE for the $\overline{ADV}$ signal and by EBU_BUSCON.ECSE for the $\overline{CS}$, $\overline{RD}$, RD/$\overline{WR}$, $\overline{BAA}$ and write data signals

The amount by which the signal is delayed depends on the ratio of to the Burst FLASH clock as follows:-

- When the ratio of to BFCLKO is 1:1, signals are asserted on the negative edge of . i.e. it is in effect delayed by an EBU_CLK high pulse width (T_{PH}) with respect to BFCLKO.
- When the ratio of to BFCLKO is 1:2 or 1:3, control signals are asserted on the next positive edge of . i.e. it is in effect delayed by an cycle (T_{CLK}) with respect to BFCLKO.
- When the ratio of to BFCLKO is 1:4, control signals are asserted on the negative edge of BFCLKO. i.e. it is in effect delayed by two cycles ($2*T_{CLK}$) with respect to BFCLKO.
- If the access is to a Cellular RAM (BUSWCONx.AGEN=3_D or 7_D), then the RD/$\overline{WR}$ signal is treated as a qualifier to the address and its timing will therefore not be affected by the setting of EBU.BUSWCON.ECSE.

For read accesses the EXTCLOCK field used will be from the BUSRAPx register of the region being accessed. For write accesses the EXTCLOCK field used will be from the BUSWAPx register of the region being accessed. However if a continuous BFCLKO is being generated (BUSRCONx.BFCMSEL=0_B), for all accesses, the EXTCLOCK value from the register with BFCMSEL set to 0_B will be used when calculating delays. This ensures that the signals are always delayed correclty relative to the clock on the BFLCKO output.

The default setting after reset has the delays disabled.

If the delay is disabled, then the signals will not be delayed in 1:1 mode (except for ADV which will be guaranteed to be after the edge of BFCLKO). In 2:1, 3:1 and 4:1 mode, the signals will be delayed by an EBU_CLK high pulse width (T_{PH}) from the start of the cycle in which they are asserted.

Table 526 $\overline{ADV}$ Signal Timing

EXTCLOCK is set to	$\overline{ADV}$ Falling Edge position		$\overline{ADV}$ Rising Edge position	
	Delay Disabled[1]	Delay Enabled	Delay Disabled[1]	Delay Enabled
00_B	Start of AP1	Start of AP1+T_{PH}	End of APn	End of APn+T_{PH}
01_B, 10_B	Start of AP1+T_{PH}	End of AP1	End of APn+T_{PH}	End of APn+ T_{CLK}
11_B	Start of AP1+T_{PH}	End of AP1 + T_{CLK}	End of APn+T_{PH}	End of APn + $2*T_{CLK}$

1) See **Figure 742** for details of this signal positioning.

Table 527 $\overline{RD}$ and RD/$\overline{WR}$ Signal Timing

EXTCLOCK is set to	Set at:		Cleared at:	
	Delay Disabled[1]	Delay Enabled	Delay Disabled	Delay Enabled
00_B	Start of CP1	Start of CP1 + T_{PH}	End of CPn[2]	End of CPn + T_{PH}
01_B, 10_B	Start of CP1 + T_{PH}	End of CP1	End of CPn + T_{PH}	End of CPn + T_{CLK}
11_B	Start of CP1 + T_{PH}	End of CP1 + T_{CLK}	End of CPn + T_{PH}	End of CPn + $2*T_{CLK}$

1) See **Figure 742** for details of this signal positioning.

2) CPn indicates the final Command Phase.

Table 528 $\overline{CS}$ Data Signal Timing

EXTCLOCK is set to	Set at:		Cleared at:	
	Delay Disabled[1]	Delay Enabled	Delay Disabled	Delay Enabled
00_B	Start of AP1	Start of AP1 + T_{PH}	End of DHn[2]	End of DHn + T_{PH}
01_B, 10_B	Start of AP1 + T_{PH}	End of AP1	End of DHn + T_{PH}	End of DHn + T_{CLK}
11_B	Start of AP1 + T_{PH}	End of AP1 + T_{CLK}	End of DHn + T_{PH}	End of DHn + 2*T_{CLK}

1) See **Figure 742** for details of this signal positioning.

2) DHn indicates the final Data Hold Phase. This is replaced by CPn if the programmed Data Hold phase length is zero clocks.

The byte control signals, $\overline{BC}$x, can either use the timing in **Table 527** if EBU_BUSCONx.BCGEN is set to 01_B or 10_B or the timing in **Table 528** if EBU_BUSCONx.BCGEN is set to 00_B. A EBU_BUSCONx.BCGEN value of 11_B is not valid for synchronous accesses.

Table 529 Write Data Signal Timing

EXTCLOCK is set to	Data Driven at:		Data Cleared at:		Data Changes at:	
	Delay Disabled[1]	Delay Enabled	Delay Disabled	Delay Enabled	Delay Disabled	Delay Enabled
00_B	Start of CP1	Start of CP1 + T_{PH}	End of DHn[2]	End of DHn + T_{PH}	Start of BP1	Start of BP1 + T_{PH}
01_B, 10_B	Start of CP1 + T_{PH}[3]	End of CP1[3]	End of DHn + T_{PH}	End of DHn + T_{CLK}	Start of BP1 + T_{PH}	End of BP1
11_B	Start of CP1 + T_{PH}[3]	End of CP1 + T_{CLK}[3]	End of DHn + T_{PH}	End of DHn + 2*T_{CLK}	Start of BP1 + T_{PH}	End of BP1 + T_{CLK}

1) See **Figure 742** for details of this signal positioning.

2) DHn indicates the final Data Hold Phase. This is replaced by CPn if the programmed Data Hold phase length is zero clocks.

3) Data bus will be enabled at the start of CP1

These delay options apply to write data only. Addresses which are output on the data bus to support devices with multiplexed address and data connections are not delayed.

Note: If the control signals are delayed a recovery phase must be used to prevent conflicts between accesses as the rising edge of the control signals will be delayed past the end of the command phase. If a multiplexed access is used without a recovery phase, the address for the next access will be delayed by one clock cycle to enforce a bus turnaround time on the data bus, resulting in the valid address being driven one clock after $\overline{ADV}$ is asserted.

45.3.14.9 Burst Flash Clock Feedback

The Memory Controller can be configured to use clock feedback to optimise the operating frequency for a given flash device. This is enabled by setting the EBU_BUSCONx.FDBKEN bit to one. With this bit enabled the first sampling stage for read data has its own clock (PD_BFCLKFEEDBK_I). This will be derived from the BFCLKO output by using a second pad (BFCLKI) to monitor the BFCLKO signal after the output pad delay.

Clock feedback should be used whenever possible as it allows the best possible performance

A side effect of using this mode is an increase in data latency by one BFCLKO cycle compared with not using clock feedback.

Note: Clock feedback will be automatically disabled for burst writes as the additional latency on the $\overline{WAIT}$ input would prevent correct operation of the memory controller.

45.3.14.10Asynchronous Address Phase

As operating frequency increases, it becomes increasingly hard to avoid violating some timing parameters. The asynchronous address phase allows the address to be latched into the flash memory using the ADV signal before the clock is enabled. This is only possible if explicitly allowed by the flash data sheet.

If the EBU_BUSCONx.AAP is set, then the clock will not start until the end of the address hold phase of the access. The rising edge of the clock will always be co-incident with the transition from the address hold phase. If the address hold phase has zero length then the first rising edge of the clock will coincide with the tranistion from the address phase. If this mode is enabled a recovery phase of one BFCLK period will be enforced at the end of the previous transaction to ensure that the clock has time to turn off before the start of the next access.

Setting this mode for any region will force the clocks on all synchronous accesses to be disabled in the recovery phase. Only one clock pulse will occur during the recovery phase.

AAP mode is incompatible with the continuous clock mode and will be disabled automatically if continuous clocking is enabled by setting any BUSRCONx.BFCMSEL bit to 0_B.

45.3.14.11 Critical Word First Read Accesses

In the default case, the memory controller will always start a burst at the lowest address possible and wrapping of the burst data is handled internally to the memory controller. However, some burst devices implement a wrapping feature which is compatible with the wrapped bursts used by the processor cache fill requests. If this is the case, there is an advantage to using the wrapping mode in the device as the instruction required by the processor (the critical word) can be fetched first from the external memory.

The mode is enabled by setting the EBU_BUSCONx.dba bit for the appropriate region. Once enabled, the memory controller will not align the start address for the burst and the device will be relied on to return data in the correct order. The memory controller must fetch all the data in a single burst. If the transaction is split into multiple accesses on the external bus by use of the FETBLEN field, the issued addresses will be incorrect.

Note: The cache line fill will use an SRI, BTR4 transfer. This translates to a 16 word burst for a 16 bit device. The device must therefore support a 16 word wrap setting. A 32 bit memory must support a 8 word wrap setting. The memory controller supports a 16 word burst using the continuous burst setting for FBBMSEL. Other burst opcodes must not be generated for accesses to the external memory by the system if this mode is enabled, otherwise data corruption will occur.

45.3.14.12Example Burst Flash Access Cycle

The figure below shows an example burst flash access without clock feedback configured.

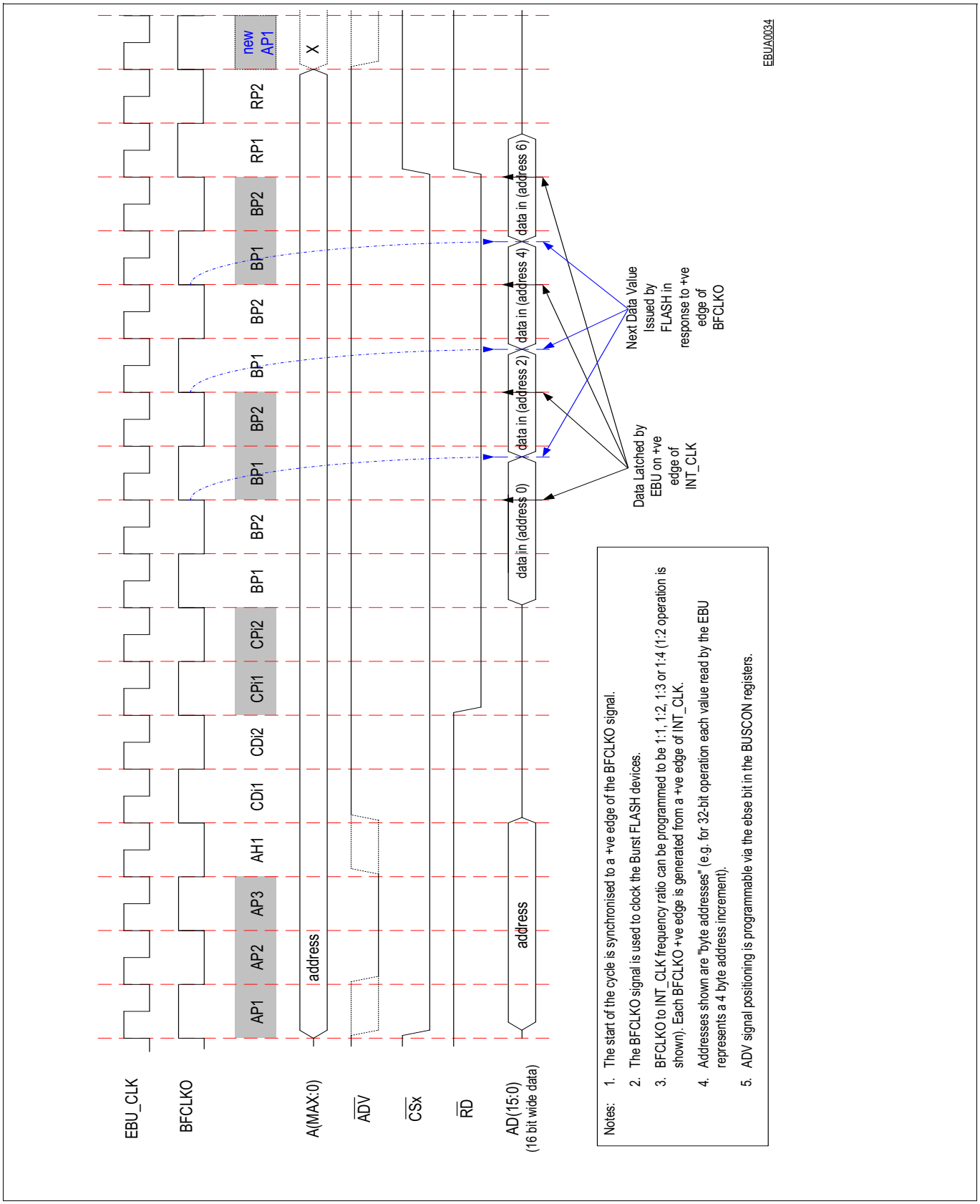


Figure 744 Burst FLASH Read without Clock Feedback (burst length of 4)

Figure 744 shows an example of a burst read access (burst length of four) to a Burst FLASH device with $\overline{\text{WAIT}}$ and clock feedback functions disabled.

Programmability of the length of the Address, Command Delay and Command phases allows flexible configuration to meet the initial read access time of a Burst FLASH device.

Data is sampled at the end of each Burst Phase cycle. The Burst Phase is repeated the appropriate number of times for the programmed burst length (programmable for lengths of 1, 2, 4 or 8 via the EBU_BUSCONx.FETBLEN bit-field).

Figure 744 shows an access cycle with the following settings:-

- Clock Feedback disabled.
- Address Phase length = 3 cycles (see ADDRC and **“Address Phase (AP)” on Page 24**).
- Command Delay Phase length = 3 cycles (see CMDDELAY and **“Command Delay Phase (CD)” on Page 25**).
- Command Phase length = 2 cycles (see WAITRDC and **“Command Phase (CP)” on Page 26**).
- Burst Phase length = 2 cycles (see EXTCLOCK, EXTDATA and **“Burst Phase (BP)” on Page 26**).
- Recovery Phase length = 2 cycles (see **“Recovery Phase (RP)” on Page 27**).
- Burst Length = 4 (see FETBLEN).
- BFCLKO frequency = 1/2 of frequency (see EXTCLOCK).

45.3.14.13External Cycle Control via the $\overline{\text{WAIT}}$ Input

Memory Controller provides control of the Burst FLASH device via the $\overline{\text{WAIT}}$ input. This allows Memory Controller to support operation of Burst FLASH while crossing Burst FLASH page boundaries. During a Burst FLASH access the $\overline{\text{WAIT}}$ input operates in one of four modes:-

- Disabled
- Early Wait for Page Load.
- Wait for Page Load.
- Abort and Retry Access.

Selection of the mode in which the $\overline{\text{WAIT}}$ input operates during Burst FLASH reads is selected via the EBU_BUSCONx.WAIT bits.

Table 530 Operation of WAIT input

Value of BUSCONx.WAIT	Mode of the $\overline{\text{WAIT}}$ input
0_D	OFF (default after reset).
1_D	Wait for page load (Early WAIT).
2_D	Wait for page load (WAIT with data).
3_D	reserved

Note: Selection of “Disabled” via the wait bit-field prevents the $\overline{\text{WAIT}}$ input having any effect on a Burst FLASH access cycle

Wait for Page Load Mode

This mode supports devices which assert a $\overline{WAIT}$ output for the duration of clock cycles in which the data output by the device is invalid or, alternatively, one clock cycle earlier than the data output is invalid. This includes Intel and AMD Burst FLASH devices (and compatibles) configured for Early Wait Generation Mode (EBU_BUSCONX.wait=01_B) and standard wait generation (EBU_BUSCONx.wait=10_B).

In operation, the burst flash controller loads a counter with the required number of samples at the start of each burst. At the end of each burst phase, the burst flash controller samples the $\overline{WAIT}$ input and the data bus at the end of each Burst phase. If $\overline{WAIT}$ is inactive, the sample is valid, the sample counter is decremented and the sampled data is passed to the datapath of the Memory Controller. This synchronous sampling means that the validity of the sample can not be determined until the clock cycle after the end of the burst phase. The Burst Flash controller will therefore overrun and generate extra burst phases until the sample counter is decremented to zero. Extra data samples returned after the sample counter is zero will be discarded.

The only difference if early wait is used is that the validity of data in burst phase "n" is determined by the value of $\overline{WAIT}$ in burst phase "n-1".

This mode of operation is compatible with the use of clock feedback as, with feedback enabled, $\overline{WAIT}$ is fed through the same resynchronisation signals as the data bus. The only effect on operation is that the number of overrun cycles will increase as the decrementing of the sample counter will be lagged by the resynchronisation stages.

During the initial phases of an access, $\overline{WAIT}$ is sampled on every edge of . This is so the first burst phase is working with an accurate value for the $\overline{WAIT}$ signal. To ensure this is the case, the command phase should be of sufficient length to allow the device to drive $\overline{WAIT}$ and for the signal to propagate to the controller.

Abort and Retry Access

In this mode, the $\overline{WAIT}$ input is polled during the address and address hold phases only. If an active state on $\overline{WAIT}$ is detected, the address phase of the access will be restarted after the end of the address hold phase.

In this mode $\overline{WAIT}$ cannot be used to extend the command phase of an access.

45.3.14.14Flash Non-Array Access Support

Several types of flash memories will assert $\overline{WAIT}$ permanently during an access which is not directed to the memory array. An example of this would be polling the status register to check if a programming operation has completed. If the BUSRCON[3:0].NAA field is set, then an access to the region with SRI A(26) set will proceed as if the appropriate wait field in BUSRCON[3:0] or BUSWCON[3:0] was set to 00_B and $\overline{WAIT}$ was disabled. When set, this field affects both read and write accesses.

45.3.14.15Termination of a Burst Access

A burst read operation is terminated by de-asserting $\overline{CSx}$ signal followed by the appropriate length Recovery Phase. **Figure 745** shows an example of termination of a burst access following the read of two locations (i.e. two Burst Phases) from a 16-bit non-multiplexed Burst FLASH device.

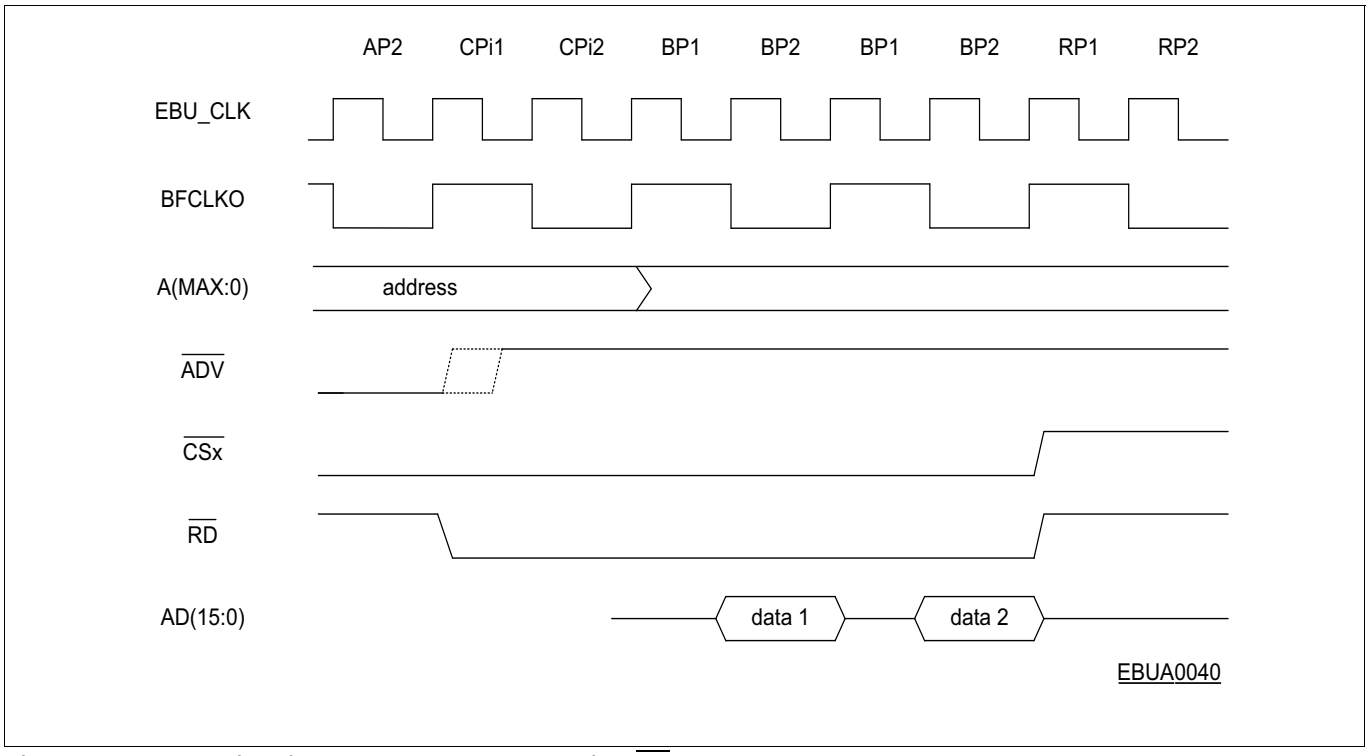


Figure 745 Terminating a Burst by de-asserting $\overline{CS}$

45.3.14.16Burst Flash Device Programming Sequences

Programming sequences for some Burst Flash devices must not be interrupted by other read/write operations to the same device. There is an optional hardware lock feature to guarantee that programming sequences are not interrupted. See **Section 45.3.6.1** for details.

45.3.14.17Cellular RAM

Cellular RAM devices have been designed to meet the growing memory and bandwidth demands of modern cellular phone designs. The devices have been designed with a “multi-protocol” interface to allow use of the devices with existing memory interfaces (i.e. by re-use of existing memory protocols). The supported interface protocols supported by Cellular RAM devices are:-

1. SRAM (Asynchronous Read and Write).
2. NOR Flash (Synchronous Burst Read, Asynchronous Write).
3. Synchronous (Synchronous Burst Read and Write).

In principle, when using previous versions of Memory Controller, the first two of the above modes (1 and 2 above) provided Cellular RAM support. For maximum performance, the Memory Controller now supports Synchronous Mode (3 above) for Cellular RAM (Synchronous Burst Read and Write).

As Cellular RAM Synchronous Mode consists of a Burst FLASH compatible Burst Read access, Cellular RAM support has been provided by enhancing the Burst FLASH interface by the inclusion of a Burst Write capability. For this reason Cellular RAM is treated as a special type of Burst FLASH device.

Cellular RAM support is selected by programming the desired region as cellular RAM via the EBU_BUSCONx.AGEN bit-field.

Synchronous Read Access

A Synchronous Cellular RAM Burst Read Access is compatible with a Burst FLASH Burst Read Access. As a result preceding sections applying to Burst FLASH devices apply and should be consulted for details of Cellular RAM Burst Read Accesses.

45.3.14.17.1 Synchronous Write Access

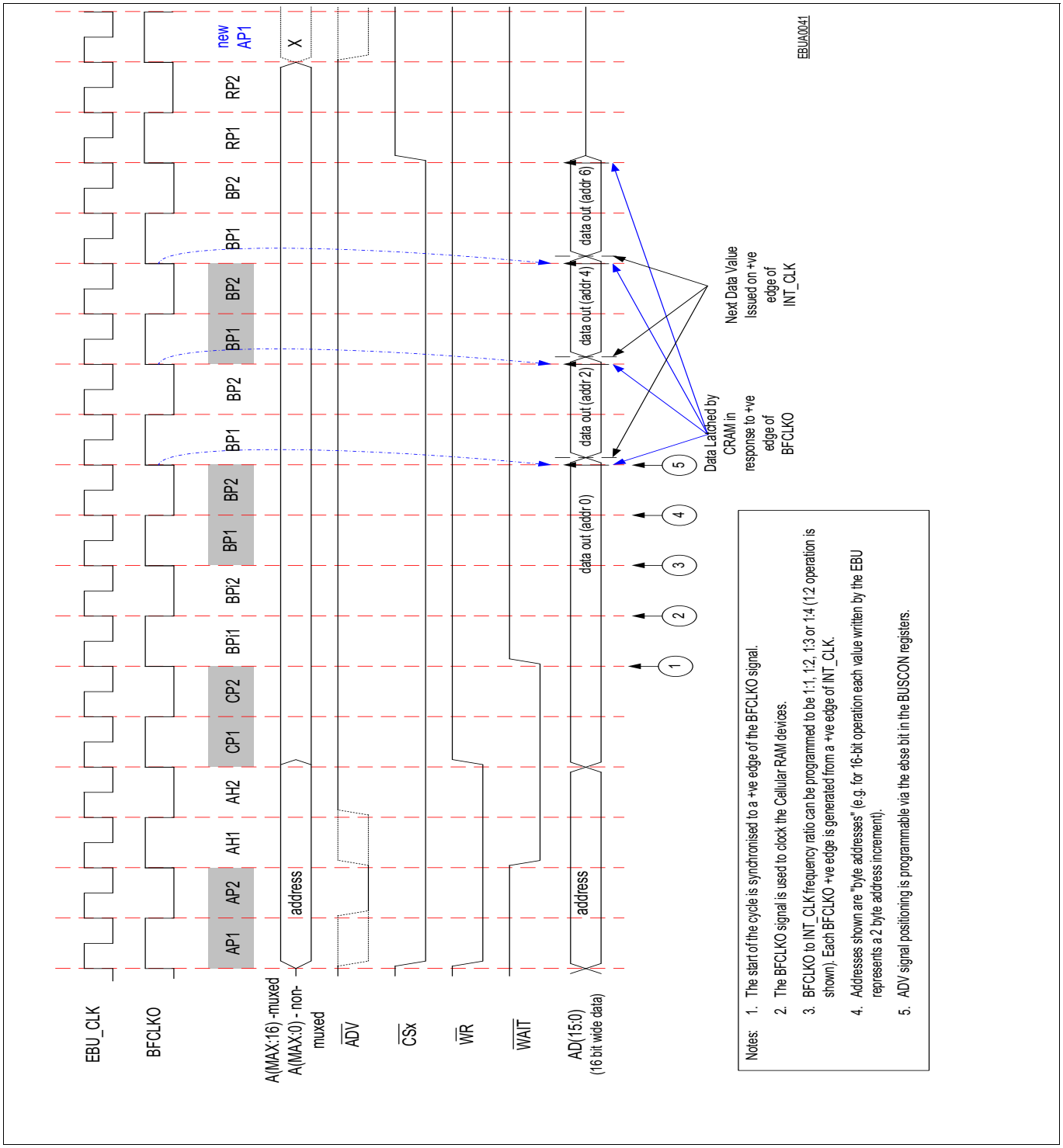


Figure 746 Burst Cellular RAM Burst Write Access (burst length of 4)

Figure 746 shows an example of a Cellular RAM burst write access.

*Note: **Figure 746** shows operation with a BFCLKO to ratio of 1:2.*

The Start of the access cycle is the same as for a Synchronous Read access (see **Figure 744**) except that the $\overline{WR}$ signal is treated as an address phase signal (i.e. it is asserted active during the Address Phase and Address Hold Phase and is then deasserted). See **"Fujitsu FCRAM Support (burst write with WR active during data phase)" on Page 56** for alternative $\overline{WR}$ timing during burst write.

The remaining sequence is as follows (with reference to the figure above):-

1. At the positive edge of labelled as '1' above the first Burst Phase starts. As the state machine is currently in the command phase, the interface samples the $\overline{\text{WAIT}}$ input. This is sampled as "active". By coincidence, in this example, the Cellular RAM also deasserts it's $\overline{\text{WAIT}}$ output as a response to this clock edge to signal that it will start to take the data from the data bus on the BFCLKO rising clock edge after the next (i.e. the rising edge of BFCLKO labelled as '5' above) - this need not be the case.
2. At the positive edge of labelled as '2' above the second programmed period of the Burst Phase begins.
3. At the positive edge of labelled as '3' above the Burst FLASH evaluates the $\overline{\text{WAIT}}$ sample from '1' above. As this sample was "active" the write data is not updated. As this clock edge is coincident with the end of a burst phase the $\overline{\text{WAIT}}$ input is resampled. The value of this new $\overline{\text{WAIT}}$ sample is "inactive".
4. At the positive edge of labelled as '5' above the Burst FLASH again evaluates the $\overline{\text{WAIT}}$ sample from '3' above. As this sample was "in-active", and the edge is coincident with the end of a burst phase, the next data value is issued to the pins and the next Burst Phase is started.

This process continues until all the data is written.

45.3.14.17.2 Fujitsu FCRAM Support (burst write with $\overline{\text{WR}}$ active during data phase)

The FCRAM device type can be supported in two ways. Later FCRAMs have a compatibility bit in the device configuration register which programmes the device to expect the $\overline{\text{WR}}$ signal to be active with the address and to be latched with the $\overline{\text{ADV}}$ signal. In this mode, FCRAM can be treated as an Infineon/Micron cellular RAM.

Alternatively, if a write is attempted to a region configured as a burst flash, the memory controller will generate a burst write with the $\overline{\text{WR}}$ signal asserted with the write data. This should be directly compatible with an FCRAM operating in its native mode.

45.3.14.18Programmable Parameters

The following table lists the programmable parameters for burst flash accesses. These parameters only apply when the EBU_BUSCONx.agen parameter for a particular memory region is set for access to synchronous burst devices (page mode or otherwise).

Table 531 Burst Flash Access Programmable Parameters

Parameter	Function	Register
ADDRC	Number of cycles in Address Phase.	EBU_BUSAPx
AHOLDC	Number of cycles in Address Hold.	EBU_BUSAPx
CMDDELAY	Number of programmed Command Delay cycles.	EBU_BUSAPx
WAITRDC	Number of programmed wait states for read accesses.	EBU_BUSAPx
WAITWRC	Number of programmed wait states for write accesses.	EBU_BUSWAPx
EXTDATA	Extended data	EBU_BUSAPx
RDRECOVC	Number of minimum recovery cycles after a read access when the next access is to the same region.	EBU_BUSRAPx
WRRECOVC	Number of minimum recovery cycles after a write access when the next access is to the same region.	EBU_BUSWAPx
RDDTACS	Number of minimum recovery cycles after a read access when the next access is to a different region.	EBU_BUSRAPx
WRDTACS	Number of minimum recovery cycles after a write access when the next access is to a different region.	EBU_BUSWAPx

Table 531 Burst Flash Access Programmable Parameters (cont'd)

Parameter	Function	Register
WAIT	Sampling of $\overline{\text{WAIT}}$ input: OFF, SYNCHRONOUS, ASYNCHRONOUS or WAIT_CELLULAR_RAM	EBU_BUSCONx
FBBMSEL	Flash synchronous burst mode: CONTINUOUS or DEFINED (as in FETBLEN)	EBU_BUSCONX
FETBLEN	Synchronous burst length: SINGLE, BURST2, BURST4 or BURST8	EBU_BUSCONx
BFCMSEL	Flash Clock Mode, continuous or gated	EBU_BUSRCONx
EXTCLOCK	Frequency of external clock at pin BFCLKO: equal, 1/2, 1/3 or 1/4 of	EBU_BUSAPx
EBSE	delay $\overline{\text{ADV}}$ output to improve hold margin	EBU_BUSCONx
ECSE	delay $\overline{\text{CS}}$, $\overline{\text{WR}}$ and write data outputs to improve hold margin	EBU_BUSCONx
LOCKCS	enable locked write sequences for this region	EBU_BUSWCONx
FDBKEN	enable clock feedback to improve read data margins	EBU_BUSRCONx
DBA	disable alignment of read bursts on external bus	EBU_BUSRCONx
AAP	enable the "asynchronous address phase" mode.	EBU_BUSCONx
PORTW	memory port width	EBU_BUSRCONx

Note: datac is not used for burst write accesses

45.4 Registers

This section describes the registers and programmable parameters of the EBU. All these registers can be read in User Mode, but can only be written in Supervisor Mode.

All registers are reset by the application reset.

SDRAM is not supported and therefore the registers **SDRMCON**, **SDRMOD**, **SDRMREF** and **SDRSTAT** are not useful.

Table 532 Register Address Space - EBU

Module	Base Address	End Address	Note
(EBU)	82000000_H	$87FFFFFF_H$	Access to External Memory via cached address range
	$A2000000_H$	$A7FFFFFF_H$	Access to external memory via non-cached address range
EBU	$F8400000_H$	$F840FFFF_H$	sri slave interface

Table 533 Register Overview - EBU (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	EBU Clock Control Register	0000000 H	U,SV,32, 64	SV,P,E,32,6 4	Application Reset	60
MODCON	EBU Configuration Register	0000004 H	U,SV,32, 64	SV,P,32,64	See page 62	62
MODID	EBU Module Identification Register	0000008 H	U,SV,32, 64	SV,P,32,64	Application Reset	64
USERCON	EBU Test/Control Configuration Register	000000C H	U,SV,32, 64	SV,P,32,64	Application Reset	84
EXTBOOT	EBU External Boot Configuration Register	0000010 H	U,SV,32	SV,P,32	Application Reset	64
ADDRSELx	EBU Address Select Register x	0000018 H+x*4	U,SV,32, 64	SV,P,32,64	See page 66	66
BUSRCONx	EBU Bus Configuration Register	0000028 $H+x*10_H$	U,SV,32, 64	SV,P,32,64	Application Reset	68
BUSRAPx	EBU Bus Read Access Parameter Register	000002C $H+x*10_H$	U,SV,32, 64	SV,P,32,64	Application Reset	73
BUSWCONx	EBU Bus Write Configuration Register	0000030 $H+x*10_H$	U,SV,32, 64	SV,P,32,64	Application Reset	71
BUSWAPx	EBU Bus Write Access Parameter Register	0000034 $H+x*10_H$	U,SV,32, 64	SV,P,32,64	Application Reset	75
SDRMCON	EBU SDRAM Control Register	0000068 H	U,SV,32, 64	SV,P,32,64	Application Reset	77
SDRMOD	EBU SDRAM Mode Register	000006C H	U,SV,32, 64	SV,P,32,64	Application Reset	79
SDRMREF	EBU SDRAM Refresh Control Register	0000070 H	U,SV,32, 64	SV,P,32,64	Application Reset	81

Table 533 Register Overview - EBU (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
SDRSTAT	EBU SDRAM Status Register	0000074 H	U,SV,32, 64	SV,P,32,64	Application Reset	83
ACCEN0	EBU Access Enable Register 0	00000B0 H	U,SV,32, 64	SV,SE,32,64	Application Reset	85
ACCEN1	EBU Access Enable Register 1	00000B4 H	U,SV,32, 64	SV,SE,32,64	Application Reset	85

Access Restrictions

Note: The EBU registers are accessible only through word or double-word accesses. Half-word and byte accesses on EBU registers will generate a bus error. Accesses to a 64bit word only partially populated with accessible registers will cause a bus error. Writes to unused address space, or writes to the endinit protected CLC register will also cause an error .

45.4.1 Clock Control Register, CLC

EBU Clock Control Register

CLC

EBU Clock Control Register **(0000000_H)** **Application Reset Value: $0055\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES								EBUDIVACK		DIV2ACK	SYNCACK	EBUDIV		DIV2	SYNC
r								r		r	r	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES							EPE	RES						DISS	DISR
r							rw	r						r	rw

Field	Bits	Type	Description
DISR	0	rw	**EBU Disable Request Bit** This bit is used for enable/disable control of the EBU. 0_B EBU disable is not requested 1_B EBU disable is requested
DISS	1	r	**EBU Disable Status Bit** DISS is always read as 0, as accessing the EBU will automatically enable it. 0_B EBU is enabled (default after reset) 1_B EBU is disabled
RES	7:2, 15:9, 31:24	r	**Reserved** Read as 0; should be written with 0.
EPE	8	rw	**Endinit Protection Enable** 0_B Disable Endinit protection of the CLC register (default after reset). 1_B Enable Endinit protection of the CLC register.
SYNC	16	rw	**EBU Clocking Mode** 0_B request EBU to run asynchronously to processor clock and use separate clock source 1_B request EBU to run synchronously to processor (default after reset)
DIV2	17	rw	**DIV2 Clocking Mode** 0_B standard clocking mode. clock input selected by SYNC bitfield (default after reset). 1_B request EBU to run off processor clock divided by 2.
EBUDIV	19:18	rw	**EBU Clock Divide Ratio** 00_B request EBU to run off input clock 01_B request EBU to run off input clock divided by 2 (default after reset) 10_B request EBU to run off input clock divided by 3 11_B request EBU to run off input clock divided by 4

Field	Bits	Type	Description
SYNCACK	20	r	**EBU Clocking Mode Status** 0_B the EBU is asynchronous to the SRI bus clock and is using a separate clock source 1_B EBU is synchronous to the SRI bus clock (default after reset)
DIV2ACK	21	r	**DIV2 Clocking Mode Status** 0_B EBU is using standard clocking mode. clock input selected by SYNC bitfield (default after reset). 1_B EBU is running off processor clock divided by 2.
EBUDIVACK	23:22	r	**EBU Clock Divide Ratio Status** 00_B EBU is running off input clock (default after reset) 01_B EBU is running off input clock divided by 2 10_B EBU is running off input clock divided by 3 11_B EBU is running off input clock divided by 4

45.4.2 Configuration Register, MODCON

EBU Configuration Register

MODCON
EBU Configuration Register **(0000004_H)** **Reset Value: Table 534**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALE	BUSSTATE	RES			OCDS_SUSP_DIS	FAST_SRI	FIFO_BYPASS	LOCKTIMEOUT							
rw	rh	r			rw	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMEOUTC								ARBMODE		ARBSYNC	EXTLOCK	CLK_COMB	SDTRI	LCKABRT	STS
rw								rw		rw	rw	rw	rw	rwh	r

Field	Bits	Type	Description
STS	0	r	**Memory Status Bit** Software access to the $\overline{WAIT}$ input pin to the EBU.
LCKABRT	1	rwh	**Lock Abort** This is a status bit which is set when a chip select lock has been cancelled by a program fetch or data read. The flag is cleared by writing 1_B to the field. See **Section 45.3.6.1** 0_B Lock function is operating normally 1_B Lock function has been aborted by processor instruction read from locked device
SDTRI	2	rw	**SDRAM Tristate** 0_B SDRAM control signals are driven by the EBU when the EBU does not own the external bus. SDRAM cannot be shared. 1_B SDRAM control signals are tri-stated by the EBU when the EBU does not own the external bus. The SDRAM can be shared. *Note: The signals affected by this setting are CKE, SDCLKO, CAS and RAS*
CLK_COMB	3	rw	**Combined External Bus Clock** Burst flash protocol and SDRAM protocol devices use the same external clock when set. 0_B The SDRAM and synchronous state machines use clocks on separate pins 1_B Both state machines use the clock on BFCLKO when accessing external memories
EXTLOCK	4	rw	**External Bus Lock Control** Allows the external bus arbitration to be locked with the EBU owning the bus. 0_B External bus is not locked after the EBU gains ownership 1_B External bus is locked after the EBU gains ownership

Field	Bits	Type	Description
ARBSYNC	5	rw	**Arbitration Signal Synchronization Control** Assumed status of the arbitration control signals, $\overline{BREQ}$ and $\overline{HLDA}$ 0_B Arbitration inputs are synchronous 1_B Arbitration inputs are asynchronous, use two resynchronisation flip-flops.
ARBMODE	7:6	rw	**Arbitration Mode Selection** Operating mode of the external bus arbitration 00_B No Bus arbitration mode selected 01_B Arbiter Mode arbitration mode selected 10_B Participant arbitration mode selected 11_B Sole Master arbitration mode selected
TIMEOUTC	15:8	rw	**Bus Time-out Control** This bit field determines the number of inactive cycles after the EBU gains ownership of the external bus before an arbitration time-out occurs and re-arbitration can occur. 00_H Time-out is disabled. 01_H Time-out is generated after 1 x 8 clock cycles. FF_H Time-out is generated after 255 x 8 clock cycles.
LOCKTIMEOUT	23:16	rw	**Lock Timeout Counter Preload** Value to be preloaded into the timeout counter for the arbitration lock. *Note: (see **Section 45.3.6.1**)*
FIFO_BYPASS	24	rw	**Reserved** Must be set to 0_B for correct operation of the SDRAM
FAST_SRI	25	rw	**Clock Cycles used for SRI Address Decode** 0_B Use two clock cycles for address decode (default after reset) 1_B Use one clock cycle for address decode. See **"Address Comparison" on Page 19** for a detailed description.
OCDS_SUSP_DIS	26	rw	**OCDS SUSPEND Disable** The EBU external arbitration will normally be disabled with the EBU owning the external bus while OCDS is active. Setting this bit will allow external bus arbitration to continue operation. See **Section 45.3.8.7**
RES	29:27	r	**Reserved** write with 0_H
BUSSTATE	30	rh	**External Bus State** Status bit controlled by the arbitration logic (See **"External Bus Arbitration" on Page 6**) reflecting the ownership of the external memory bus. 0_B The EBU does not own the external bus and attached memories cannot be accessed. 1_B The EBU owns the external bus.
ALE	31	rw	**ALE Mode** Switch the $\overline{ADV}$ output to be an active high ALE signal instead of active low $\overline{ADV}$. 0_B Output is $\overline{ADV}$ 1_B Output is ALE

Table 534 Reset Values of MODCON

Reset Type	Reset Value	Note
Application Reset	$0000\ 0020_H$	Internal Boot Mode
Application Reset	$0000\ 00E0_H$	External Boot Mode

EBU Module Identification Register

MODID

EBU Module Identification Register (0000008_H) Application Reset Value: $0014\ C00B_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID_VALUE															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID_VALUE															
r															

Field	Bits	Type	Description
ID_VALUE	31:0	r	**Module Identification Value**

45.4.3 External Boot Configuration Control Register, EXTBOOT

EBU External Boot Configuration Register

EXTBOOT

EBU External Boot Configuration Register (0000010_H) Application Reset Value: $0000\ 0001_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EBUCFG	RES														
w	r														

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES														CFGERR	CFGEND
r														rh	rh

Field	Bits	Type	Description
CFGEND	0	rh	**Configuration End** 0_B Configuration fetch is running 1_B Configuration fetch has completed or not started

Field	Bits	Type	Description
CFGERR	1	rh	**Configuration Fetch Error** 0_B No error 1_B The configuration fetch has returned a word with all bits set. This indicates an unprogrammed or missing flash
RES	30:2	r	**Reserved** Read as 0; must be written with 0.
EBUCFG	31	w	**Configuration Word Fetch** Write 1_B to trigger automatically set the EBU to sole master arbitration mode and fetch a configuration word from external memory. Always reads 0_B

45.4.4 Address Select Register, ADDRSELx (x=0-2)

EBU Address Select Register x

ADDRSELx (x=0-2)
EBU Address Select Register x **(0000018_H+x*4)** **Reset Value: Table 535**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASE															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE				ALTSEG				MASK				GLOBALCS	WPROT	ALTENAB	REGENAB
rw				rw				rw				rw	rw	rw	rw

Field	Bits	Type	Description
REGENAB	0	rw	**Memory Region Enable** 0_B Memory region is disabled (default after reset[1)]). 1_B Memory region is enabled.
ALTENAB	1	rw	**Alternate Segment Comparison Enable** 0_B ALTSEG is never compared to SRI address (default after reset). 1_B ALTSEG is always compared to SRI address.
WPROT	2	rw	**Memory Region Write Protect** 0_B Region is enabled for write accesses 1_B Region is write protected.
GLOBALCS	3	rw	**Combined Chip Select Control** Controls whether the $\overline{\text{CSCOMB}}$ output should be asserted for valid accesses to this region. 0_B $\overline{\text{CSCOMB}}$ not asserted 1_B $\overline{\text{CSCOMB}}$ asserted
MASK	7:4	rw	**Memory Region Address Mask** Specifies the number of right-most bits in the base address starting at bit 26, which should be included in the address comparison. Bits [31:27] will always be part of the comparison.
ALTSEG	11:8	rw	**Memory Region Alternate Segment** Alternate segment to be compared to SRI address bit [31:28].
BASE	31:12	rw	**Memory Region Base Address** Base address to be compared to SRI address in conjunction with the mask control.

1) except for ADDRSEL0, REGENAB in register ADDRSEL0 is 1_B after reset.

Table 535 Reset Values of ADDRSELx (x=0-2)

Reset Type	Reset Value	Note
Application Reset	$A000\ 0001_H$	Value for ADDRSEL0
Application Reset	$0000\ 0000_H$	Value for other ADDRSEL registers

45.4.5 Bus Read Configuration Register, BUSRCONx (x=0-2)

EBU Bus Configuration Register

*Note: When in external boot mode (see **Page 14**) the reset value of BUSCON0 is overwritten automatically (subsequent to the release of reset) as a result of the external Boot Configuration Value fetch.*

BUSRCONx (x=0-2)
EBU Bus Configuration Register **(0000028_H+x*10_H)** **Application Reset Value: $00D3\ 0040_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AGEN				LCKABRT	AAP	WAIT		PORTW		BCGEN		WAITINV	DBA	EBSE	ECSE
rw				rwh	rw	rw		rw		rw		rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES1								NAA	BFCMSEL	FDBKEN	RESERVED	FBBMSEL	FETBLEN		
r								rw	rw	rw	r	rw	rw		

Field	Bits	Type	Description
FETBLEN	2:0	rw	**Burst Length for Synchronous Burst** Defines maximum number of burst data cycles which are executed by Memory Controller during a burst access to a Synchronous Burst device. 000_B 1 data access (default after reset). 001_B 2 data accesses. 010_B 4 data accesses. 011_B 8 data accesses. 100_B reserved. … 111_B reserved.
FBBMSEL	3	rw	**Synchronous burst buffer mode select** 0_B Burst buffer length defined by value in fetblen (default after reset). 1_B Continuous mode. All data required for transaction is transferred in a single burst.
RESERVED	4	r	**Reserved** 0_B Reserved bit always reads ‘0’.
FDBKEN	5	rw	**Burst FLASH Clock Feedback Enable** 0_B BFCLK feedback not used. 1_B Incoming data and control signals (from the Burst FLASH device) are re-synchronised to the BFCLKI input.
BFCMSEL	6	rw	**Burst Flash Clock Mode Select** 0_B Burst Flash Clock runs continously with values selected by this register 1_B Burst Flash Clock is disabled between accesses

Field	Bits	Type	Description
NAA	7	rw	**Enable flash non-array access workaround** set to logic one to enable workaround when region is accessed with address bit 28 set. See **"Flash Non-Array Access Support" on Page 52**
RES1	15:8	r	**Reserved** 00_H Reserved Value
ECSE	16	rw	**Early Chip Select for Synchronous Burst** 0_B CS is delayed. 1_B CS is not delayed. *Note: (see **Control of ADV & Other Signal Delays During Asynchronous Accesses** and **Control of ADV & Control Signal Delays During Synchronous Accesses**)*
EBSE	17	rw	**Early Burst Signal Enable for Synchronous Burst** 0_B ADV is delayed. 1_B ADV is not delayed. *Note: (see **Control of ADV & Other Signal Delays During Asynchronous Accesses** and **Control of ADV & Control Signal Delays During Synchronous Accesses**)*
DBA	18	rw	**Disable Burst Address Wrapping** 0_B Memory Controller automatically re-aligns any non-aligned synchronous burst access so that data can be fetched from the device in a single burst transaction. 1_B Memory Controller always starts any burst access to a synchronous burst device at the address specified by the SRI request. Any required address wrapping must be automatically provided by the Burst FLASH device. *Note: Care must be taken with the use of this feature. The Burst capable device must be programmed to wrap at the appropriate address boundary prior to selection of this mode. **"Critical Word First Read Accesses" on Page 49***
WAITINV	19	rw	**Reversed polarity at WAIT** 0_B **OFF**, input at WAIT pin is active low (default after reset). 1_B **Polarity_reversed**, input at WAIT pin is active high.
BCGEN	21:20	rw	**Byte Control Signal Control** This bit field selects the timing mode of the byte control signals. 00_B Byte control signals follow chip select timing. 01_B Byte control signals follow control signal timing ($\overline{RD}$, RD/$\overline{WR}$) (default after reset). 10_B Byte control signals follow write enable signal timing (RD/$\overline{WR}$ only). 11_B SDRAM access type. Signals used for DQM.
PORTW	23:22	rw	**Device Addressing Mode** See **Table 513** and **Table 515**

Field	Bits	Type	Description
WAIT	25:24	rw	**External Wait Control** Function of the WAIT input. This is specific to the device type (i.e. the agen field). For Asynchronous Devices, see **"External Extension of the Command Phase by WAIT" on Page 34** For Synchronous Burst Devices, see **"External Cycle Control via the WAIT Input" on Page 51**
AAP	26	rw	**Asynchronous Address phase:** Enables an access mode for synchronous memories where the clock is not started until after the address hold phase. 0_B Clock is enabled at beginning of access. 1_B Clock is enabled at after address phase.
LCKABRT	27	rwh	**Lock Abort** This is a status bit which is set when the chip select lock has been cancelled by a program fetch or data read. The flag is cleared by writing 1_B to the field. See **Section 45.3.6.1** 0_B Lock function is operating normally 1_B Lock function has been aborted by processor instruction read from locked device
AGEN	31:28	rw	**Device Type for Region** See **Section 45.3.11.1Programmable Device Types**

45.4.6 Bus Write Configuration Register, BUSWCONx (x=0-2)

EBU Bus Write Configuration Register

*Note: When in external boot mode (see **Page 14**) the reset value of BUSCON0 is overwritten automatically (subsequent to the release of reset) as a result of the external Boot Configuration Value fetch.*

BUSWCONx (x=0-2)
EBU Bus Write Configuration Register (0000030_H+x*10_H) Application Reset Value: 00D3 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AGEN				LOCKCS	AAP	WAIT		PORTW		BCGEN		WAITINV	RES1	EBSE	ECSE
rw				rw	rw	rw		r		rw		rw	r	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								NAA	RES			FBBMSEL	FETBLEN		
r								r	r			rw	rw		

Field	Bits	Type	Description
FETBLEN	2:0	rw	**Burst Length for Synchronous Burst** Defines maximum number of burst data cycles which are executed by Memory Controller during a burst access to a Synchronous Burst device. 000_B 1 data access (default after reset). 001_B 2 data accesses. 010_B 4 data accesses. 011_B 8 data accesses. 100_B reserved. … 111_B reserved.
FBBMSEL	3	rw	**Synchronous burst buffer mode select** 0_B Burst buffer length defined by value in FETBLEN (default after reset). 1_B Continuous mode. All data required for transaction transferred in single burst
RES	6:4	r	**Reserved, always reads 0**
NAA	7	r	**Enable flash non-array access workaround** set to logic one to enable workaround when region is accessed with address bit 28 set. See **"Flash Non-Array Access Support" on Page 52**. Mirror of equivalent field in BUSRCON register.
RES0	15:8	r	**Reserved** 00_H Reserved Value

Field	Bits	Type	Description
ECSE	16	rw	**Early Chip Select for Synchronous Burst** 0_B CS is delayed. 1_B CS is not delayed. *Note: (see **Control of ADV & Other Signal Delays During Asynchronous Accesses** and **Control of ADV & Control Signal Delays During Synchronous Accesses**)*
EBSE	17	rw	**Early Burst Signal Enable for Synchronous Burst** 0_B ADV is delayed. 1_B ADV is not delayed. *Note: (see **Control of ADV & Other Signal Delays During Asynchronous Accesses** and **Control of ADV & Control Signal Delays During Synchronous Accesses**)*
RES1	18	r	**Reserved, always reads 0**
WAITINV	19	rw	**Reversed polarity at WAIT** 0_B **OFF**, input at WAIT pin is active low (default after reset). 1_B **Polarity_reversed**, input at WAIT pin is active high.
BCGEN	21:20	rw	**Byte Control Signal Control** This bit field selects the timing mode of the byte control signals. 00_B Byte control signals follow chip select timing. 01_B Byte control signals follow control signal timing ($\overline{RD}$, RD/$\overline{WR}$) (default after reset). 10_B Byte control signals follow write enable signal timing (RD/$\overline{WR}$ only). 11_B Reserved.
PORTW	23:22	r	**Device Addressing Mode** See **Table 513** and **Table 515**
WAIT	25:24	rw	**External Wait Control** Function of the WAIT input. This is specific to the device type (i.e. the agen field). For Asynchronous Devices, see **"External Extension of the Command Phase by WAIT" on Page 34** For Synchronous Burst Devices, see **"External Cycle Control via the WAIT Input" on Page 51**
AAP	26	rw	**Asynchronous Address phase:** Enables an access mode for synchronous memories where the clock is not started until after the address hold phase. 0_B Clock is enabled at beginning of access. 1_B Clock is enabled at after address phase.
LOCKCS	27	rw	**Lock Chip Select** Enable Chip Select for Automatic Locking in the event of a write access 0_B Chip Select cannot be locked (default after reset). 1_B Chip Select will be automatically locked when written to from the processor data port.
AGEN	31:28	rw	**Device Type for Region** See **Section 45.3.11.1Programmable Device Types**

45.4.7 Bus Read Access Parameter Register, BUSRAPx (x=0-2)

EBU Bus Read Access Parameter Register

*Note: When in external boot mode (see **Page 14**), the reset value of BUSAP0 is overwritten automatically (subsequent to the release of reset) as a result of the external Boot Configuration Value fetch.*

BUSRAPx (x=0-2)

EBU Bus Read Access Parameter Register ($000002C_H$+x*10_H) Application Reset Value: FFFF $FFFF_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRC				AHOLDC				CMDDELAY				EXTDATA		EXTCLOCK	
rw				rw				rw				rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAC				WAITRDC					RDRECOVC			RDDTACS			
rw				rw					rw			rw			

Field	Bits	Type	Description
RDDTACS	3:0	rw	**Recovery Cycles between Different Regions** This bit field determines the number of clock cycles of the Recovery Phase between consecutive accesses directed to different regions or different types of access. See **"Recovery Phase (RP)" on Page 27** 0_H No Recovery Phase clock cycles available. 1_H 1 clock cycle selected. E_H 14 clock cycles selected. F_H 15 clock cycles selected.
RDRECOVC	6:4	rw	**Recovery Cycles after Read Accesses** This bit field determines the basic number of clock cycles of the Recovery Phase at the end of read accesses. 000_B No Recovery Phase clock cycles available. 001_B 1 clock cycle selected. 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.
WAITRDC	11:7	rw	**Programmed Wait States for read accesses** Number of programmed wait states for read accesses. For synchronous accesses, this will always be adjusted so that the phase exits on a rising edge of the external clock. 00_H 1 wait state. 01_H 1 wait states. 02_H 2 wait state. $1E_H$ 30 wait states. $1F_H$ 31 wait states.

Field	Bits	Type	Description
DATAC	15:12	rw	**Data Hold Cycles for Read Accesses** This bit field determines the basic number of Data Hold phase clock cycles during read accesses. This is used for a limited number of read cycle types. See **"Data Hold Phase (DH)" on Page 26**. 0_H No Recovery Phase clock cycles available. 1_H 1 clock cycle selected. E_H 14 clock cycles selected. F_H 15 clock cycles selected.
EXTCLOCK	17:16	rw	**Frequency of external clock at pin BFCLKO or SDCLKO** See **"External Bus Clock Generation" on Page 6**
EXTDATA	19:18	rw	**Extended data** See **Burst Phase (BP)** 00_B external memory outputs data every BFCLK cycle 01_B external memory outputs data every two BFCLK cycles 10_B external memory outputs data every four BFCLK cycles 11_B external memory outputs data every eight BFCLK cycles
CMDDELAY	23:20	rw	**Command Delay Cycles** This bit field determines the basic number of Command Delay phase clock cycles. 0_H 0 clock cycle selected. 1_H 1 clock cycle selected. 6_H 6 clock cycles selected. 7_H 7 clock cycles selected.
AHOLDC	27:24	rw	**Address Hold Cycles** This bit field determines the number of clock cycles of the address hold phase.. 0_H 1 clock cycle selected 1_H 1 clock cycle selected E_H 14 clock cycles selected F_H 15 clock cycles selected
ADDRC	31:28	rw	**Address Cycles** This bit field determines the number of clock cycles of the address phase. 0_H 1 clock cycle selected 1_H 1 clock cycle selected 6_H 6 clock cycles selected 7_H 7 clock cycles selected

45.4.8 Bus Write Access Parameter Register, BUSWAPx (x=0-2)

EBU Bus Write Access Parameter Register

BUSWAPx (x=0-2)

EBU Bus Write Access Parameter Register (0000034_H+x*10_H) Application Reset Value: FFFF $FFFF_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRC				AHOLDC				CMDDELAY				EXTDATA		EXTCLOCK	
rw				rw				rw				rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAC				WAITWRC					WRRECOVC			WRDTACS			
rw				rw					rw			rw			

Field	Bits	Type	Description
WRDTACS	3:0	rw	**Recovery Cycles between Different Regions** This bit field determines the number of clock cycles of the Recovery Phase between consecutive accesses directed to different regions or different types of access. See **"Recovery Phase (RP)" on Page 27** 0_H No Recovery Phase clock cycles available. 1_H 1 clock cycle selected. E_H 14 clock cycles selected. F_H 15 clock cycles selected.
WRRECOVC	6:4	rw	**Recovery Cycles after Write Accesses** This bit field determines the basic number of clock cycles of the Recovery Phase at the end of write accesses. See **"Recovery Phase (RP)" on Page 27** 000_B No Recovery Phase clock cycles available. 001_B 1 clock cycle selected. 110_B 6 clock cycles selected. 111_B 7 clock cycles selected.
WAITWRC	11:7	rw	**Programmed Wait States for write accesses** Number of programmed wait states for write accesses. For synchronous accesses, this will always be adjusted so that the phase exits on a rising edge of the external clock. 00_H 1 wait state. 01_H 1 wait states. 02_H 2 wait state. $1E_H$ 30 wait states. $1F_H$ 31 wait states.

Field	Bits	Type	Description
DATAC	15:12	rw	**Data Hold Cycles for Write Accesses** This bit field determines the basic number of Data Hold phase clock cycles during write accesses. 0_H No Recovery Phase clock cycles available. 1_H 1 clock cycle selected. E_H 14 clock cycles selected. F_H 15 clock cycles selected.
EXTCLOCK	17:16	rw	**Frequency of external clock at pin BFCLKO or SDCLKO** See **"External Bus Clock Generation" on Page 6**.
EXTDATA	19:18	rw	**Extended data** See **Burst Phase (BP)** 00_B external memory outputs data every BFCLK cycle 01_B external memory outputs data every two BFCLK cycles 10_B external memory outputs data every four BFCLK cycles 11_B external memory outputs data every eight BFCLK cycles
CMDDELAY	23:20	rw	**Command Delay Cycles** This bit field determines the basic number of Command Delay phase clock cycles. 0_H 0 clock cycles selected. 1_H 1 clock cycles selected. 6_H 6 clock cycles selected. 7_H 7 clock cycles selected.
AHOLDC	27:24	rw	**Address Hold Cycles** This bit field determines the number of clock cycles of the address hold phase.. 0_H 1 clock cycle selected 1_H 1 clock cycle selected E_H 14 clock cycles selected F_H 15 clock cycles selected
ADDRC	31:28	rw	**Address Cycles** This bit field determines the number of clock cycles of the address phase. 0_H 1 clock cycle selected 1_H 1 clock cycle selected E_H 14 clock cycles selected F_H 15 clock cycles selected

45.4.9 SDRAM Control Register, SDRMCON

EBU SDRAM Control Register

SDRMCON
EBU SDRAM Control Register **(0000068$_H$)** **Application Reset Value: 1000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SDCMSEL	PWR_MODE		CLKDIS	RES			BANKM			CRC					
rw	rw		rw	r			rw			rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCD		AWIDTH		CRP		CRSC		CRFSH				CRAS			
rw		rw		rw		rw		rw				rw			

Field	Bits	Type	Description
CRAS	3:0	rw	**Row to precharge delay counter** Number of clock cycles between row activate command and a precharge command. 0-15$_H$: Minimum Cras + 1 clock cycles (default after reset Cras is 0)
CRFSH	7:4	rw	**Initialization refresh commands counter** Number of refresh commands issued during power-up initialization sequence. 0-15$_H$: Perform Crfsh + 1 refresh cycles (default after reset Crfsh is 0)
CRSC	9:8	rw	**Mode register set-up time** Number of NOP cycles after a mode register set command. 0-3$_H$: Insert Crsc + 1 NOP cycles (default after reset Crsc is 0)
CRP	11:10	rw	**Row precharge time counter** Number of NOP cycles inserted after a precharge command. The actual number performed can be greater due to CAS latency and burst length. 0-3$_H$: Insert Crsc + 1 NOP cycles (default after reset Crp is 0)
AWIDTH	13:12	rw	**Width of column address** Number of address bits from bit 0 to be used for column address. e.g. for 16 bit DRAMs 00_B **reserved**, do not use 01_B Address(8:0) 10_B Address(9:0) 11_B Address(10:0)
CRCD	15:14	rw	**Row to column delay counter** Number of NOP cycles between a row address and a column address. 0-3$_H$: Insert Crcd + 1 NOP cycles (default after reset Crcd is 0).
CRC	21:16	rw	**Refresh cycle time counter** Number of NOP cycles following a refresh command before another command (other than a NOP) can be issued to the SDRAM. Combined with the Crce bit as follows:- 0-3F$_H$: Insert Crc + 1 NOP cycles.

Field	Bits	Type	Description
BANKM	24:22	rw	**Mask for bank tag** SRI address bits to be used for determining bank number. 000_B Reserved; (default after reset) 001_B Address bit 21 to 20 010_B Address bit 22 to 21 011_B Address bit 23 to 22 100_B Address bit 24 to 23 101_B Address bit 25 to 24 110_B Address bit 26 to 25 111_B Address bit 26
RES	27:25	r	**Reserved**
CLKDIS	28	rw	**Disable SDRAM clock output** 0_B clock enabled 1_B clock disabled
PWR_MODE	30:29	rw	**Power Save Mode used for gated clock mode** 00_B precharge before clock stop (default after reset) 01_B auto-precharge before clock stop 10_B active power down (stop clock without precharge) 11_B clock stop power down
SDCMSEL	31	rw	**SDRAM clock mode select** 0_B clock continuously runs 1_B clock disabled between accesses

45.4.10 SDRAM Mode Register, SDRMOD

EBU SDRAM Mode Register

SDRMOD

EBU SDRAM Mode Register **(000006C$_H$)** **Application Reset Value: 0000 0020$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XBA		XOPM													
rw		rw													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COLDSTART	RES	OPMODE							CASLAT			BTYP	BURSTL		
w	r	rw							rw			rw	rw		

Field	Bits	Type	Description
BURSTL	2:0	rw	**Burst length** Number of locations can be accessed with a single command. 000_B 1 (default after reset) 001_B 2 010_B 4 011_B 8 100_B 16 *Note: Other values reserved*
BTYP	3	rw	**Burst type** Memory Controller only supports sequential burst. 0_B Only this value should be written (default after reset) 1_B Reserved
CASLAT	6:4	rw	**CAS latency** Number of clocks between a READ command and the availability of data. 010_B Two clocks (default after reset) 011_B Three clocks *Note: Other values reserved*
OPMODE	13:7	rw	**Operation Mode** Memory Controller only supports burst write standard operation. 00_H Only this value should be written (default after reset) *Note: Other values reserved*
RES	14	r	**Reserved** 0_B Fixed Value
COLDSTART	15	w	**SDRAM coldstart** This bit will always read 0. If a write to the **SDRMOD** register takes place with this bit set, the SDRAM device mode register will be updated to match the data written to the register.

Field	Bits	Type	Description
XOPM	29:16	rw	**Extended Operation Mode** Value to be written to the extended mode register of a “Mobile” SDRAM device. This value is issued to the SDRAM via it’s address inputs during an extended mode register write. This field is wider than current extended mode registers to allow support of future enhanced “Mobile” SDRAM devices. *Note: Consult the appropriate SDRAM documentation for the function of these bits.*
XBA	31:30	rw	**Extended Operation Bank Select** Value to be written to the bank select pins of a “Mobile” SDRAM device during an extended mode register write operation. Control of these bits is provided to allow support of future enhanced “Mobile” SDRAM devices. *Note: Care must be taken when programming these bits to ensure that a valid extended mode register access occurs (e.g. it is possible to generate an extra unwanted standard mode register write by incorrect programming of these bits).* *4. Consult the appropriate SDRAM documentation for the function of these bits.*

45.4.11 SDRAM Refresh Control Register, SDRMREF

EBU SDRAM Refresh Control Register

SDRMREF
EBU SDRAM Refresh Control Register (0000070$_H$) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES				RES_DLY			ARFSH	SELFREX_DLY							
r				rw			rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERFSHC		AUTO SELFR	SELFR EN	SELFR ENST	SELFR EX	SELFR EXST	REFRESHR			REFRESHC					
rw		rw	rw	rh	rw	rh	rw			rw					

Field	Bits	Type	Description
REFRESHC	5:0	rw	**Refresh counter period** Number of clock cycles between refresh operations. Uses ((erfshc x 64) +refreshc) as a single counter value where 0_D means that no refresh is needed an the refresh generator is disable. This is the default setting after reset. For all other vales, the refresh period is ((erfshc x 64) + refreshc) x 64 clock cycles
REFRESHR	8:6	rw	**Number of refresh commands** The number of additional refresh commands issued to SDRAM each time a refresh is due. 000_B 1 refresh command is issued (default after reset). 001_B 1 additional refresh command is issued. 110_B 6 additional refresh commands are issued. 111_B 7 additional refresh commands are issued.
SELFREXST	9	rh	**Self Refresh Exit Status.** If this bit is set to ’1’, it means the Self Refresh Exit command has been successfully issued. This bit is reset when bit selfren is set to ’1’ or a reset takes place.
SELFREX	10	rw	**Self Refresh Exit (Power Up).** When this bit is written with ’1’ the Self Refresh Exit command is issued to all SDRAM devices, regardless whether they are attached to type 0 or type 1. This is also used after power is applied to drive CKE high
SELFRENST	11	rh	**Self Refresh Entry Status.** If this bit is set to ’1’, it means the Self Refresh Entry command has been successfully issued. This bit is reset when bit selfrex is set to ’1’ or a reset takes place.
SELFREN	12	rw	**Self Refresh Entry** When this bit is written with ’1’ the Self Refresh Entry command is issued to all SDRAM devices, regardless whether they are attached to type 0 or type 1.

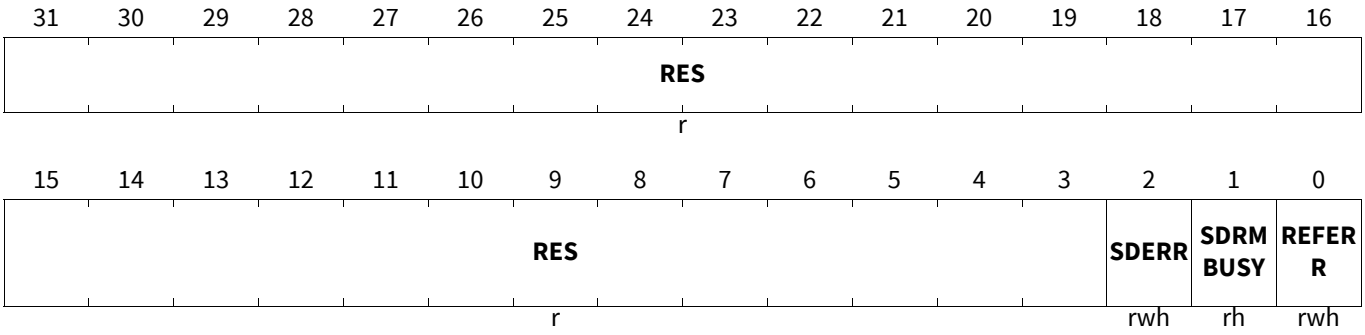
Field	Bits	Type	Description
AUTOSELFR	13	rw	**Automatic Self Refresh** When this bit is set to '1', Memory Controller will automatically issue the Self Refresh Entry command to all SDRAM devices when it gives up control of the external bus, and will automatically issue Self Refresh Exit when it regains control of the bus.
ERFSHC	15:14	rw	**Extended Refresh Counter Period** This field is used to increase the range of the refreshc field from 6 bits to 8 bits with erfshc being used as bits 7 and 6 of the extended field and refreshc as bit 5 to 0.
SELFREX_DLY	23:16	rw	**Self Refresh Exit Delay** Number of NOP cycles inserted after a self refresh exit before a command is permitted to the SDRAM/DDRAM.
ARFSH	24	rw	**Auto Refresh on Self refresh Exit** If set to one, an auto refresh cycle will be performed on exiting self refresh before the self refresh exit delay. If set to zero, no refresh will be performed.
RES_DLY	27:25	rw	**Delay on Power Down Exit** Number of NOPs after the SDRAM controller exits power down before an active command is permitted.
RES	31:28	r	**Reserved** 0_H Reserved Value

45.4.12 SDRAM Status Register, SDRSTAT

EBU SDRAM Status Register

SDRSTAT

EBU SDRAM Status Register (0000074$_H$) Application Reset Value: 0000 0000$_H$



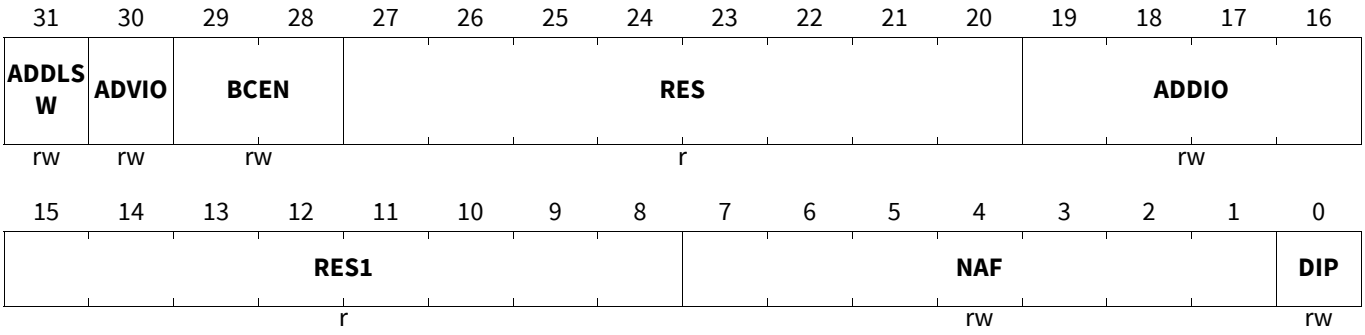
Field	Bits	Type	Description
REFERR	0	rwh	**SDRAM Refresh Error** Unsuccessful previous refresh request collides with a new request. This bit latches at 1_B and is reset by writing 0_B. 0_B No refresh error. 1_B Refresh error occurred.
SDRMBUSY	1	rh	**SDRAM Busy** The status of power-up initialization sequence. 0_B Power-up initialization sequence is not running 1_B Power-up initialization sequence is running
SDERR	2	rwh	**SDRAM read error** SDRAM controller has detected an error when returning read data. 0_B Reads running successfully 1_B Read error condition has been detected *Note: This bit latches at 1_B and is reset by writing 0_B.*
RES	31:3	r	**Reserved** 00000000_HReserved Value

45.4.13 Test/Control Configuration Register, USERCON

EBU Test/Control Configuration Register

USERCON

EBU Test/Control Configuration Register (000000C$_H$) Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
DIP	0	rw	**Disable Internal Pipelining**
NAF	7:1	rw	**No Assigned Function** Reserved for future requirements
RES1	15:8	r	**Reserved** Read as 0; should be written with 0.
ADDIO	19:16	rw	**Address Pins to GPIO Mode** Individual Control Bits for Address Bus Bits 19 downto 16 respectively. 0_H Address Bit is required for addressing memory 1_H Address Bit switched to GPIO function
RES	27:20	r	**Reserved**
BCEN	29:28	rw	**Byte Control Enable** 00_B Byte control off. Pins available for address or GPIO as set by Ports logic 01_B 8 bit byte control. Byte control 0 only. A(22:20) available for address or GPIO as set by Ports logic 10_B 16 bit byte control. $\overline{BC}$(1:0) available. A(21:20) available for address or GPIO as set by Ports logic 11_B 32 bit byte control. $\overline{BC}$(3:0) available.
ADVIO	30	rw	**ADV Pin to GPIO Mode** Control Bit for the $\overline{ADV}$/ALE output 0_B $\overline{ADV}$ pin is required for controlling memory 1_B $\overline{ADV}$ pin is switched to GPIO function
ADDLSW	31	rw	**Enable Address LSW, A(15:0) for GPIO** Switch the least significant 16 bits of the address bus to GPIO mode. 0_B A(15:0) in use by EBU 1_B A(15:0) used as GPIO.

45.4.14 Access Enable Registers, ACCEN0 and ACCEN1

EBU Access Enable Register 0

The Access Enable Register 0 controls write access for transactions to registers with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The LMU is prepared for an 6 bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6 bit TAG ID encoding. Write accesses not permitted by the setting of the register will be errored.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, … ,EN31 -> TAG ID 011111B.

ACCEN0

EBU Access Enable Register 0 **($00000B0_H$)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENx (x=0-31)	x	rw	**Access Enable for Master TAG ID x - ENn** This bit enables write access to the LMU register addresses for transactions with the Master TAG ID n 0_B Write access will not be executed. Read accesses will be executed. 1_B Write and read accesses will be executed

EBU Access Enable Register 1

The Access Enable Register 1 controls write access for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). These tags are not used in this system and so no programmable bits are provided.

ACCEN1

EBU Access Enable Register 1 **($00000B4_H$)** **Application Reset Value: FFFF FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RES															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES															
r															

Field	Bits	Type	Description
RES	31:0	r	**Reserved** Read as 1; should not be written.

45.5 IO Interfaces

This section provides information on the module connections.

Table 536 List of EBU0 Interface Signals

Interface Signals	I/O	Description
sx_sri		**sri slave interface**
		Access to external memory via non-cached address range
		Access to External Memory via cached address range
ADV_FDBK	in	**ADV Control Signal Feedback**
ADV	out	**Address Valid Control Signal**
HOLD	in	**EBU Arbitration**
SDCLKI	in	**SDRAM Clock Feedback**
BFCLKI	in	**Burst Flash Clock Feedback**
CKE	out	**SDRAM Clock Enable**
AD_IN(31:0)	in	**Data Bus Input**
AD(31:0)	out	**Data Bus Output**
A_IN(15:0)	in	**Address Input**
SDRAMA(13:0)	out	**SDRAM Address Bus**
RD	out	**Read Control**
RD_FDBK	in	**Read Feedback**
WR	out	**Write Control**
WR_FDBK	in	**Write Feedback**
BC(3:0)	out	**Byte Control**
CS_FDBK(2:0)	in	**Chip Select Feedback**
WAIT	in	**Wait Input**
BAA	out	**Burst Address Advance**
BA(1:0)	out	**Bank Address**
BFCLKO	out	**Burst Flash Clock Output**
SDCLKO	out	**SDRAM Clock Output**
CS(2:0)	out	**Chip Select**
HOLDA	out	**Hold Acknowledge**
DQM(1:0)	out	**SDRAM Write Data Valid**
BREQ	out	**Bus Request**
DQ(15:0)	out	**Data Line**
A(23:0)	out	**Address Output**

45.5.1 Bus State During Reset

The state of the various bus signals is controlled by Memory Controller during reset as follows:-

Table 537 Memory Controller External Bus pin states during reset

Pin Name	State during Reset[1)]	State during Idle[2)]	State during "no bus" mode
	GPIO	High Impedance - pull ups enabled to pull to '1'.	GPIO
	High impedance pull to '1' (high).	High Impedance - pull ups enabled to pull to '1'.	High Impedance - pull ups enabled to pull to '1' (High).
A(:19)	GPIO	Driven to '0' after reset, otherwise last used address	GPIO
A(:0)	High impedance pull to '1' (high).	Driven to '1' (High) at reset, last used address when idle	High impedance pull to '1' (high).
$\overline{CS}$	High impedance pull to '1' (high).	Driven to '1' (High).	High impedance pull to '1' (high).
$\overline{CS}$	GPIO	Driven to '1' (High).	GPIO
$\overline{RD}$	High impedance pull to '1' (high).	Driven to '1' (High).	High impedance pull to '1' (high).
$\overline{WR}$	High impedance pull to '1' (high).	Driven to '1' (High).	High impedance pull to '1' (high).
$\overline{CAS}$	GPIO	Driven to '1' (High).	GPIO
$\overline{RAS}$	GPIO	Driven to '1' (High).	GPIO
CKE	GPIO	Dependant on SDRAM clocking/power save mode	GPIO
$\overline{ADV}$	GPIO	Driven to '1' (High).	GPIO
SDCLKO	GPIO	Driven to '0' (Low).	GPIO
SDCLKI	GPIO	High Impedance	GPIO
BFCLKI	GPIO	High Impedance	GPIO
BFCLKO	GPIO	Driven to '0' (Low).	GPIO
$\overline{BC}$	GPIO	Driven to '1' (High).	GPIO
$\overline{BC}$	High impedance pull to '1' (high).	Driven to '1' (High).	High impedance pull to '1' (high).
$\overline{WAIT}$	GPIO	Always an input (must have a pull-resistor to inactive state).	GPIO
$\overline{HOLD}$	GPIO	depends on state of bus arbitration logic	GPIO
$\overline{HLDA}$	GPIO		GPIO
$\overline{BREQ}$	GPIO		GPIO
$\overline{BAA}$	GPIO	Driven to '1' (High).	GPIO

1) GPIO controlled pins should be high impedance with pull up during reset.

2) Assuming that the pins have not been made available as GPIO.

Applicable reset is cgu_con_clk_rst_n_i.

45.6 Revision History

Table 538 Revision History

Reference	Change to Previous Version	Comment
V4.0.10		
Page 58	Non-cached/cached exchanged in register address space table.	–
Page 68	List of reserved bit-fields changed in Bus Read Configuration Register.	–
Page 71	List of reserved bit-fields changed in Bus Write Configuration Register.	–
Page 58	Wrong write access mode “BE” for register CLC removed in register overview table.	
V4.0.11		
Page 86	Included table changed to fix formatting issues.	
V4.0.12		
Page 1	**Section 45**,“**SRI External Bus Unit (EBU)**“ updated figure “EBU interface Diagram“.	
Page 1	**Section 45.1**,“**Feature List**“ removed “SDRAM support “ and “External bus frequency for SDRAM”.	
Page 2	**Section 45.2**,“**Overview**“ updated figure “Typical External Memory System“.	
Page 2	**Section 45.3.1**, “**References**“ updated text.	
-	Section “SDRAM Interface“ is changed to “Internal“.	
Page 58	**Section 45.4**, “**Registers**“ updated to remove SDRAM support.	

46 SD- and eMMC Interface (SDMMC)

The purpose of the SDMMC module is to enable communication to external managed NAND Flashes using the SD- or eMMC interface. The focus of the module is set to the communication with a single embedded (eMMC) memory device or a single SD-card.

46.1 Feature List

This section describes the features of the SDMMC module.

- Communication to eMMC memories
  - Communication using 1-, 4- or 8-data lines
  - Legacy MMC card- and High-speed SDR mode supported
- Communication to SD-cards
  - Communication using 1- or 4-data lines
  - Default- and High Speed Mode supported
  - Communication to SDIO cards
- Communication support using different levels of DMA support
  - Programmed Input Output (PIO) - Operation without DMA
  - Single Operation DMA (SDMA)
  - Advanced DMA-2 (ADMA2)
  - Advanced DMA-3 (ADMA3)
- Block buffer of 2x512 Bytes
- Compliant to “Embedded Multi-Media Card (eMMC) Standard (5.1)”with the following exceptions:
  - no dual voltage eMMC support, only high voltage (3.3V)
  - no high clock frequencies for HS200/HS400, only 0-50MHz
  - no large 4kB sector device support, only 512B sectors
  - no DDR signaling only SDR signaling
  - no data strobe support
  - no high speed DDR, HS200 or HS400 speed mode support, only Legacy MMC card and High Speed SDR
  - no “Interrupt” mode, only “device identification”, “data transfer” and “inactive” mode
  - no clock tuning support
  - no multi device support
- Compliant to “SD Specifications Part A2 SD Host Controller Standard Specification v4.10” with the following exceptions:
  - no Multi Slot Support , only single cards
  - no suspend and resume functionality
  - no Sampling Clock Tuning supported, only Default Speed Mode and High Speed Mode supported
  - no Suspend/Resume operation in an SDIO card
- Compliant to “SD Specifications Part 1 Physical Layer Specification v4.2” with the following exceptions
  - no UHS-II voltages, only high voltage SD Memory Card
  - no UHS-I and UHS-II speed modes, only Default Speed Mode and High Speed Mode
  - no physical write protection switch support
  - no card insertion detection, only sensing of pull-up on DAT line

– no SPI card support

46.2 Functional Description

46.2.1 Architecture

46.2.1.1 System Description

The DWC_mshc controller supports master bus interface and the slave bus interface. It also contains a DMA for data transfer, a registers unit and a FIFO controller.

Figure 747 describes the main functional building blocks of the DWC_mshc core.

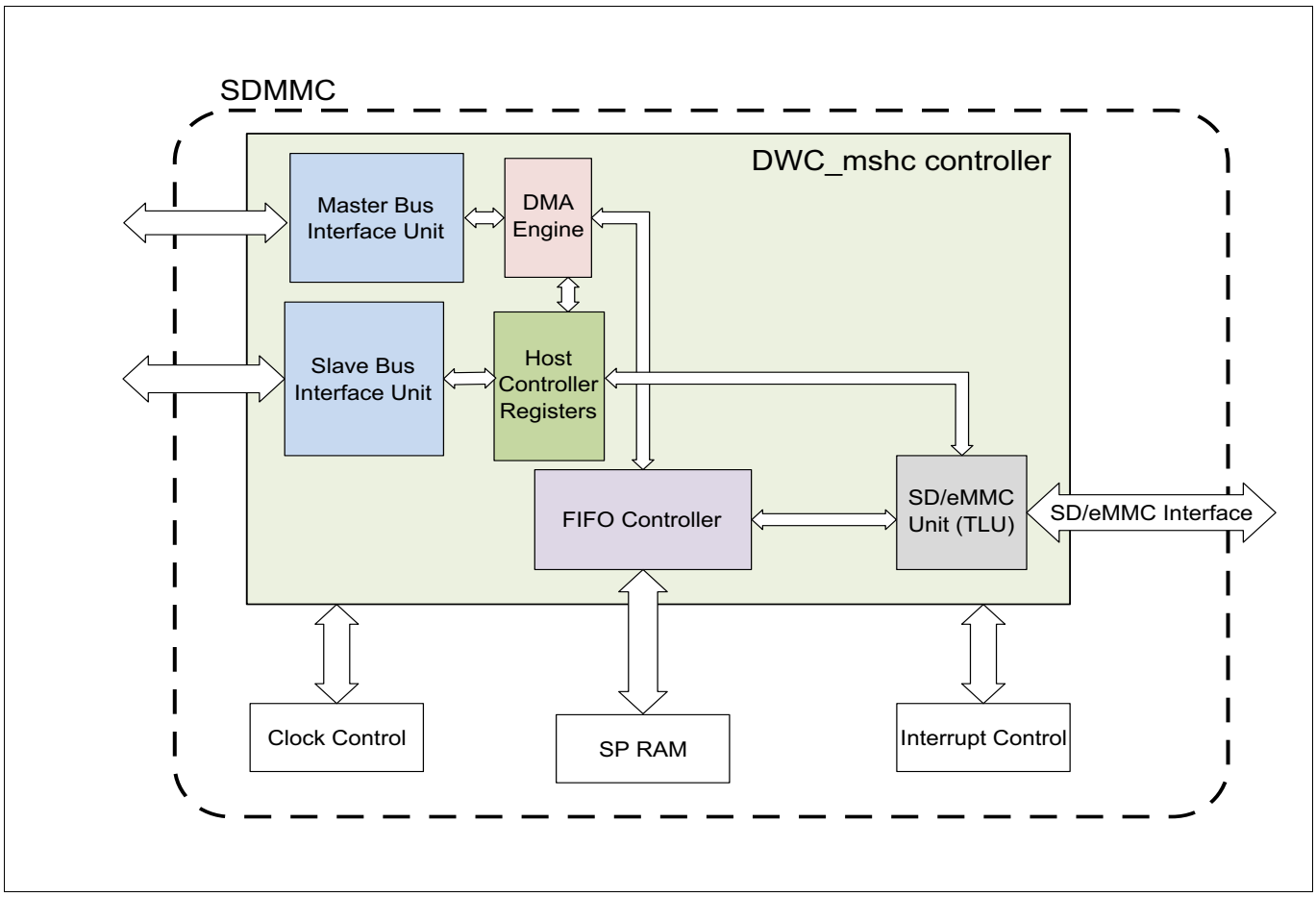


Figure 747 System-Level Block Diagram

Following are various modules in the Mobile Storage Host Controller:

- **The Master Bus Interface Unit** on **Page 4**
- **The Slave Bus Interface (SBI) Module** on **Page 4**
- **DMA Engine** on **Page 4**
- **DWC_mshc Registers** on **Page 4**
- **FIFO Controller** on **Page 4**
- **SD/eMMC Unit** on **Page 4**
- **SPRAM (single port RAM)** on **Page 5**
- **Interrupts** on **Page 5**

The Master Bus Interface Unit

This Master Bus Interface Unit (MBIU) implements the logic to transfer data on the AMBA High-Performance bus. The AHB interface transfers data to and from the system memory through the AHB master bus interface.

The Slave Bus Interface (SBI) Module

The AHB slave bus interface (SBI) module implements the logic to primarily access the DWC_mshc registers by using an external AMBA high-performance (AHB) bus. This module supports only the little endian scheme for register accesses.

DMA Engine

The DMA Engine unit handles data transfer between DWC_mshc and system memory. The key features of this unit are as follows.

- Support SDMA/ADMA2/ADMA3 modes
- Write back the received data packets to the system memory
- All bus accesses by the DMA engine are executed in user mode
- Support of SINGLE and burst (INCR4 and INCR8) transfers
  - In order of achieving optimal performance with burst transfers, data should be aligned to 256-bit for INCR8 and 128-bit for INCR4. E.g. in case of INCR8 and non 256-bit aligned data, the DMA engine will use SINGLE and INCR4 transfers until 256-bit alignment is achieved.
- 32-bit addressing support
- 32-bit data width

DWC_mshc Registers

The host controller register unit comprises of the standard SD host controller registers as specified in the SD Specifications Part A2 SD Host Controller Standard Specification Version 4.20a.

For the following registers the internal clock controlled by CLK_CTRL.INTERNAL_CLK_EN must be enabled before accessing them:

- BLOCKSIZE
- BLOCKCOUNT
- ARGUMENT
- XFER_MODE
- CMD

FIFO Controller

The data FIFO controller interfaces the internal FIFO of the host and the transaction controller unit.

SD/eMMC Unit

SD/eMMC unit is responsible for managing SD/eMMC interface protocols. The key features of this unit are to:

- Generate DS/HS command and data packets
- Generate CRC and check for command and data packets
- Handle packet timeouts
- Support 1-bit DAT, 4-bit DAT, and 8-bit DAT modes
- SD/eMMC interface signals
  - SDMMC_DAT0-7: data lines

- SDMMC_CMD: command line
- SDMMC_CLK: clock line

For the different data widths 1-, 4- and 8-wire communication different enable signals for the corresponding Data Pins are required to activate and deactivate these lines according to the output width selection.

SPRAM (single port RAM)

The SPRAM is used for the storage of data blocks. This RAM has twice the size of the SD/eMMC blocksize of 512bytes. This results in a 1kbyte buffer.

The single port RAM is not initialized after reset. Reading the RAM should only be done after the uninitialized values got overwritten (either data was received by the card, or the RAM was written by application software to prepare a write to the card). This is to avoid checksum errors (safety alerts) when the software accesses uninitialized RAM cells.

Interrupts

The DWC_mshc_controller supports two types of interrupts:

- Status Interrupts (corresponding to register NORMAL_INT_STAT_R) and
- Error Interrupts (corresponding to register ERROR_INT_STAT_R)

46.2.1.2 Error Handling

DWC_mshc is capable of detecting different types of error in SD and eMMC. Error is detected in either command or data portion of the transaction. When an error is detected, the Error interrupt in Normal interrupt status register (NORMAL_INT_STAT_R) is set.

DWC_mshc supports Error Interrupt Status register (ERROR_INT_STAT_R) to capture the interrupt status. When error occurs in the SD and eMMC mode, one of the bits is set in the Error Interrupt Status register. command is used to recover from error detected during data transfer. In addition to these two registers, there are two more error status registers, namely Auto CMD Error Status register and ADMA Error Status registers that are supported in both SD and eMMC modes.

Table 539 lists error types and categories into which they can be grouped for SD/eMMC mode.

Table 539 Error Types and Categories for SD/eMMC Mode

Error Type	Categories
Command Errors	• Command Timeout Error • Command CRC Error • Command End Bit Error • Command Index Error • Command Conflict Error • Response Error
Auto Command Errors	• Command not issued by Auto CMD12 Error • Auto Command Timeout Error • Auto Command CRC Error • Auto Command End Bit Error • Auto Command Index Error • Auto Command Conflict Error • Auto CMD response Error

Table 539 Error Types and Categories for SD/eMMC Mode (cont'd)

Error Type	Categories
Data Errors	• Data Timeout Error • Data CRC Error • Data End bit Error • ADMA Error
Command Errors	• Timeout for CMD_RES • TID Error • Framing Error • CRC Error • RES Packet Error • EBSY Error • Header Error
Data Errors	• Timeout for Deadlock • ADMA Error • Unrecoverable Error • TID Error • Framing Error • CRC Error • Retry Expired Error • Header Error • EBSY Error

46.2.1.3 OCDS Suspend

The SDMMC module has the following behaviour for OCDS hard and soft suspend on TC39x.

- During OCDS suspend the SDMMC registers are not readable and writable with the exception of the registers starting from offset 0x300.
- The DMA master interface is stopped in operation.
  - If the suspend is asserted during an ongoing DMA transfer, then this transfer is ed and an error interrupt is generated if this is interrupt is enabled.

On all devices other than TC39x the OCDS suspend signal has no effect.

46.2.2 DWC_mshc Programming Sequences

46.2.2.1 Pin enabling

When using the SDMMC module the direction registers of the corresponding ports need to be configured to input. Other pin configurations (such as pulls and driver strength) need to be selected according to the application needs.

46.2.2.2 Programming Overview

A certain programming sequence is required for DWC_mshc and the card subsystem before starting data transfer with a connected card. Initially, both the host controller and the card needs to be initialized.

Following is the programming sequence required before starting the data transfer with the card:

1. Start providing hclk to DWC_mshc and apply its reset hresetn. This clock is required for card detection logic and for accessing most of the registers in DWC_mshc.
2. Check if the card is already inserted by reading Present State Register (PSTATE_REG). This step is required for a removable card. If the card is already inserted, then goto Step 4.
3. Wait for the card insertion interrupt. This step is required for removable card. Card detection sequence is described in the “Card Detection” programming sequence.
4. Set up basic settings for DWC_mshc as described in “Host Controller Setup Sequence”. It involves settings such as the bus power voltage level for the card, timeout counter value, setting clock generation parameters. Different setup sequences for SD card (“SD Interface”) and eMMC card (“eMMC Device”) are provided. This step can also be executed immediately after Step 1.
5. Enable input clocks (other than hclk) that are available in DWC_mshc. Internal clocks (bclk and tmclk) are enabled followed by the card clocks (cclk_tx/cclk_rx) as discussed in the programming sequence in “Host Controller Clock Setup Sequence”. At this stage, DWC_mshc is ready to communicate with the card.
6. Identify the type of card that is connected to DWC_mshc as explained in the programming sequence in “SD Card Interface Detection”. If DWC_mshc is connected to an eMMC card, then programming sequence described in “eMMC Card Interface Setup” must be executed. This is a simplified sequence especially for an eMMC card as the type of card is always known. Both the programming sequences provide power and clock to the card.
7. After powering up the card, the card is initialized. Appropriate sequence as shown below must to be executed based on type of the connected card.

   Card Initialization Sequence for Different Cards

Type of Card	Card Initialization Sequence
SD	SD Card Initialization and Identification
eMMC	Initializing and Identifying an eMMC Card

8. As both host controller and card are initialized, the host can start sending commands to the card to perform the data transfer. Control and data commands are two types of commands that can be issued to the card. The control command is used to read or write any register in the card. The data command is used for writing or reading data to or from the card. The table below lists the control and data command sequences for SD and eMMC cards.

   a) Data transfer can be ed using command.

   b) If errors are detected during the transfer, recovery should be done using the Error Recovery sequence.

   Command Sequence for Different Cards

Type of Card	Command Sequence	
SD/eMMC	Control CMD	Issuing CMD without Data Transfer
	Data CMD	Issuing CMD with Data Transfer

46.2.2.3 Card Detection

Figure 748 shows the sequence for detecting a card. The procedure outlined in **Figure 748** applies to both SD and SDIO cards. This programming sequence is not required for an eMMC device as it is non-removable.

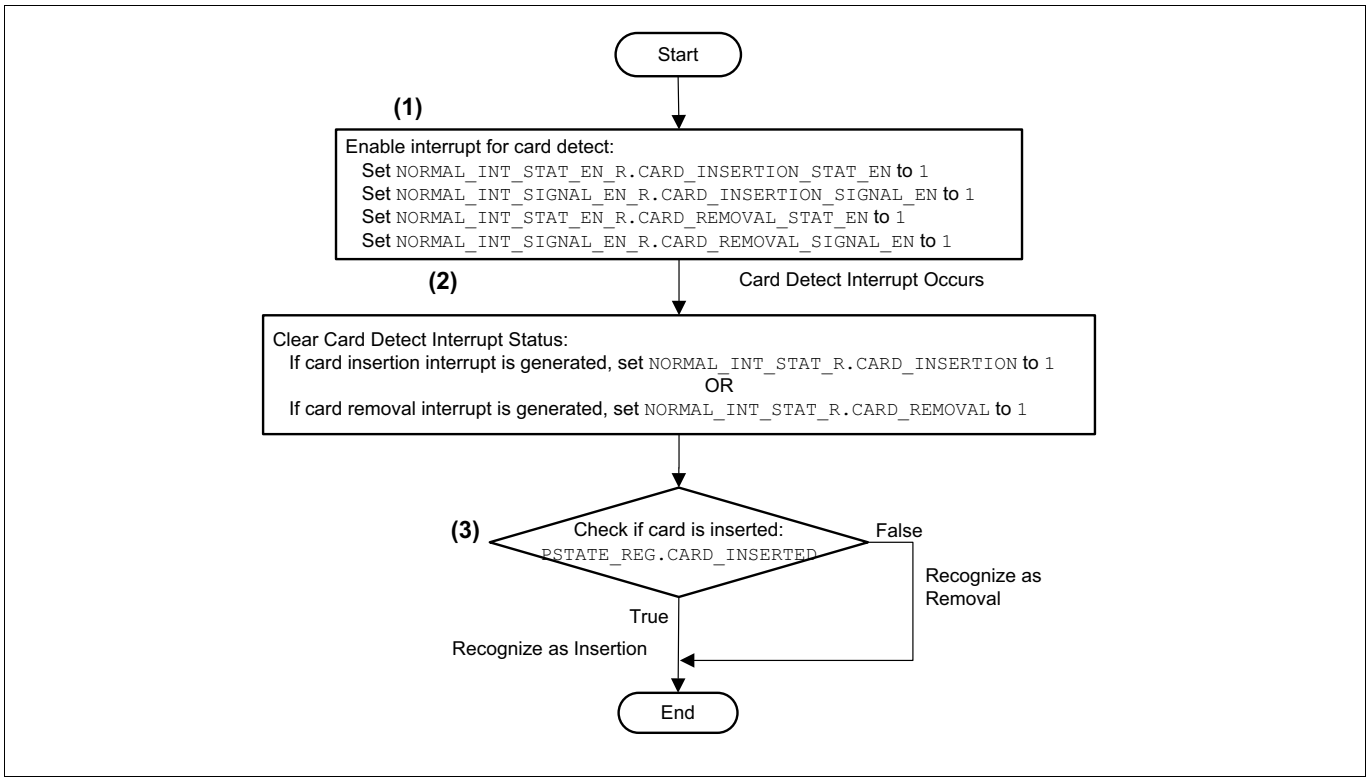


Figure 748 Card Detection

46.2.2.4 Host Controller Setup Sequence

This section discusses the host controller setup sequence for SD and an eMMC device. This section discusses the following sequences:

- **Host Controller Setup Sequence for an SD Interface** on page **Page 9**
- **Host Controller Setup Sequence for an eMMC Device** on page **Page 10**

46.2.2.4.1 Host Controller Setup Sequence for an SD Interface

The host controller setup sequence for an SD interface involves basic settings for DWC_mshc. The procedure involves settings such as the bus power voltage level for the card, timeout counter value, setting clock generation parameters.

Figure 749 shows the host controller setup sequence. The register parameters set in this sequence retain their values irrespective of bus speed mode changes.

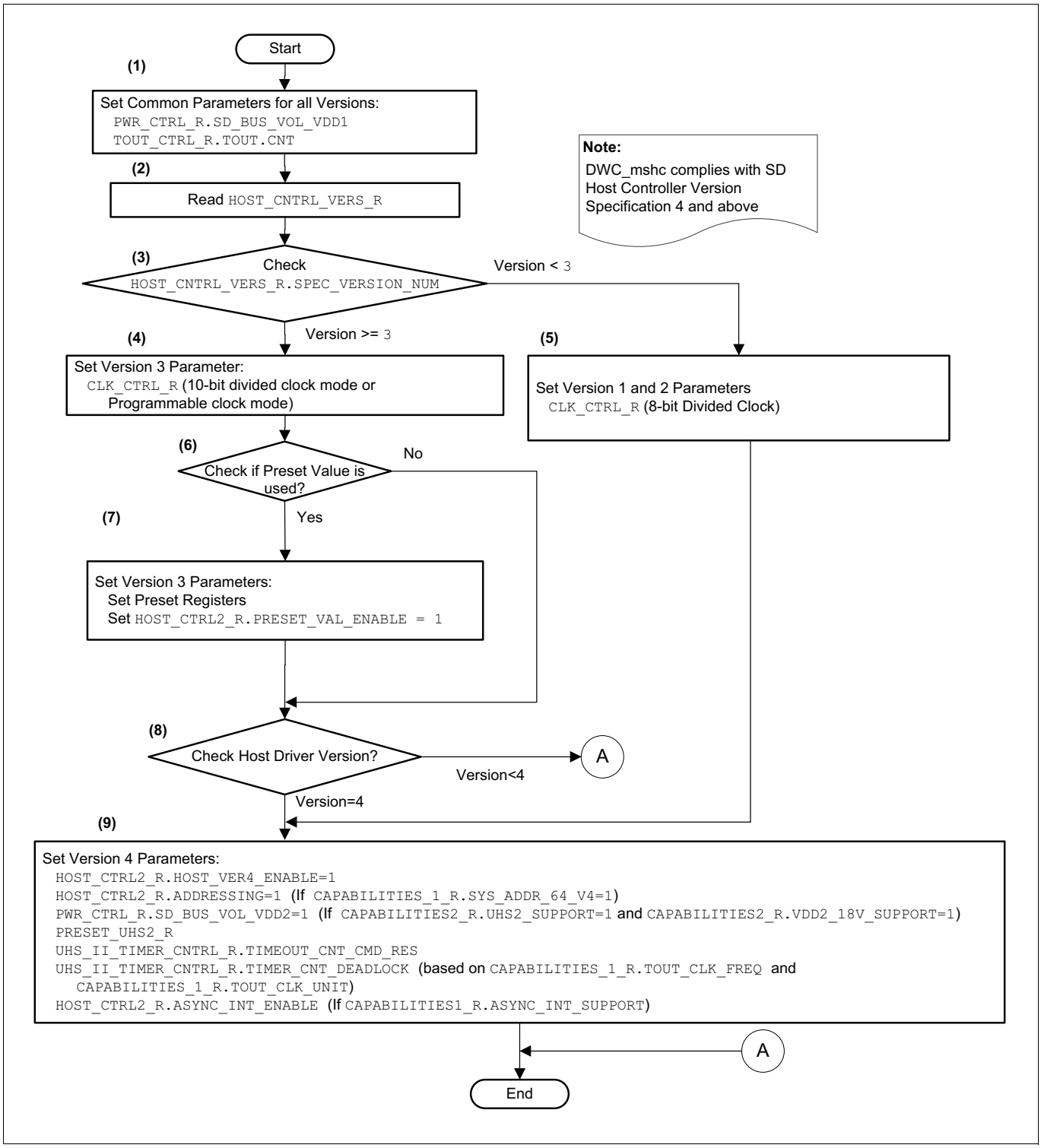


Figure 749 Host Controller Setup Sequence for SD Interface

46.2.2.4.2 Host Controller Setup Sequence for an eMMC Device

The host controller setup sequence for an eMMC device involves basic settings for DWC_mshc. The procedure involves setting certain parameters common to both version 3 and 4, clock control register, and version 4 parameters.

Figure 750 shows the host controller setup sequence for an eMMC device. In addition to completing the programming sequence in Figure 4-3, you must set the parameter EMMC_CRTL_R.CARD_IS_EMMC = 1.

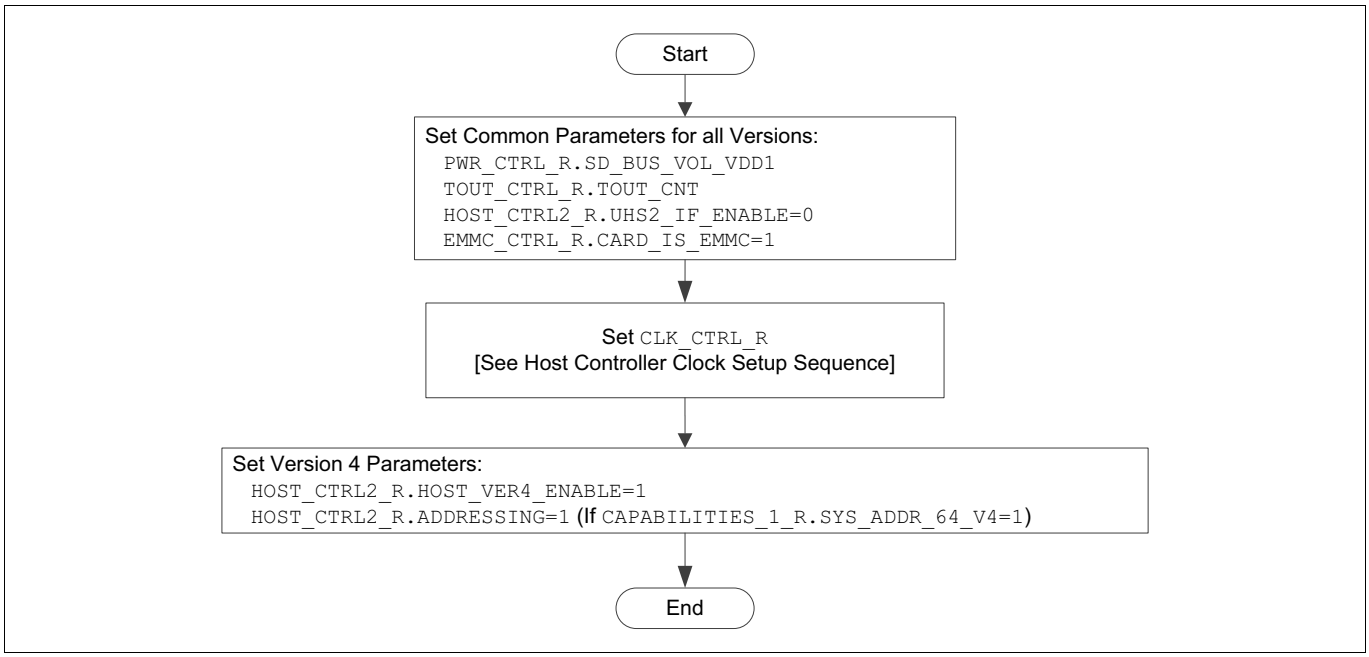


Figure 750 Host Controller Setup Sequence for eMMC Interface

46.2.2.5 Clock Control

This section discusses the programming sequence for setting up internal clocks and card clock provided to DWC_mshc and for supplying and setting up a clock provided to the card, and for changing the card clock frequency.

This section includes the following sequences:

- **Host Controller Clock Setup Sequence** on page **Page 11**
- **Card Clock Supply and Stop Sequence** on page **Page 12**
- **SD Clock Frequency Change Sequence** on page **Page 12**

46.2.2.5.1 Host Controller Clock Setup Sequence

The DWC_mshc requires number of clock signals from the system. All the input clocks are asynchronous to each other. Based on controls available for enabling clocks required for DWC_mshc, they are categorized into two types, namely Internal clock and Card clock.

Figure 751 shows the sequence for setting up clocks for DWC_mshc.

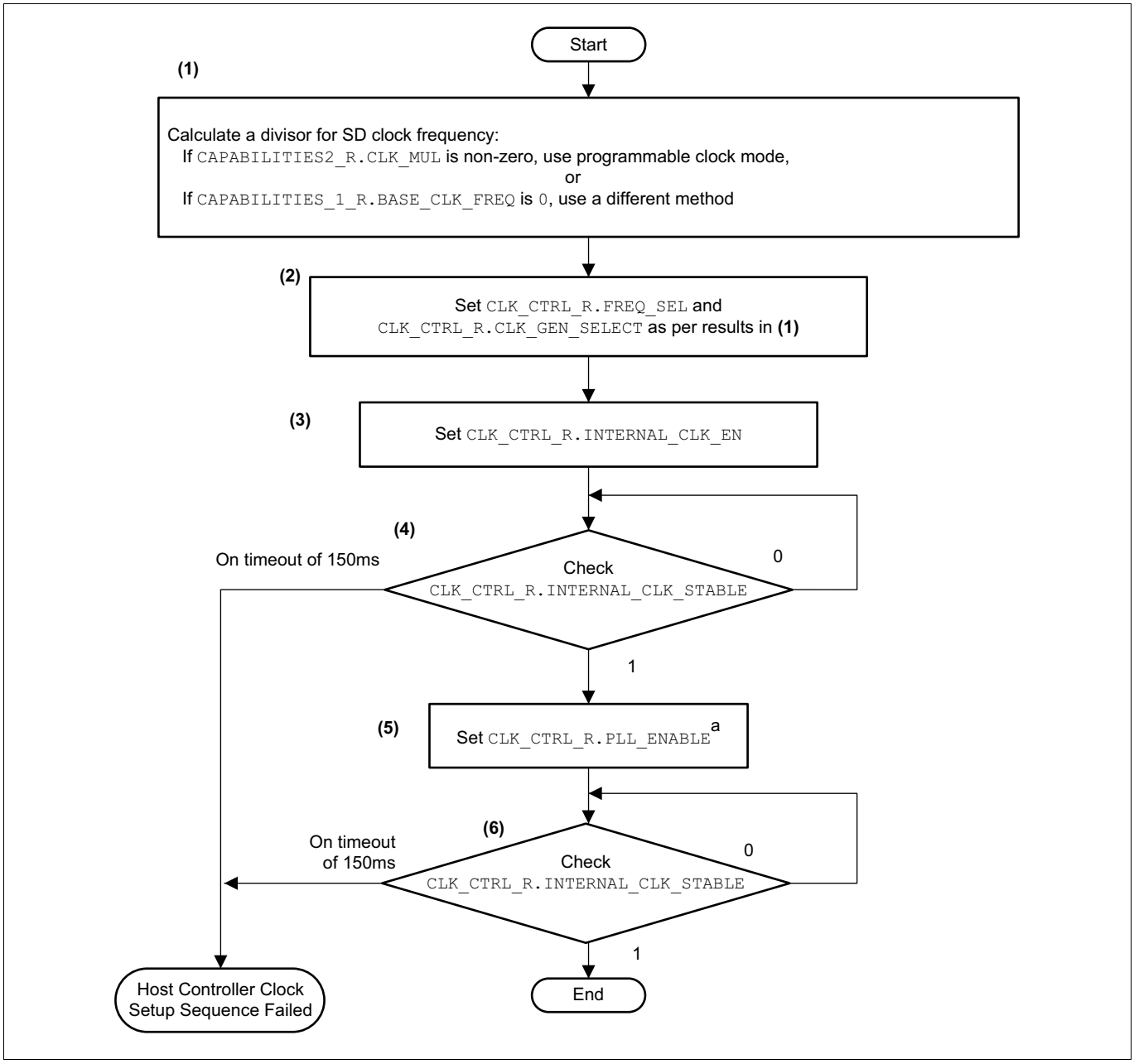


Figure 751 Host Controller Clock Setup Sequence

[a] Step to be ignored as there is no PLL.

46.2.2.5.2 Card Clock Supply and Stop Sequence

A clock Supply and Stop Sequence can be initiated, only when there is no active data transfer between the DWC_mshc and the card.

Figure 752 shows the flow chart for stopping the clock to card. The procedure outlined in Figure 4-5 applies to both SD and eMMC cards.

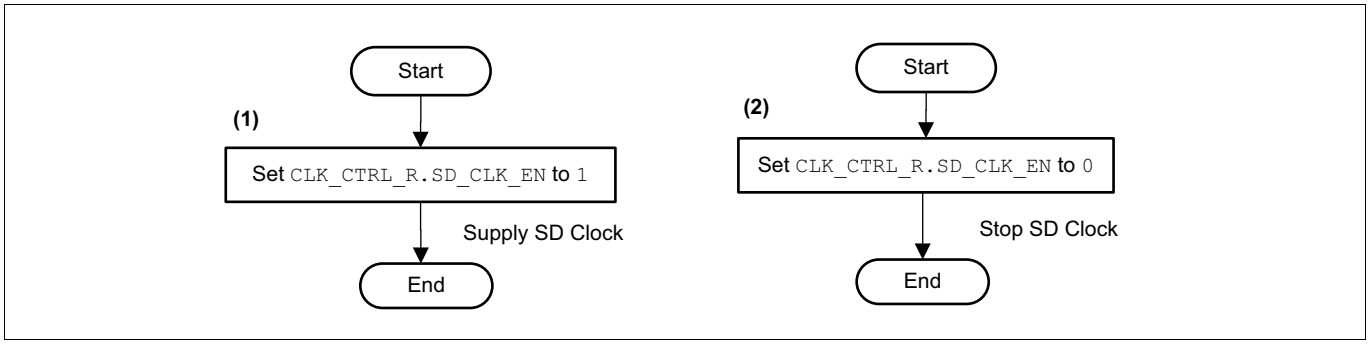


Figure 752 Card Clock Supply and Stop Sequence

46.2.2.5.3 SD Clock Frequency Change Sequence

The method used to change the SD clock frequency can either be a clock multiplier of programmable clock generator, or any different mode (if the Base Clock Frequency bit in the Capabilities register is 0).

Figure 753 shows the sequence for changing SD clock frequency.

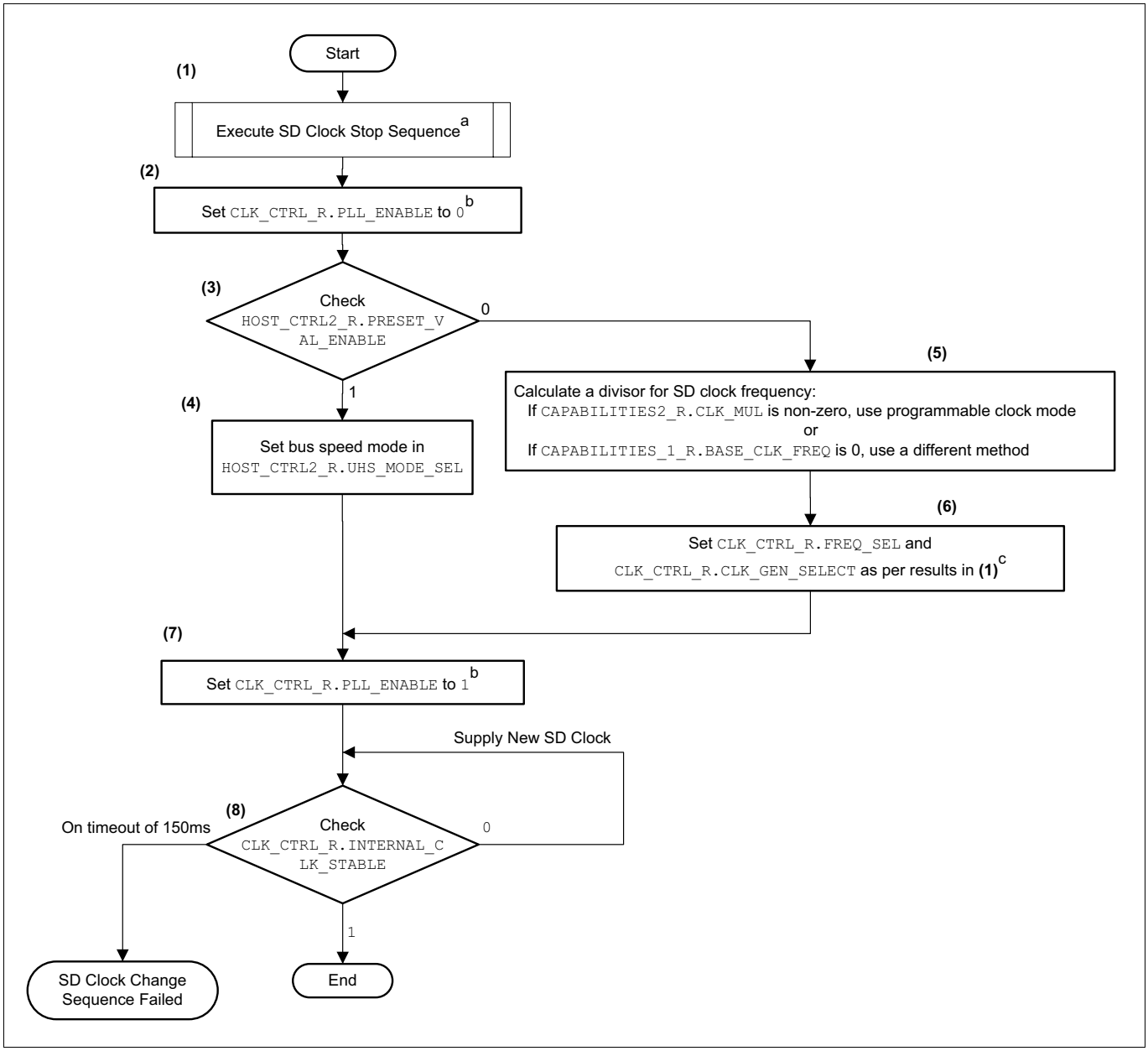


Figure 753 SD Interface Clock Frequency Change Sequence

[a] Refer "**Card Clock Supply and Stop Sequence**" on **Page 12**.

[b] Step to be ignored as there is no PLL.

[c] Step to be ignored as Clock Multiplier option is not supported.

46.2.2.6 Card Interface Setup Sequence

This section discusses the programming sequence for setting up the card interface for an SD, UHS-II and eMMC cards.

This section discusses the following topics:

46.2.2.6.1 SD Card Interface Setup Sequence

The SD card interface setup sequence involves the procedure to detect the SD card, set the necessary SD register bits, change the SD clock frequency, supply the SD clock and perform SD 4-bit mode initialization.

Figure 754 shows the programming sequence for detecting and setting up the SD card interface.

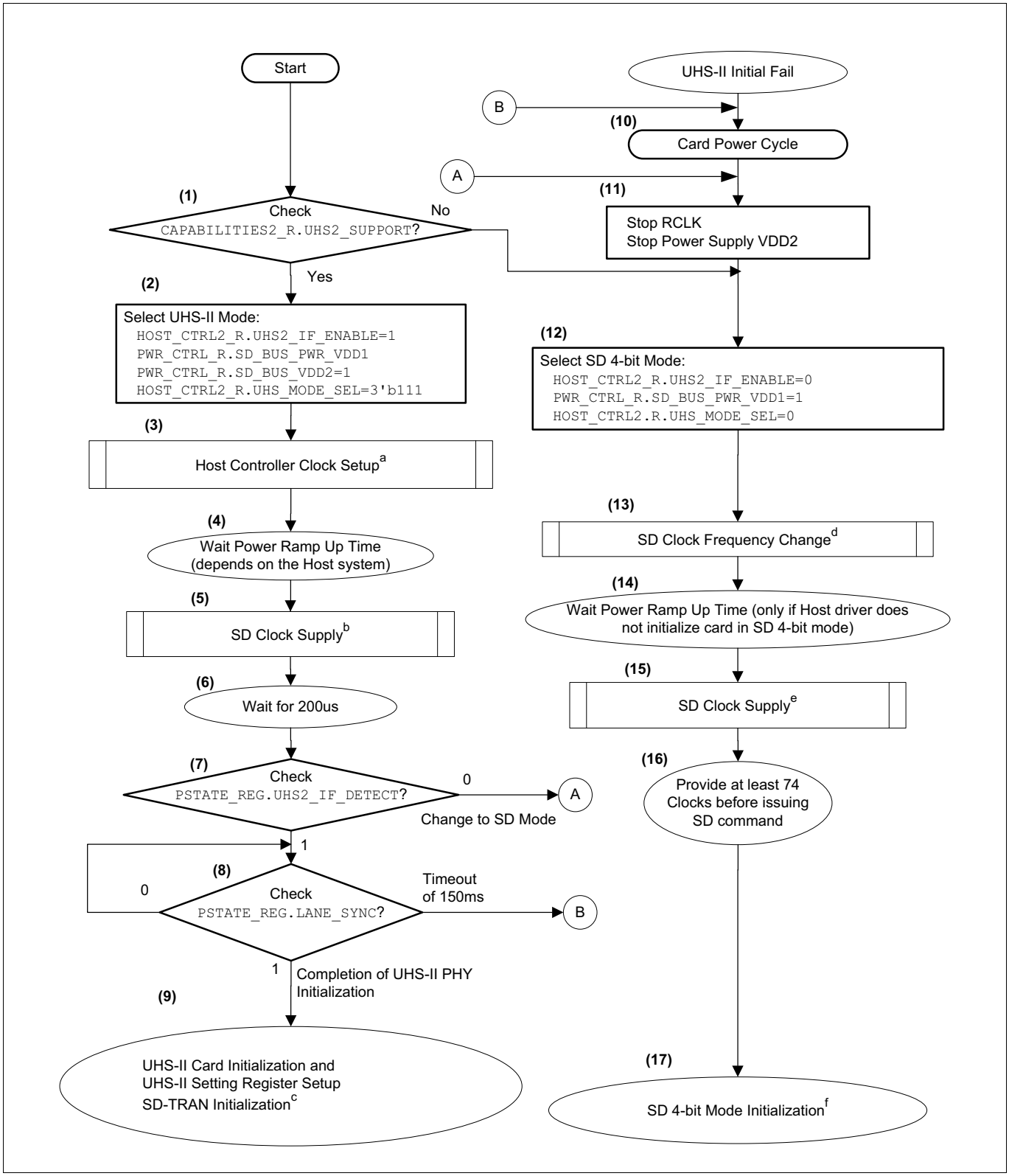


Figure 754 SD Card Interface Detection Sequence

[a] Refer "**Host Controller Clock Setup Sequence**" on **Page 11**.

[b e] Refer "**Card Clock Supply and Stop Sequence**" on **Page 12**.

[c f] Refer "**SD Card Initialization and Identification**" on **Page 21**.

[d] Refer "**SD Changing Bus Speed Mode**" on **Page 32**.

46.2.2.6.2 eMMC Card Interface Setup

The eMMC card interface setup sequence involves the procedure to apply the power to the bus, change the eMMC clock frequency, supply the SD clock and finally, perform the card initialization.

Figure 755 shows the programming sequence to set up an eMMC device.

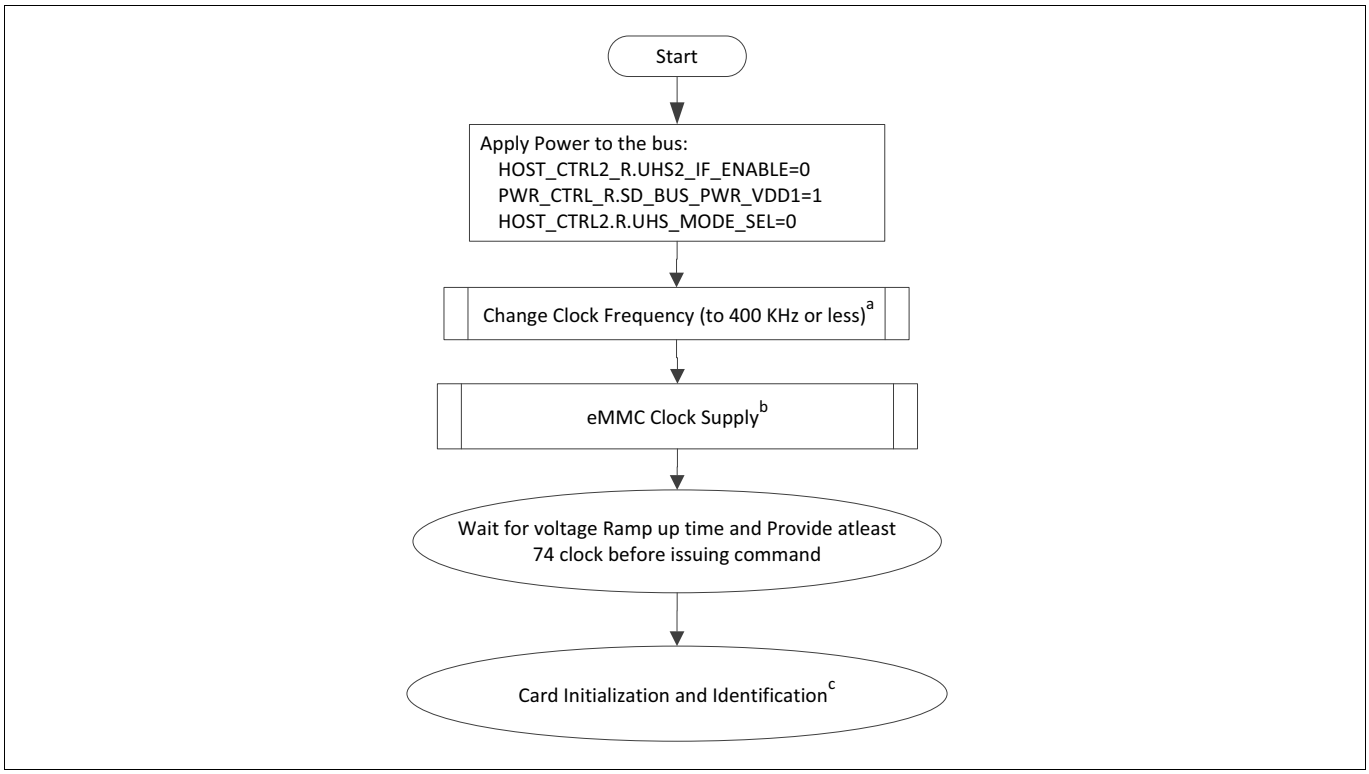


Figure 755 eMMC Card Interrupt Setup Sequence

[a] Refer the programming sequence "**SD Changing Bus Speed Mode**" on **Page 32**.

[b] Refer the programming sequence "Refer "**Card Clock Supply and Stop Sequence**" on **Page 12**."

[c] Refer the programming sequence **SD Card Initialization and Identification**" on **Page 21**.

46.2.2.7 Timeout Setting for an SD/eMMC Bus

To detect timeout errors, DWC_mshc must calculate a divisor using the Capabilities register, and as per the divisor value, set the data timeout counter value.

Figure 756 shows the timeout setting for an SD/eMMC bus.

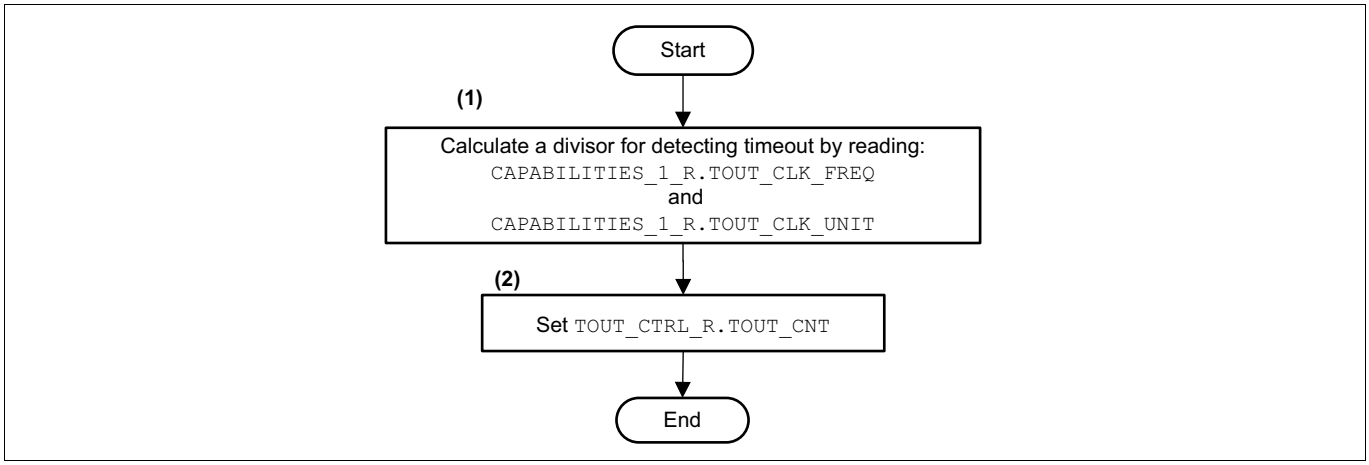


Figure 756 Timeout Setting Sequence for an SD/eMMC Bus

46.2.2.8 Abort Transaction

An abort command to the card is issued to abort an ongoing data transaction. To abort a transaction on an SD or eMMC card, CMD12 is issued, whereas to abort a transaction on an SDIO card, CMD52 is issued.

Following are the instances when an abort command must be issued:

- To stop an infinite block transfer
- To stop transfer due to a request from an application
- To stop a transfer due to an error detection

The sequence to issue an abort command is shown in “Abort Command Sequence”. An abort command can be issued synchronously or asynchronously during data transfer. In asynchronous abort, abort command is issued at any time irrespective of the state of data transfer as a subcommand.

Synchronous abort ensures that the data transfer is stopped (at block gap) before issuing an abort command. Both methods are explained in “Asynchronous Abort” and “Synchronous Abort” respectively.

This section discusses the following sequences:

- **Abort Command Sequence** on page **Page 17**
- **Asynchronous Abort** on page **Page 18**
- **Synchronous Abort** on page **Page 19**

46.2.2.8.1 Abort Command Sequence

Figure 757 shows the abort command sequence.

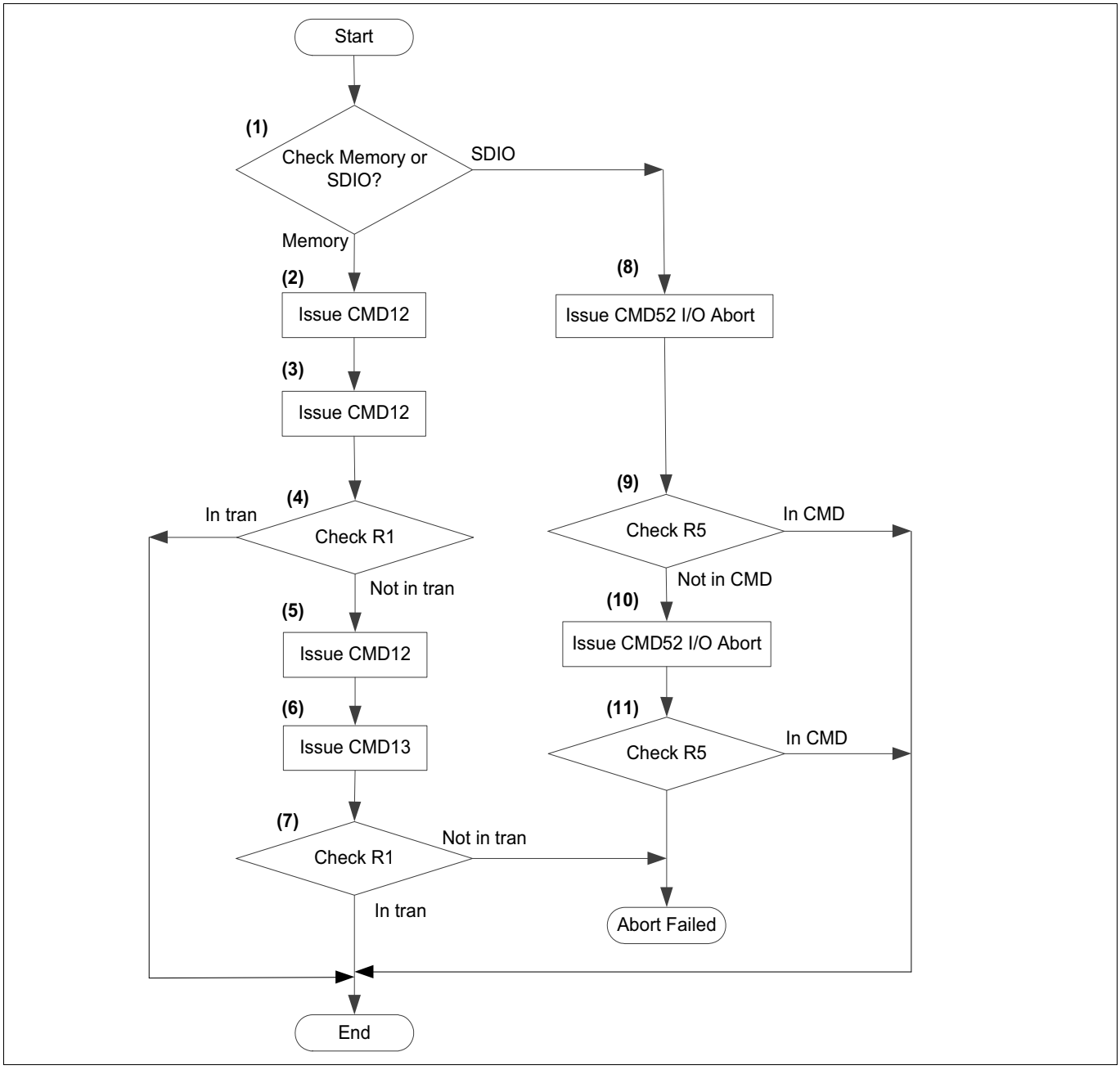


Figure 757 Abort Command Sequence

46.2.2.8.2 Asynchronous Abort

Figure 758 shows the asynchronous abort.

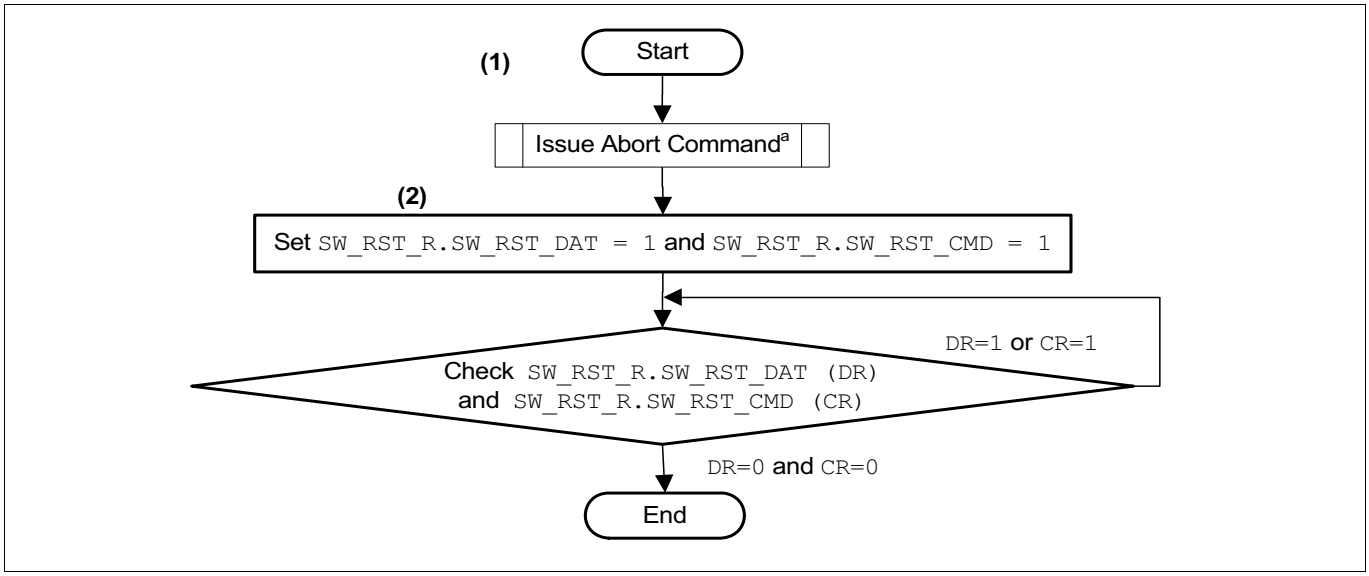


Figure 758 Asynchronous Abort Sequence

[a] Refer "**Abort Command Sequence**" on **Page 17**.

46.2.2.8.3 Synchronous Abort

Figure 759 shows the synchronous abort.

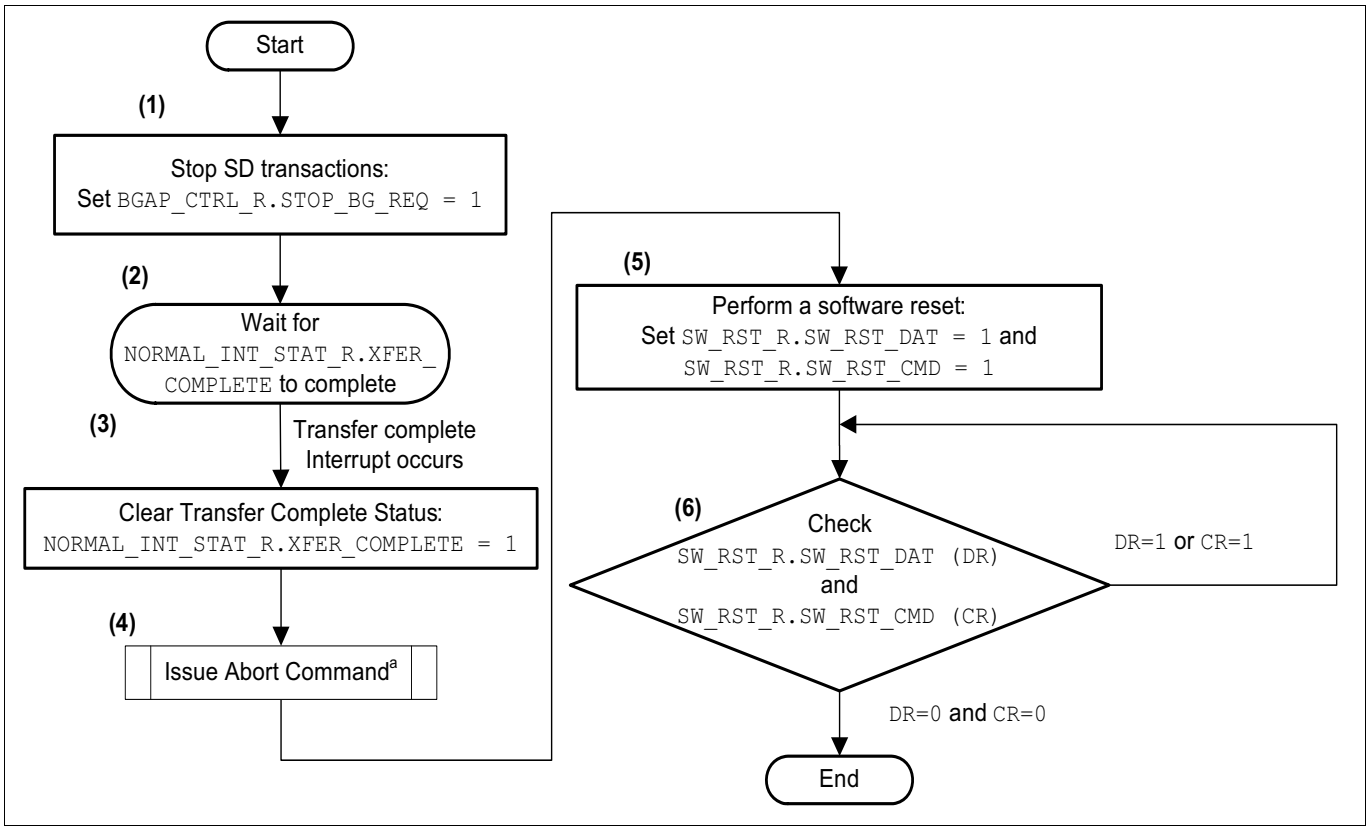


Figure 759 Synchronous Abort Sequence

[a] Refer "**Abort Command Sequence**" on **Page 17**.

46.2.2.9 SD/SDIO Transaction Mode

This section includes the sequences to generate and control different types of SD/SDIO transactions.

This section discusses the following programming sequences:

46.2.2.9.1 SD Card Initialization and Identification

After powering up the SD card, it must be initialized so that the host can start sending commands to the card to perform the data transfer.

Figure 760 and **Figure 761** show the sequence for SD card initialization and identification.

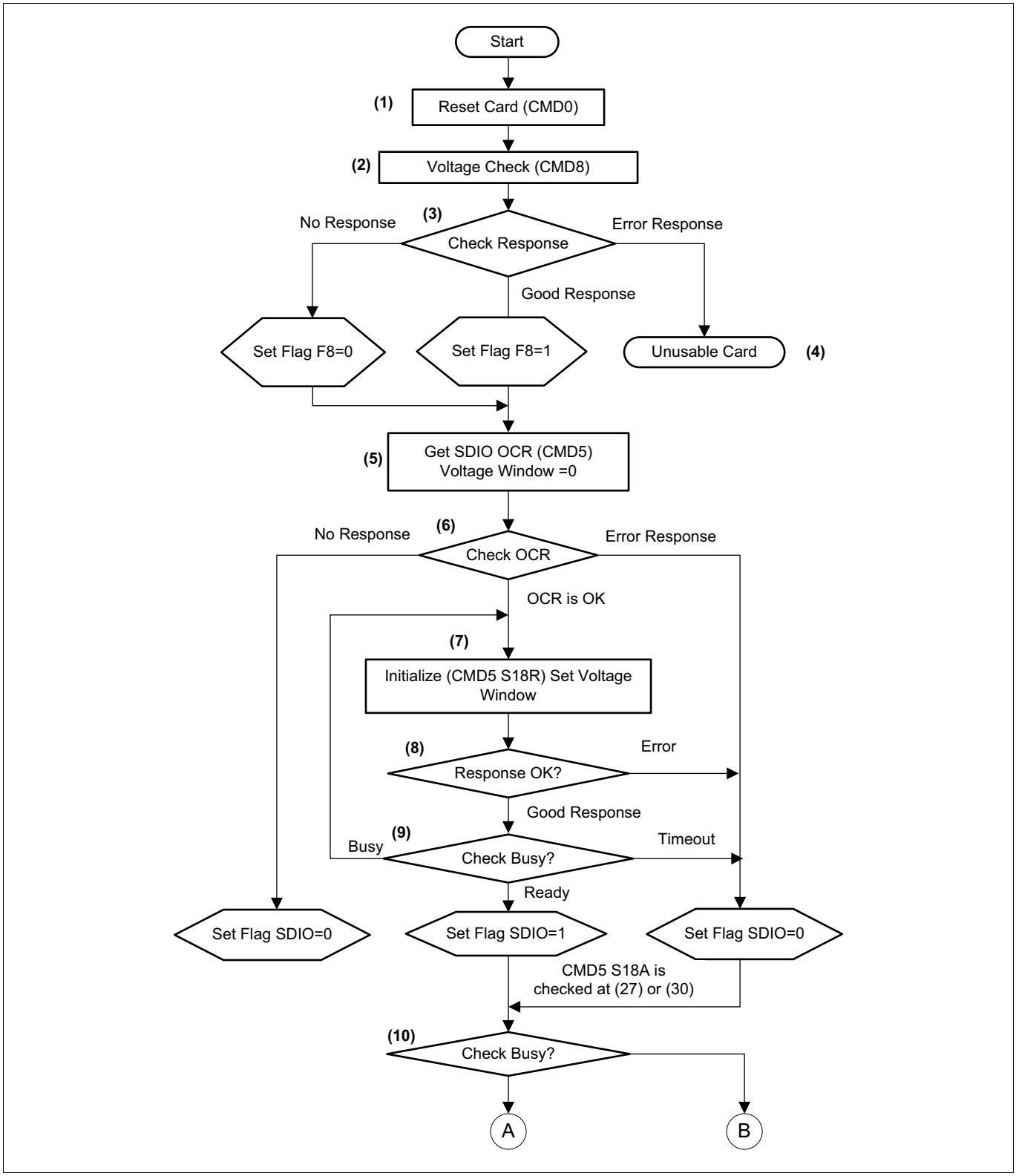


Figure 760 SD Card Initialization and Identification Part 1

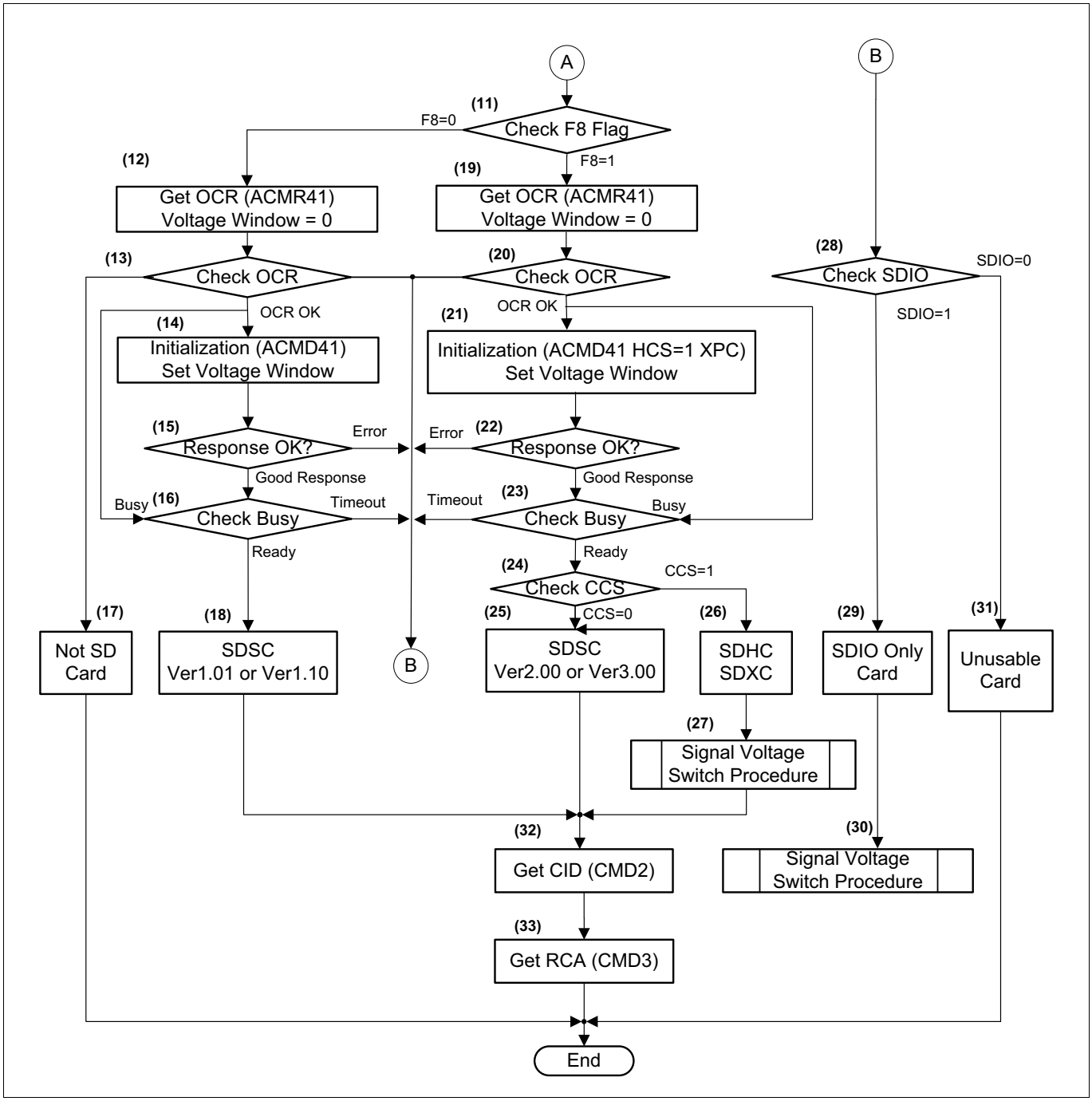


Figure 761 SD Card Initialization and Identification Part 2

Note: During the initialization sequence if the card supports the Read Wait feature, it is recommended that the Read Wait feature be enabled in DWC_mshc by programming the RD_WAIT_CTRL field in BGAP_CTRL_R register.

46.2.2.9.2 Changing SD Bus Width

Based on the data width of the SD card, the host driver must change the data width of the Host controller to 1-bit or 4-bit.

Figure 762 shows the sequence for changing data bus bit mode on an SD Bus.

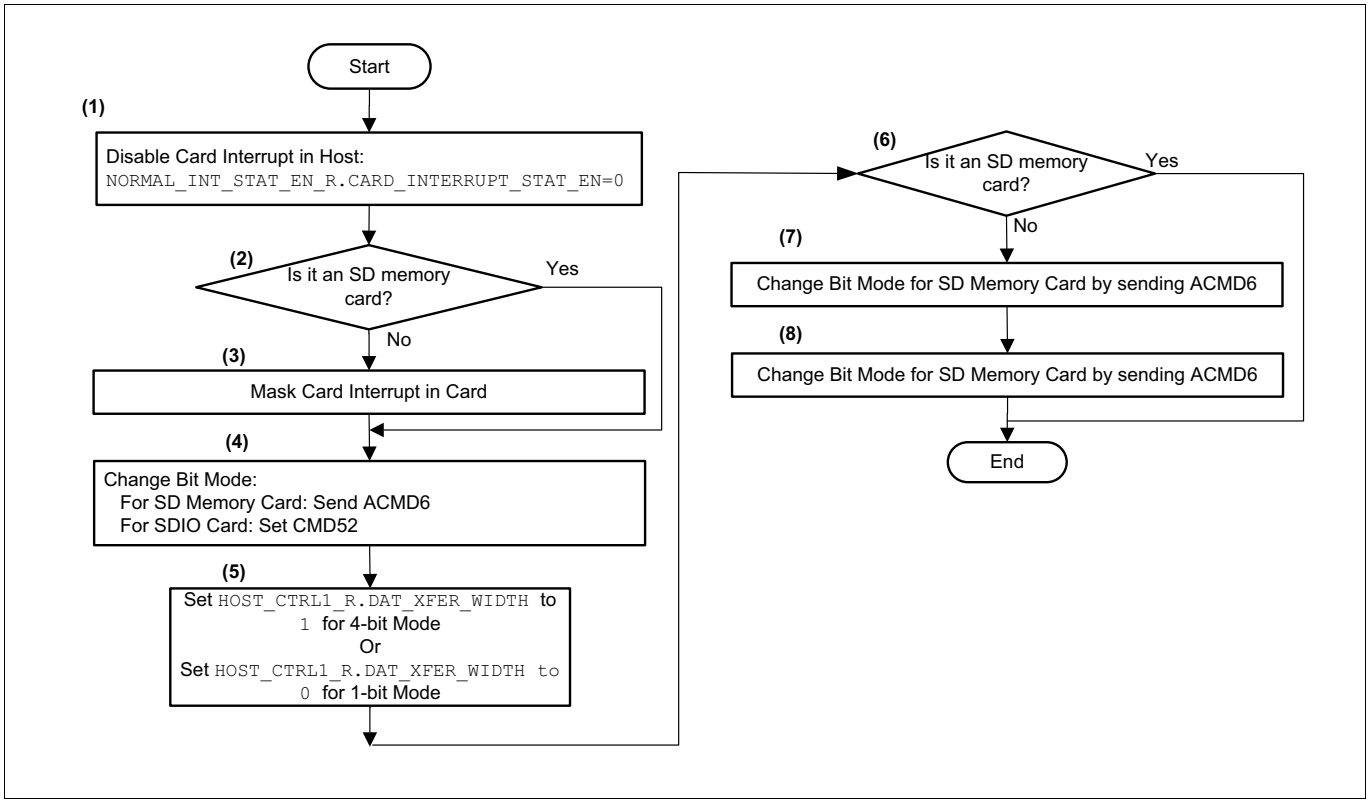


Figure 762 Changing SD Bus Width Sequence

46.2.2.9.3 SD Bus Power Control

In order to control the bus power of the card, the maximum voltage that the Host controller supports must be set and this voltage is selected by the host driver for an SD card.

Figure 763 shows the sequence for controlling the SD Bus Power.

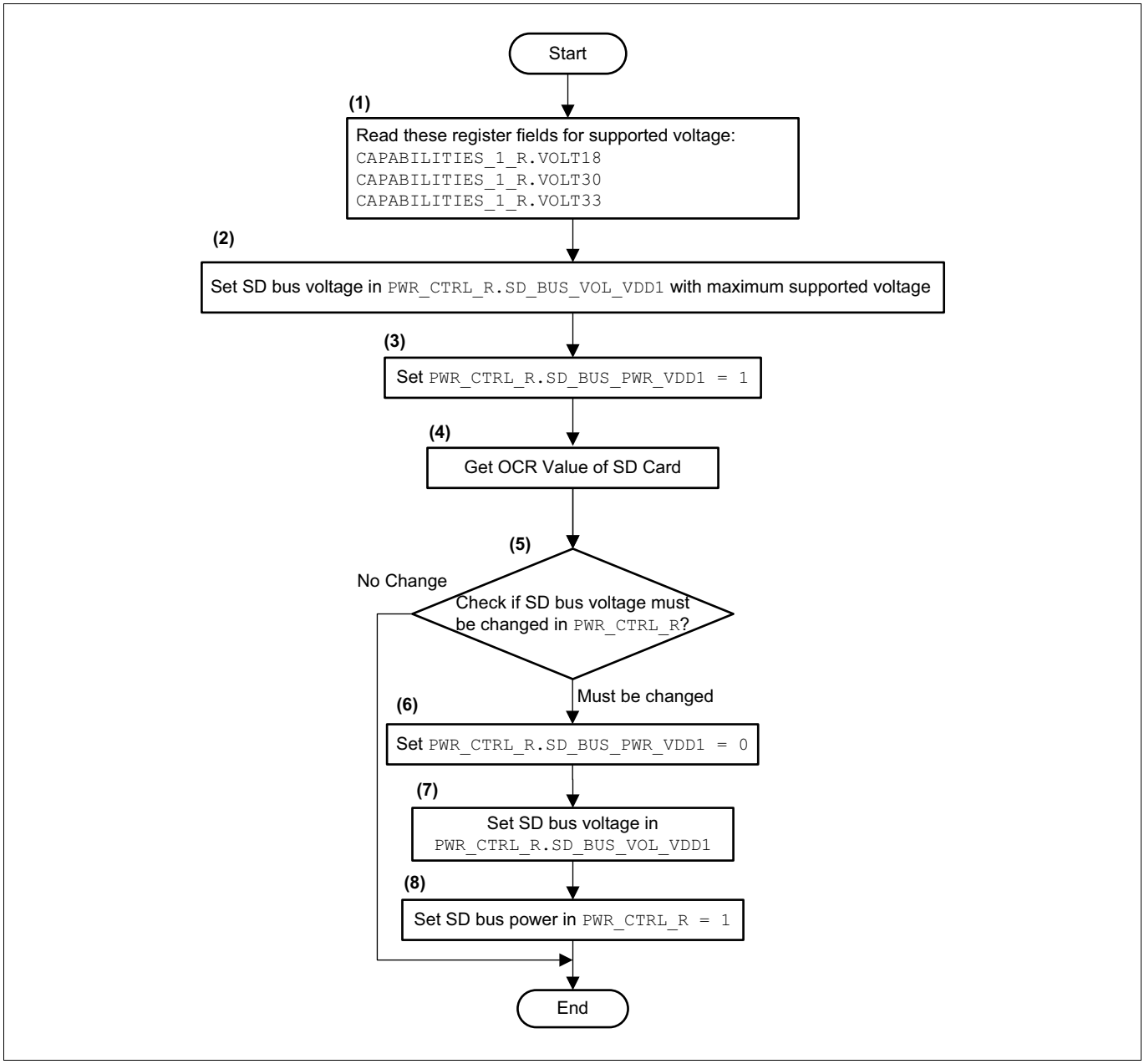


Figure 763 SD Bus Power Control Sequence

46.2.2.9.4 Issuing CMD Without Data Transfer

An SD command may or may not use the data line (DAT) for data transfer.

Figure 764 shows the sequence on how to issue and complete an SD command using the data line and without transferring any data.

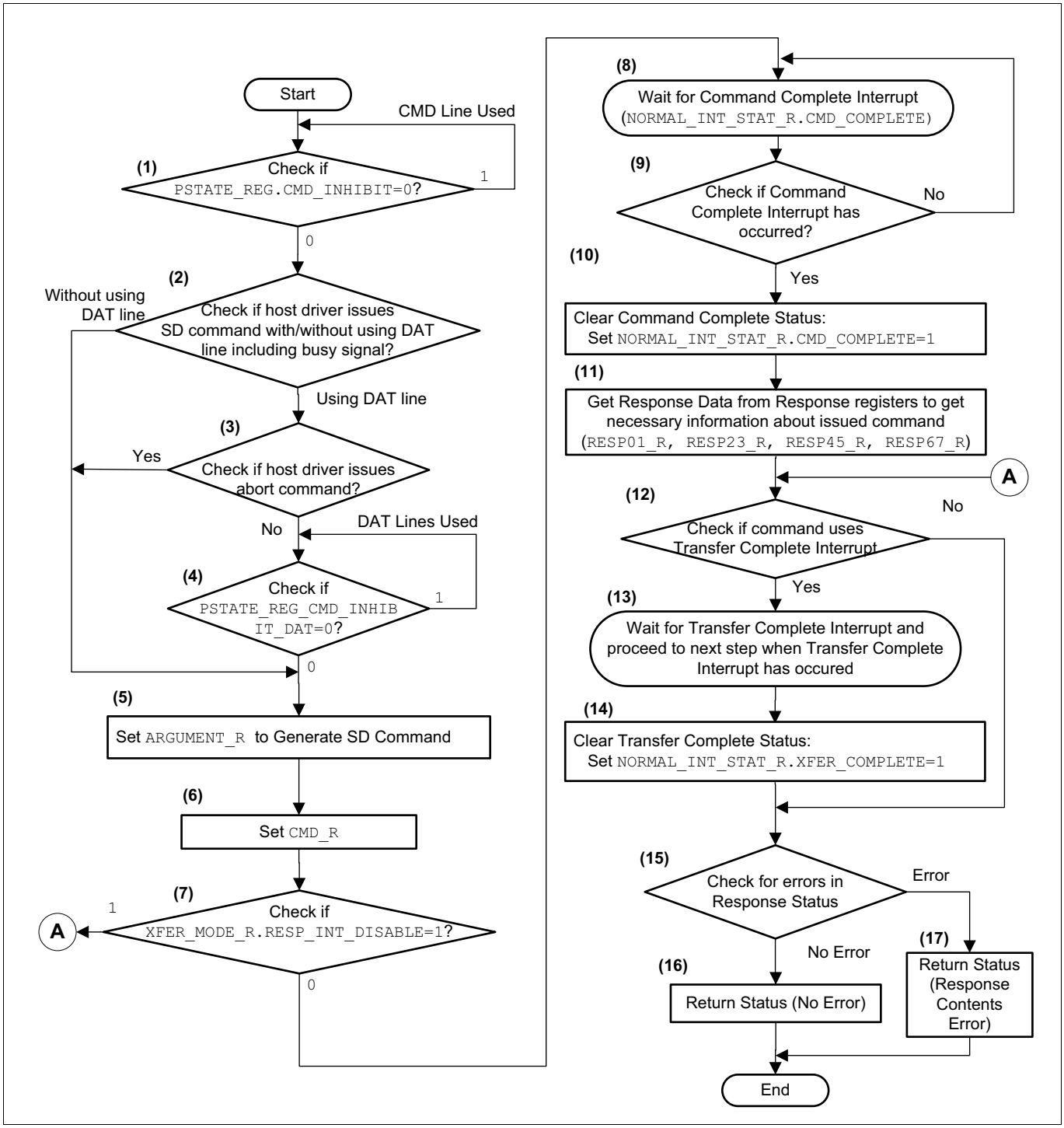


Figure 764 SD Command Issue and Complete

46.2.2.9.5 Issuing CMD with Data Transfer (Not Using DMA/PIO)

An SD command can perform data transfer on a DAT line with or without using DMA/PIO method.

Figure 765 and **Figure 766** show the transaction that does not use DMA or Programmed Input/Output (PIO) method for data transfer using the DAT line.

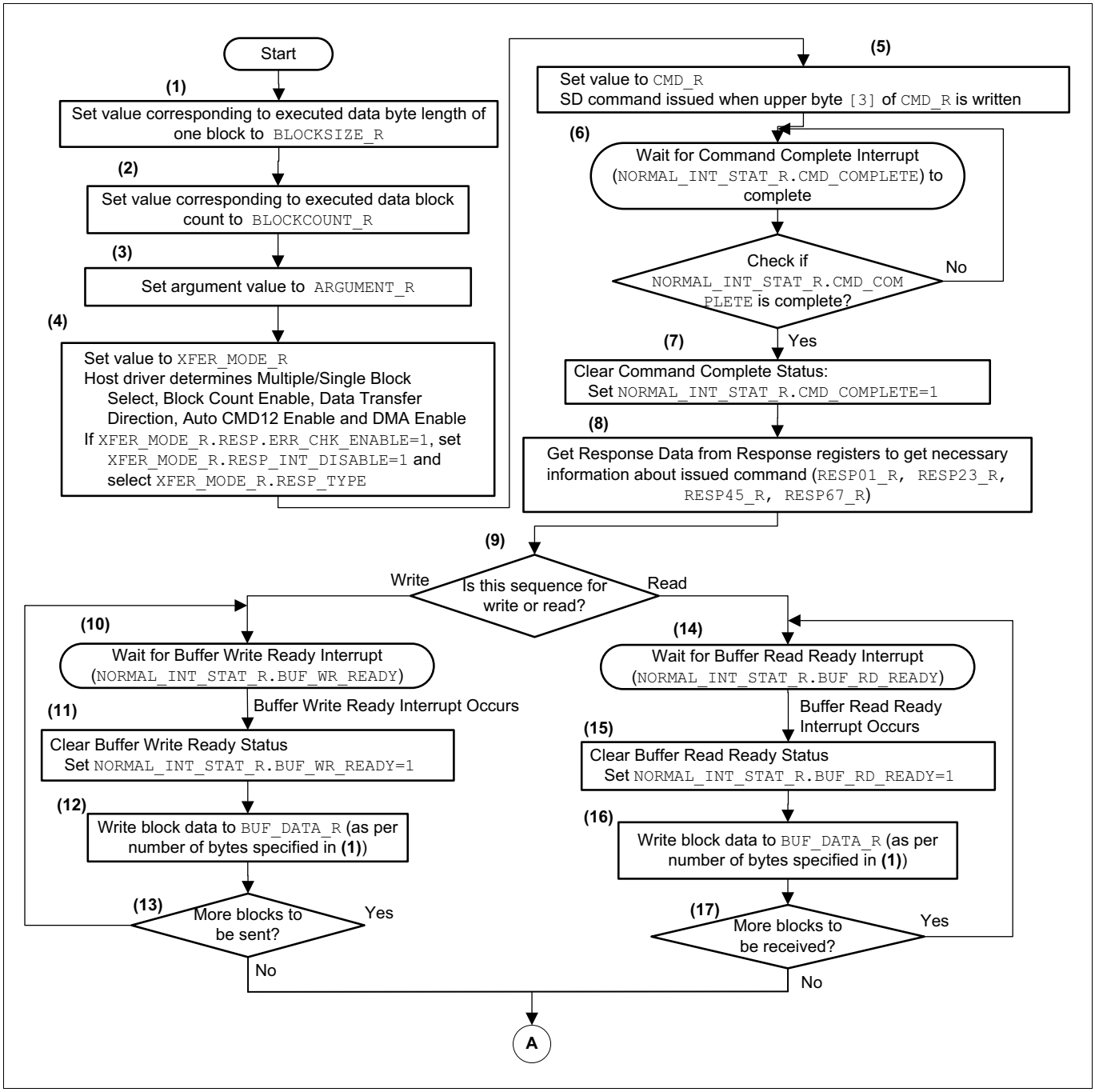


Figure 765 Transaction Control with Data Transfer Using DAT Line (Not Using DMA/PIO) Part 1

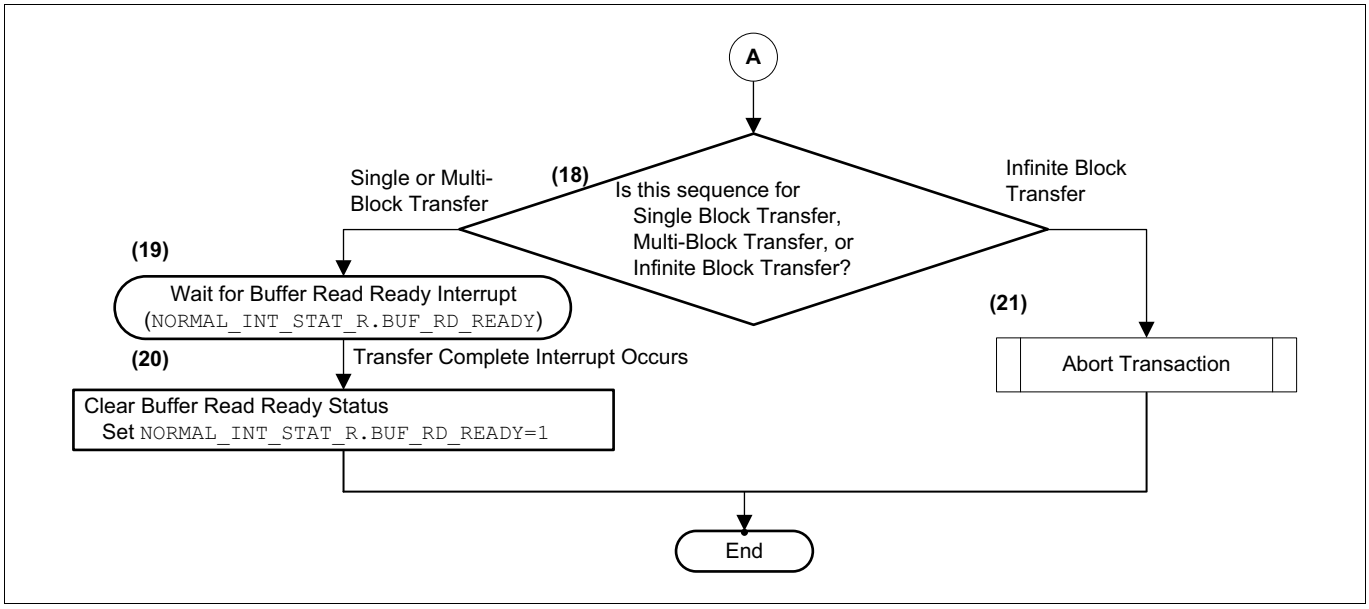


Figure 766 Transaction Control with Data Transfer Using DAT Line (Not Using DMA/PIO) Part 2

46.2.2.9.6 Issuing CMD with Data Transfer (Using SDMA)

When an SD command uses Single operation DMA (SDMA) for data transfer using the DATA line, only single SD command transaction can be executed for each SDMA operation. SDMA is appropriate for short data transfer.

Figure 767 shows the transaction that uses SDMA for data transfer using the DAT line.

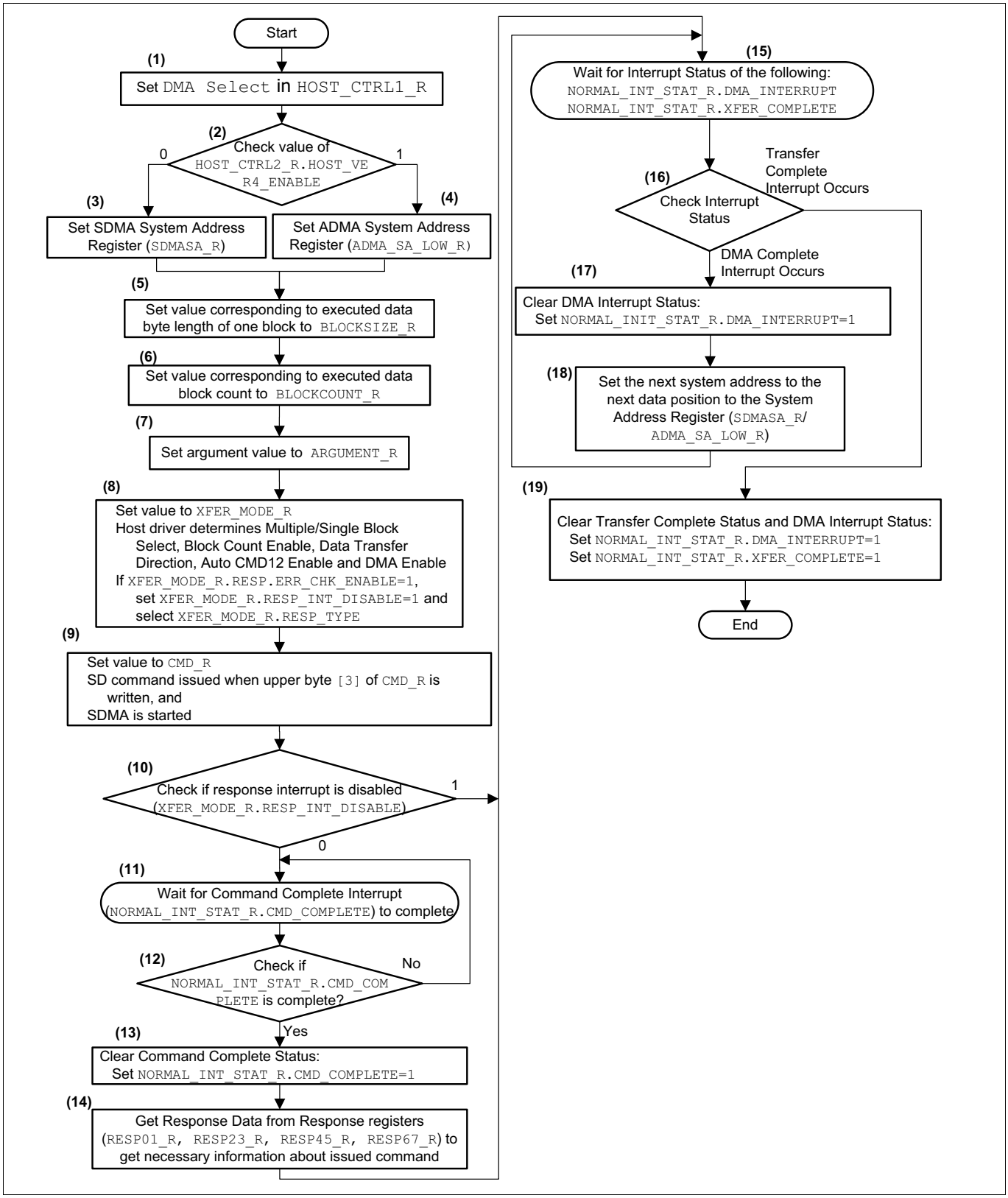


Figure 767 Transaction Control with Data Transfer using SDMA

46.2.2.9.7 Issuing CMD with Data Transfer (Using ADMA2)

ADMA2 is one of the DMA transfer types that can be used for lengthy data transfers. DWC_mshc supports the ADMA2 algorithm for data transfer between the system memory and the SD card without interrupting CPU operation. Therefore, ADMA is appropriate for lengthy data transfers. ADMA2 uses the scatter gather DMA algorithm to obtain high data transfer speed.

Figure 768 and **Figure 769** show the transaction that uses ADMA2 for data transfer using the DAT line.

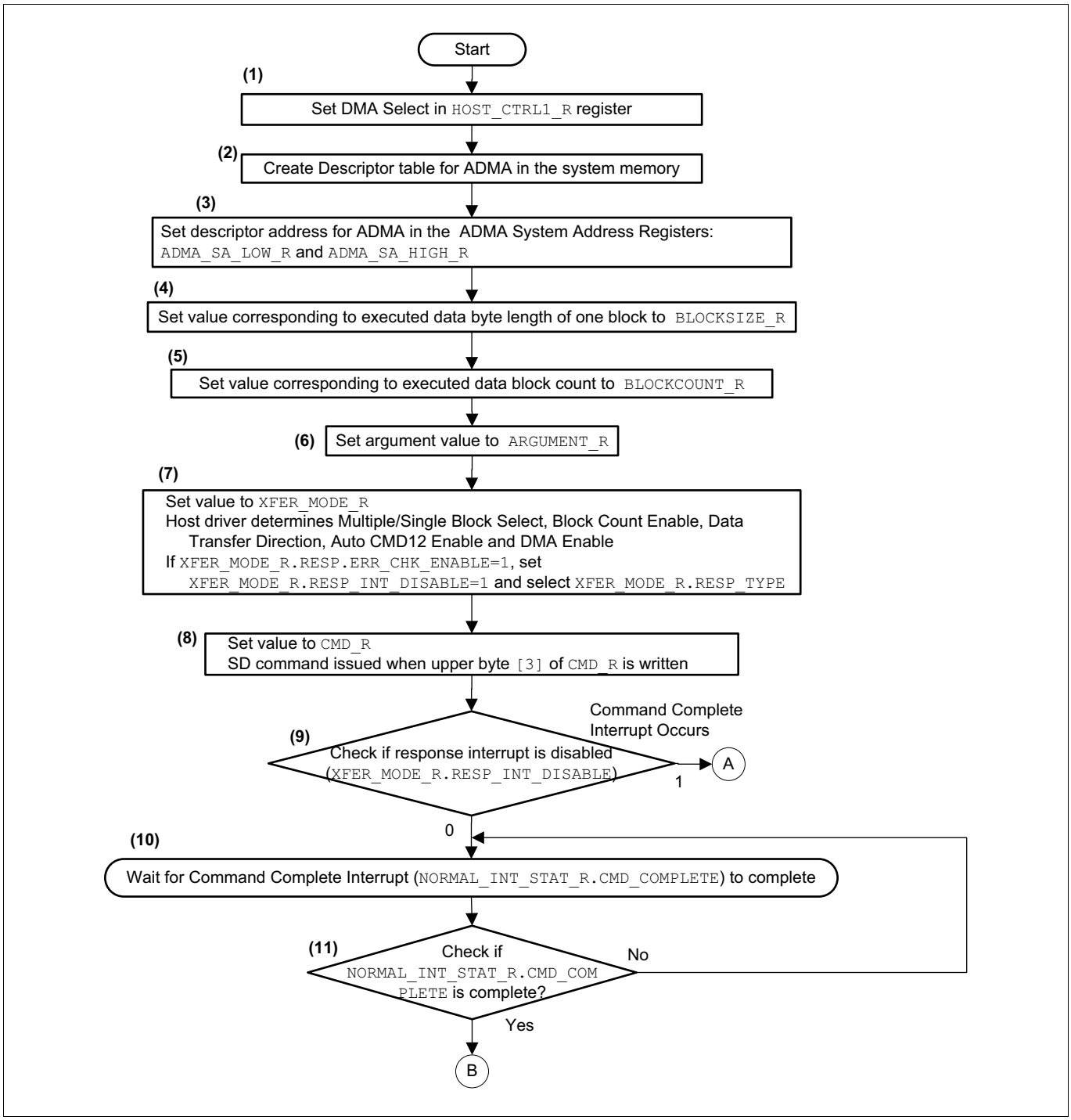


Figure 768 Transaction Control with Data Transfer Using DAT Line (Using ADMA2) Part 1

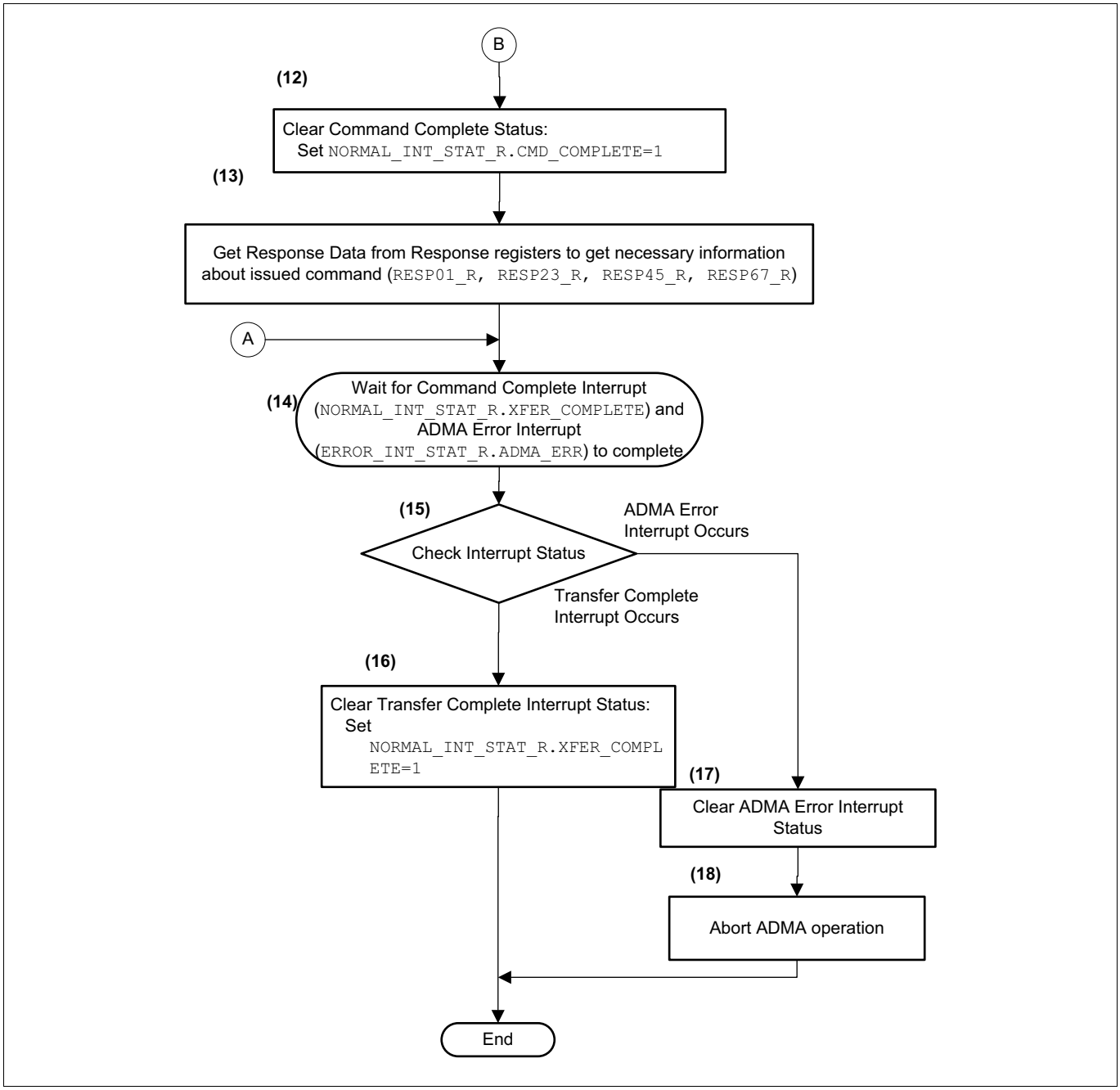


Figure 769 Transaction Control with Data Transfer Using DAT Line (Using ADMA) Part 2

46.2.2.9.8 Issuing CMD with Data Transfer (Using ADMA3)

ADMA3 is one of the DMA transfer types that can be used for very lengthy data transfers. ADMA3 performs multiple read/write SD command operations at a time and uses the scatter gather DMA algorithm to obtain high data transfer speed.

Figure 770 shows the transaction that uses ADMA3 for data transfer using the DAT line.

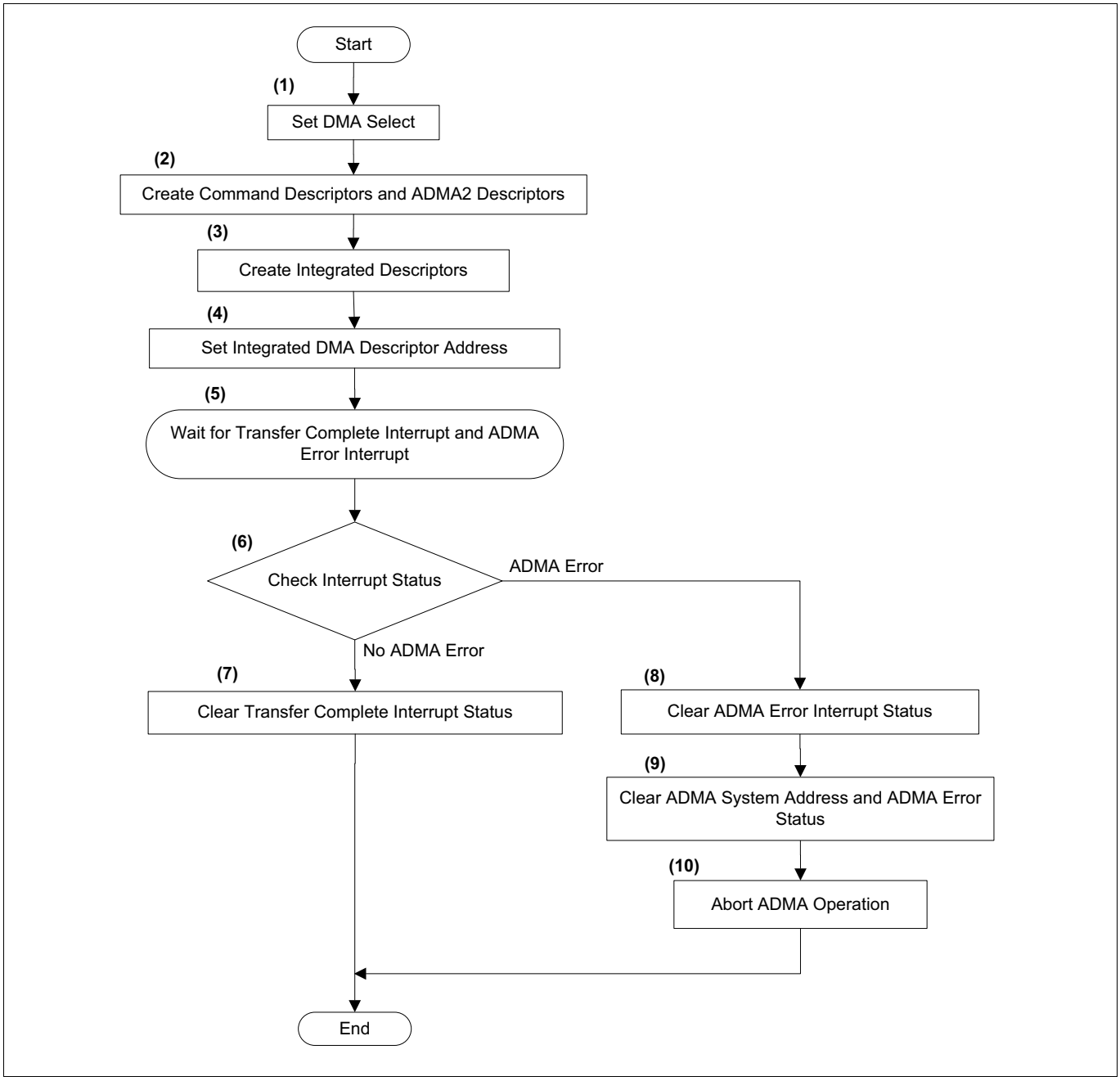


Figure 770 Transaction Control with Data Transfer Using DAT Line (Using ADMA3)

46.2.2.9.9 SD Changing Bus Speed Mode

The SD bus speed mode of the controller must be changed to match the host driver speed mode. The switch command (CMD6) is used to change the bus speed mode. The different bus speed modes are default speed (DS) mode and high speed (HS) mode.

Figure 771 shows the procedure to change the bus speed mode for an SD card.

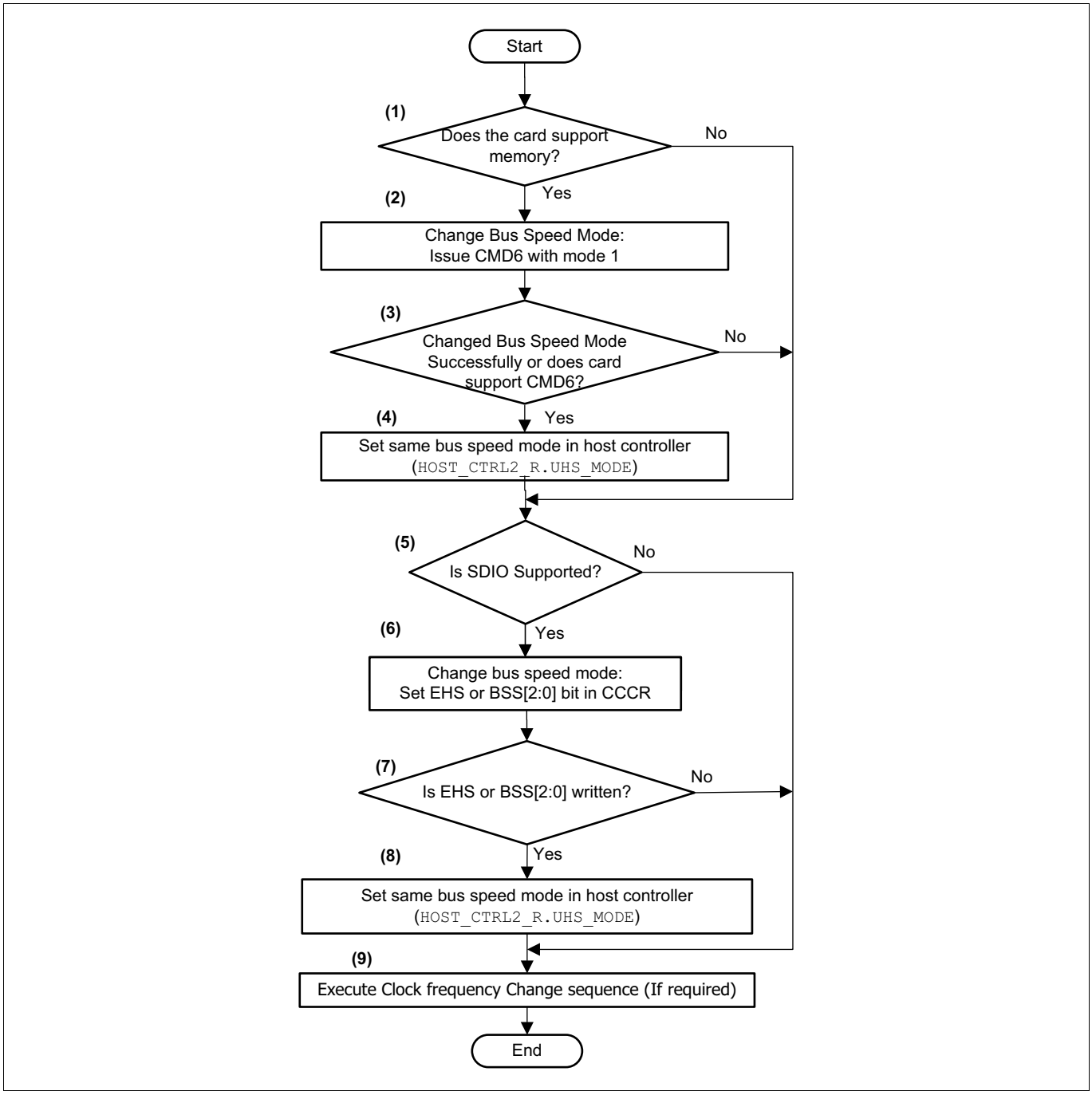


Figure 771 Changing Bus Speed Mode for SD Card

46.2.2.9.10 SDIO Card Interrupt

This section discusses the sequence for including an SDIO card interrupt.

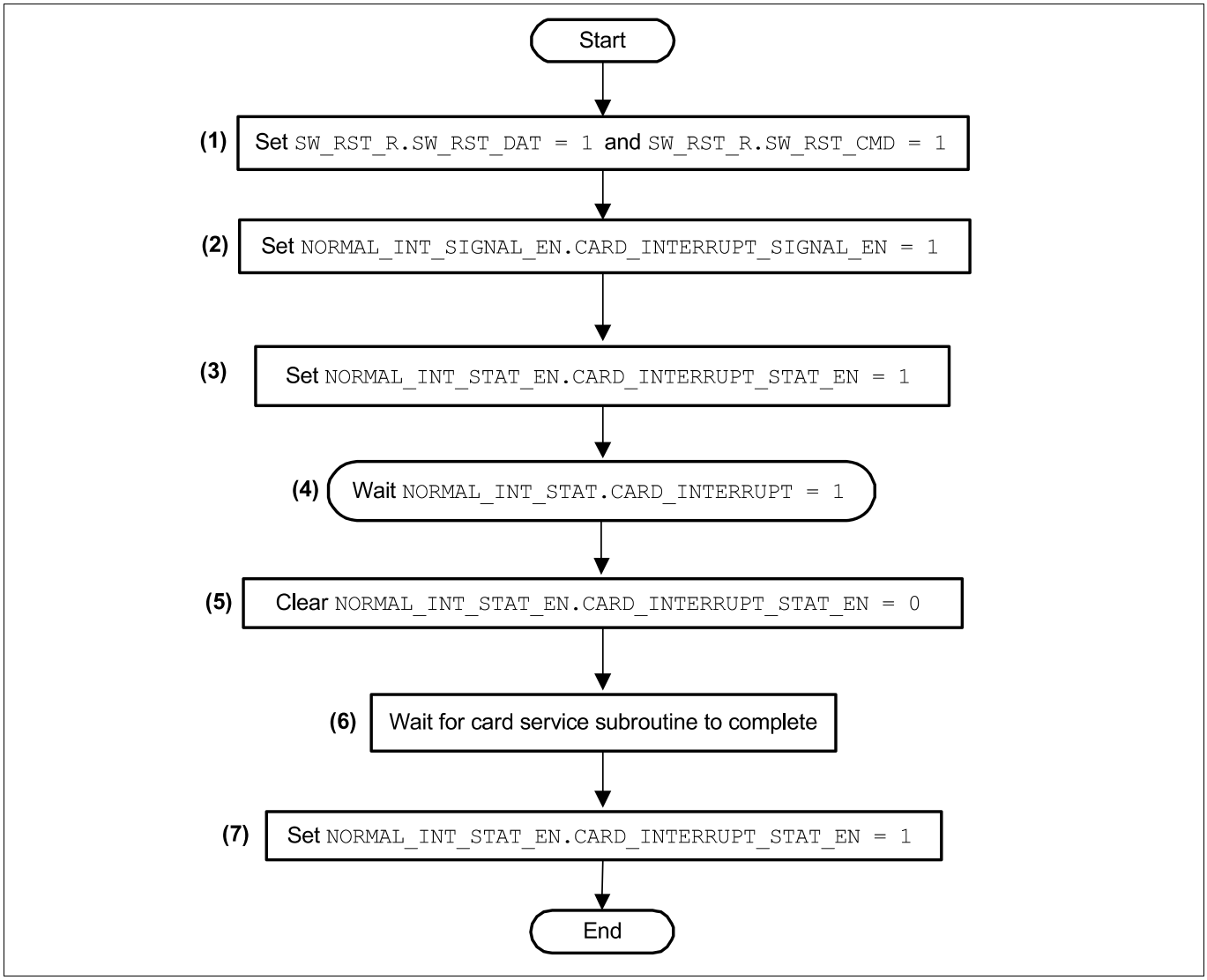


Figure 772 SDIO Card Interrupt Sequence

46.2.2.10 eMMC Transaction Mode

This section discusses the programming sequence for setting the host controller for an eMMC interface.

This section discusses the following programming sequences:

- **Initializing and Identifying an eMMC Device** on page **Page 34**
- **Issue CMD without Data Transfer for an eMMC Device** on page **Page 35**
- **Issue CMD with Data Transfer for an eMMC Device** on page **Page 35**
- **Switch to Various Speed Modes in an eMMC Device** on page **Page 35**
- **Changing the Data Bus Width for an eMMC Device** on page **Page 36**

46.2.2.10.1 Initializing and Identifying an eMMC Device

After powering up the eMMC device, it must be initialized so that the host can start sending commands to the device to perform the data transfer.

Figure 773 shows the programming sequence to initialize and identify an eMMC device.

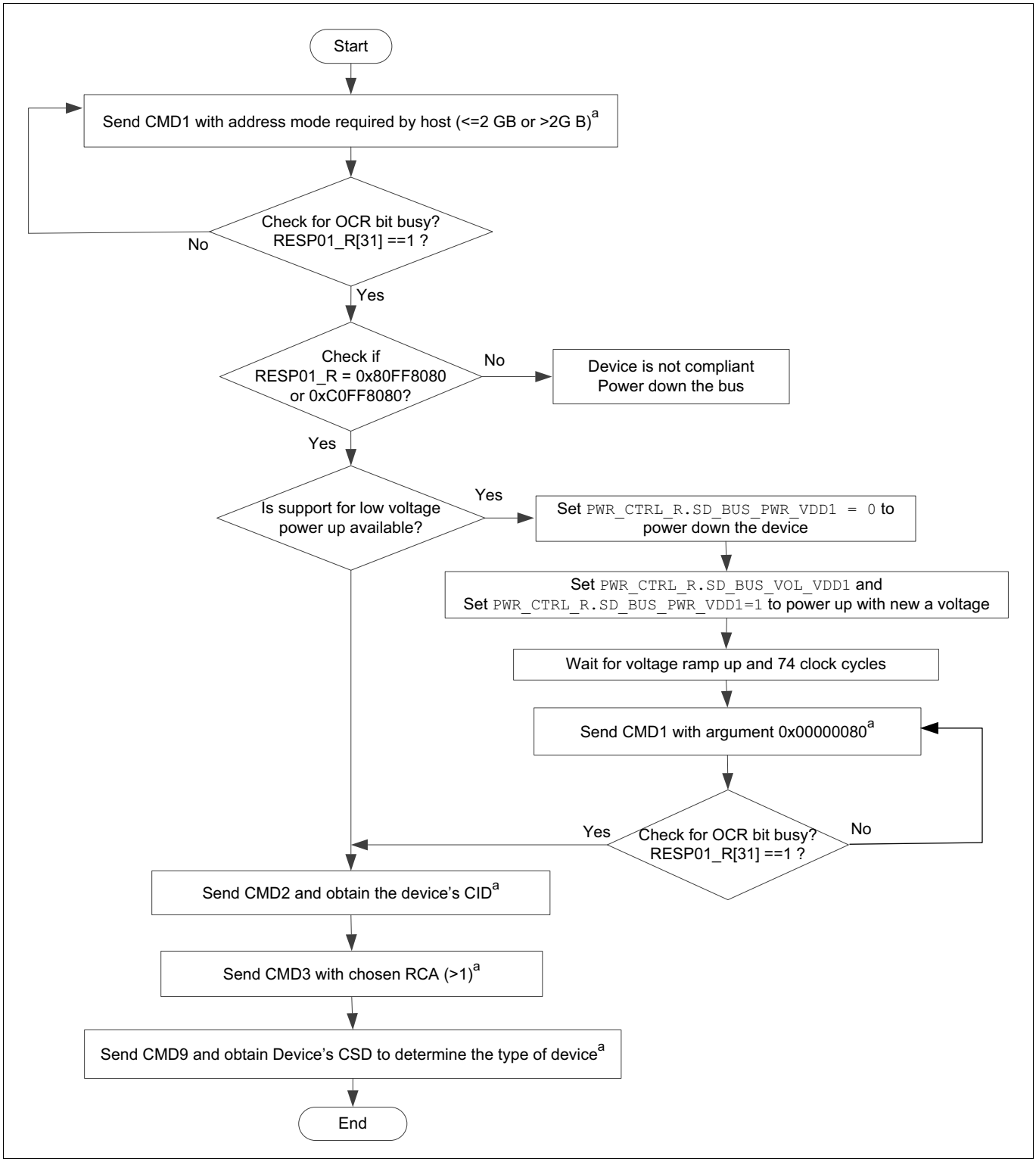


Figure 773 Card Initialization and Identification Programming Sequence

[a] Refer programming sequence for "**Issuing CMD Without Data Transfer**" on **Page 25**.

46.2.2.10.2 Issue CMD without Data Transfer for an eMMC Device

An eMMC device may or may not use the data line (DAT) for data transfer.

The programming sequence for issuing a command without data transfer for an eMMC device is similar to the sequence mentioned in **Issuing CMD Without Data Transfer** on **Page 25**.

46.2.2.10.3 Issue CMD with Data Transfer for an eMMC Device

For an eMMC device, data transfer can occur on the DAT line with or without using DMA/PIO method, and while using DMA, data can be transfer using SDMA, ADMA2, or ADMA3 algorithms.

The programming sequence for issuing a command with data transfer for an eMMC device is similar to the programming sequence mentioned in **Issuing CMD with Data Transfer (Not Using DMA/PIO)** on **Page 26**, **Issuing CMD with Data Transfer (Using SDMA)** on **Page 28**, **Issuing CMD with Data Transfer (Using ADMA2)** on **Page 29** Data transfer can be done with or without using DMA. The supported DMA modes are SDMA, ADMA2, and ADMA3.

46.2.2.10.4 Switch to Various Speed Modes in an eMMC Device

The eMMC bus speed mode of the controller must be changed to match the host driver speed mode. During this process, DWC_mshc must sample the CMD line using data strobe. The switch command (CMD6) is used to change the bus speed mode, and this command is effective only during transfer mode.

Figure 774 shows the programming sequence to switch to various speed modes in an eMMC device. In **Figure 774**, note that CMD6 is effective only during the transfer state.

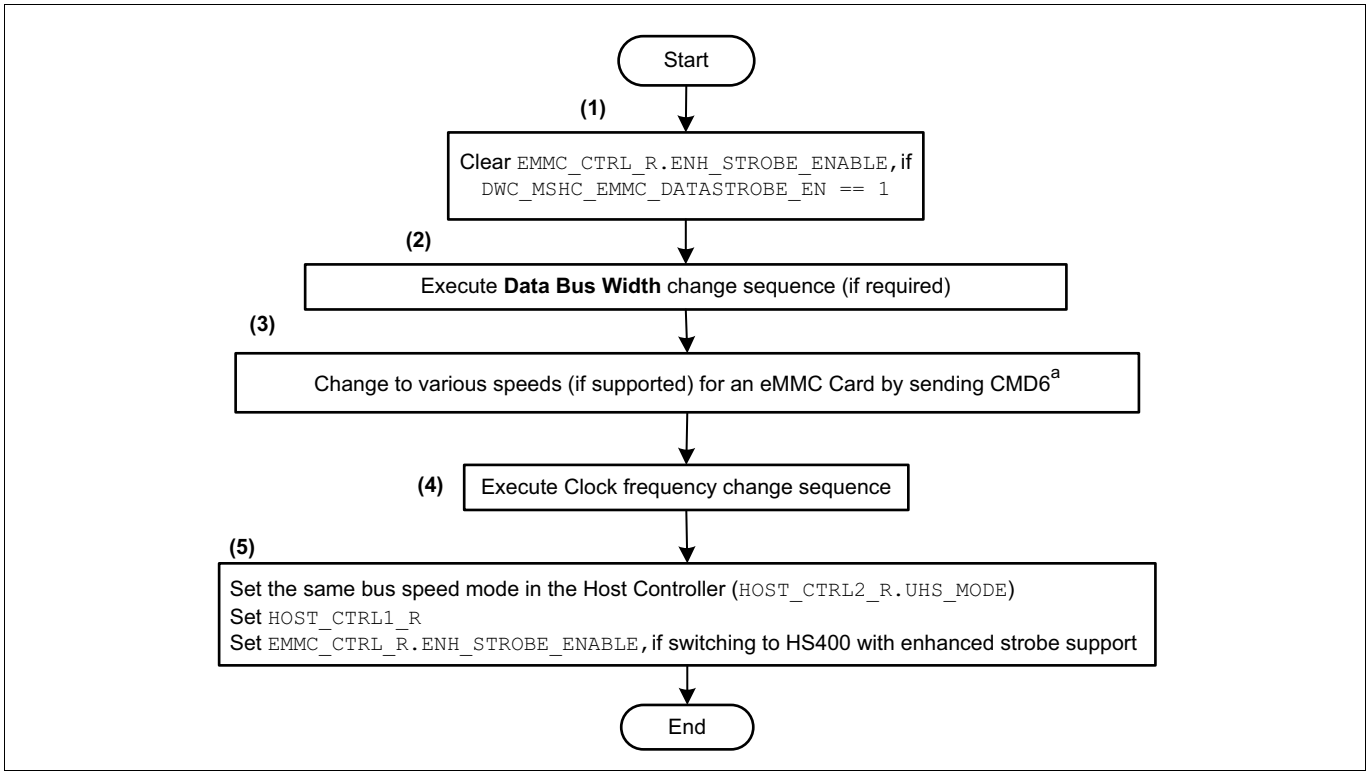


Figure 774 Programming Sequence to Switch to Various Speed Modes in an eMMC Device

[a] Refer programming sequence for "**Issuing CMD Without Data Transfer**" on **Page 25**.

46.2.2.10.5 Changing the Data Bus Width for an eMMC Device

Based on the data bus width of the eMMC device, the host driver must change the data bus width of the Host controller to 1-bit, 4-bit, or 8-bit using CMD6.

Figure 775 shows the programming sequence to change the data bus width for an eMMC device. In **Figure 775**, note that CMD6 is effective only during transfer state.

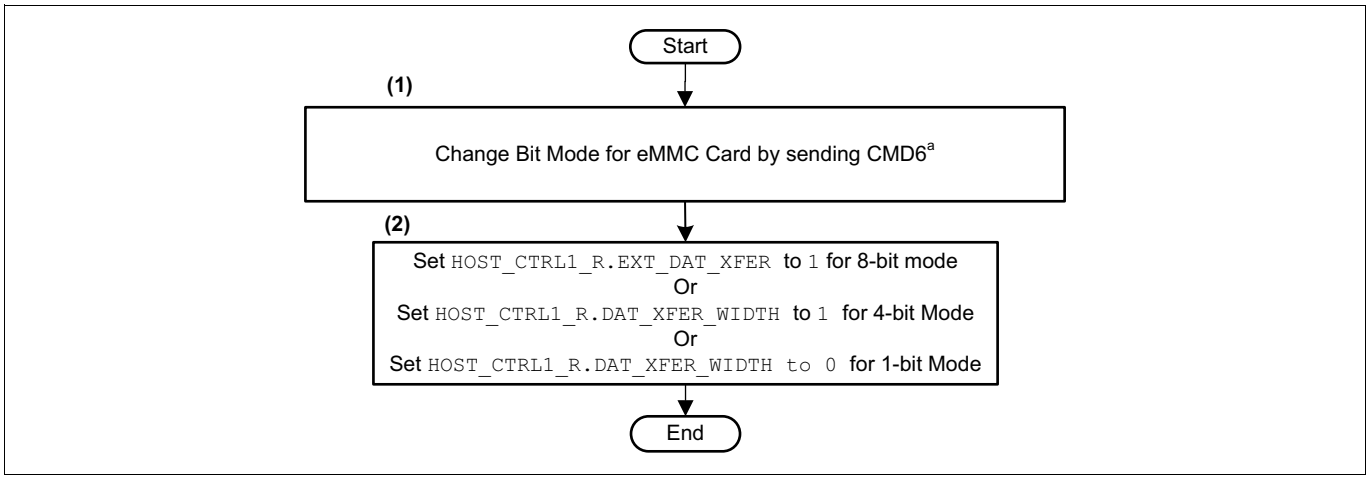


Figure 775 Programming Sequence to Change Data Bus Width for an eMMC Device

46.2.2.11 Boot and Abort Programming Sequences for eMMC

This section discusses the following programming sequences for booting:

- **Preparing for a Boot** on page **Page 37**
- **Initiating a Mandatory Boot in Non-DMA Mode** on page **Page 38**
- **Initiating a Mandatory Boot in SDMA Mode** on page **Page 39**
- **Initiating a Mandatory Boot in ADMA2 Mode** on page **Page 40**
- **Abort Mandatory Boot** on page **Page 41**
- **Initiating Alternate Boot** on page **Page 42**
- **Alternate Boot** on page **Page 43**

46.2.2.11.1 Preparing for a Boot

Before a device can be booted, necessary register bits must be set so that values such as signally level, bus speed mode, bus width, DMA mode are set.

Figure 776 shows the programming sequence to prepare for a boot.

Note: The Host Controller does not support boot in ADMA3 mode.

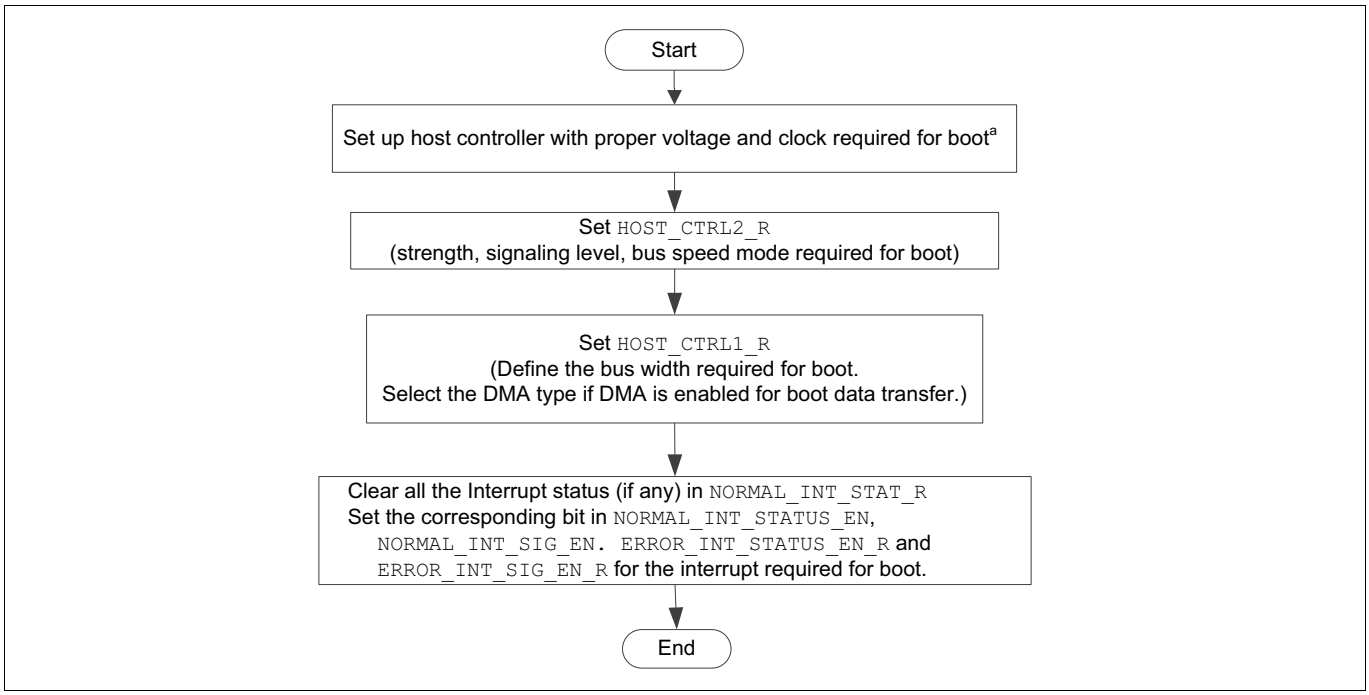


Figure 776 Programming Sequence to Prepare for a Boot

[a] Refer the programming sequence for **Host Controller Clock Setup Sequence**

46.2.2.11.2 Initiating a Mandatory Boot in Non-DMA Mode

While initiating a mandatory boot, the boot is initiated after setting the mandatory boot enable bit in Boot Control Register. In this case, the CMD line is held low for entire boot operation. It is considered that the device is in pre-boot state when boot is initiated. In a non-DMA mode, the process is slower compared to that of a process in a DMA mode.

Figure 777 shows the programming flow to initiate a mandatory boot in non-DMA mode.

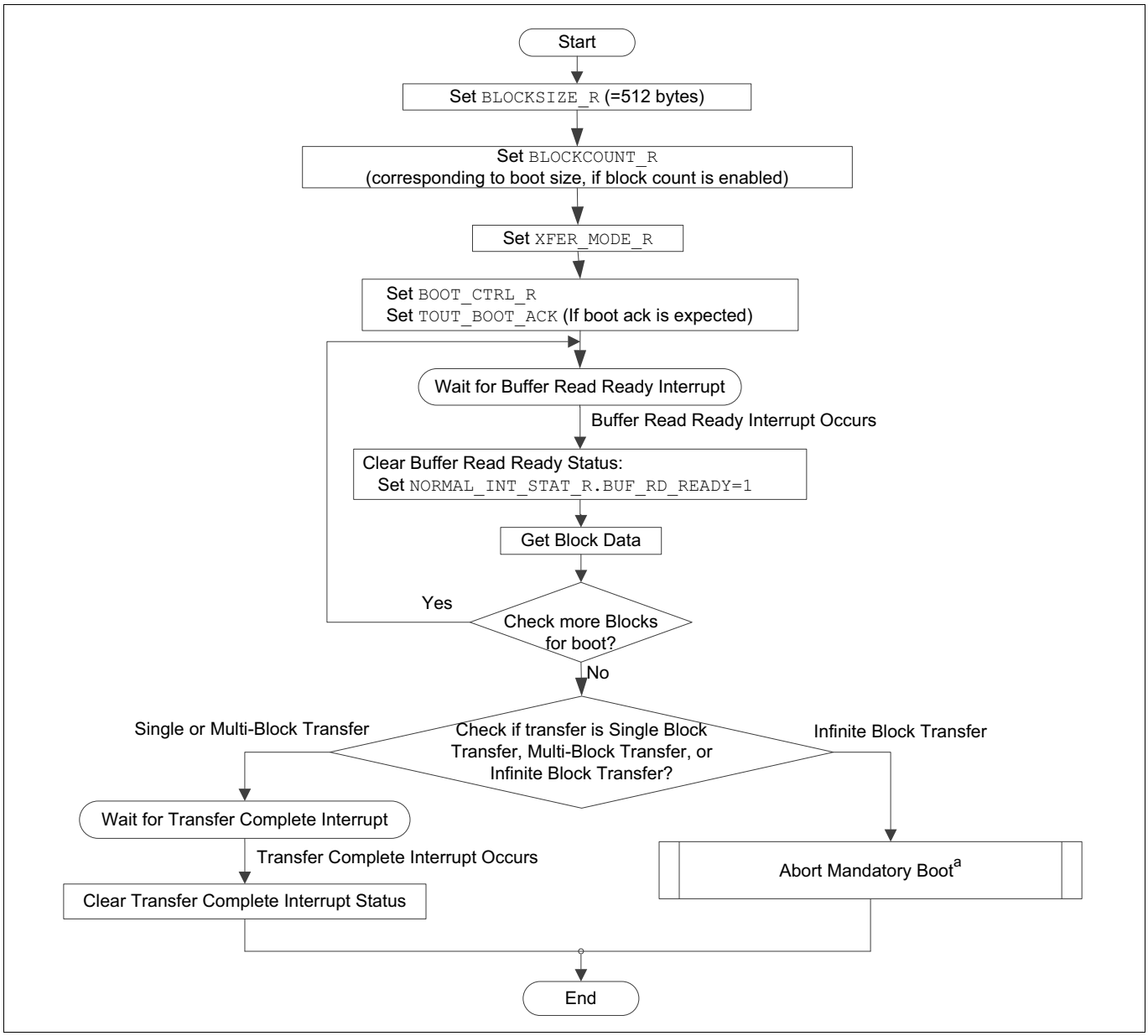


Figure 777 Programming Sequence for Initiating a Mandatory Boot in Non-DMA Mode

[a] Refer programming sequence for "**Abort Mandatory Boot**" on **Page 41**.

46.2.2.11.3 Initiating a Mandatory Boot in SDMA Mode

The procedure to initiate a mandatory boot in SDMA mode involves setting SDMA setting address register and other values related to data transfer.

Figure 778 shows the programming sequence to initiate a mandatory boot in SDMA mode.

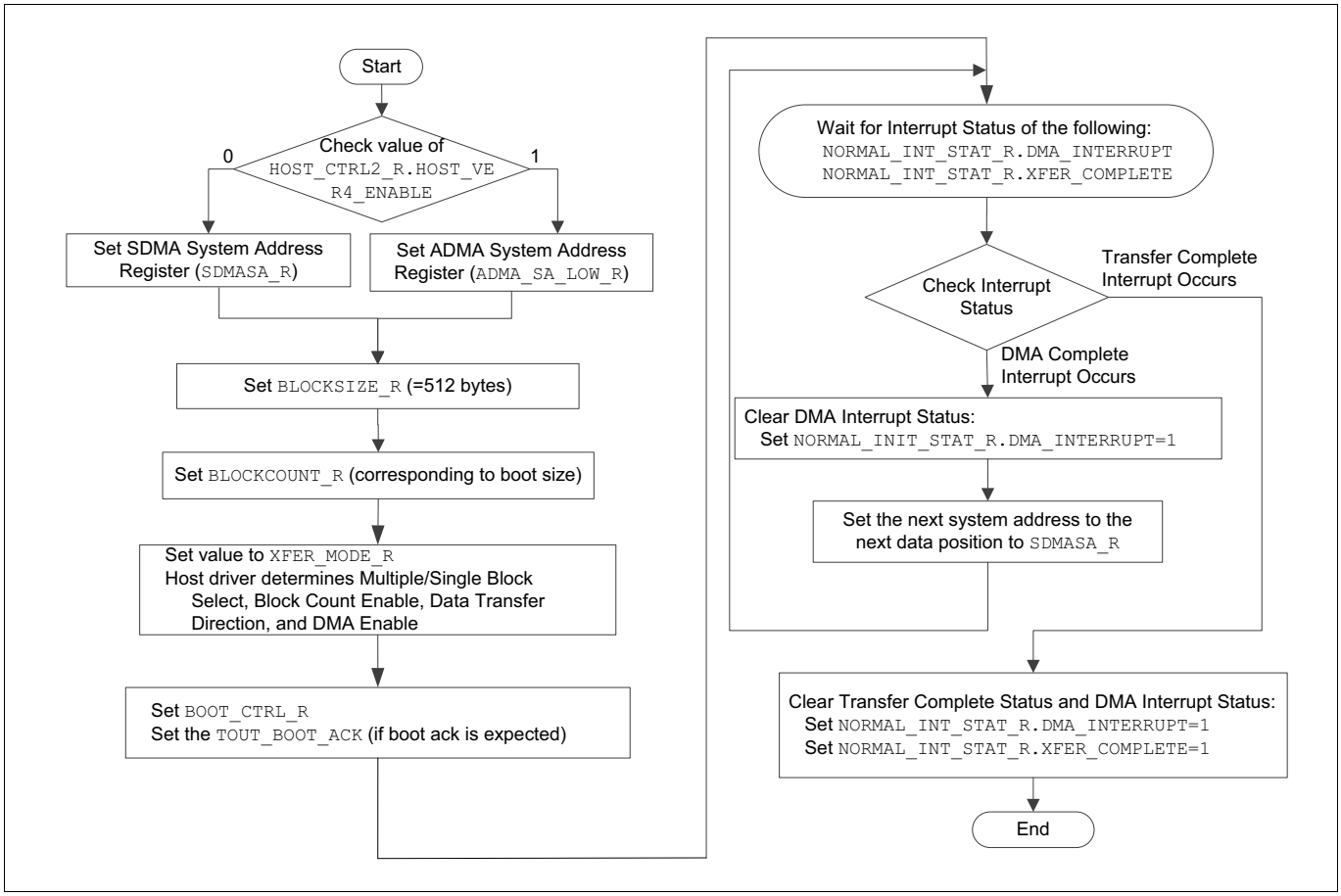


Figure 778 Programming Sequence to Initiate a Mandatory Boot in SDMA Mode

46.2.2.11.4 Initiating a Mandatory Boot in ADMA2 Mode

The procedure to initiate a mandatory boot in ADMA2 mode involves setting ADMA setting address register and other values related to data transfer.

Figure 779 shows the programming sequence for initiating a mandatory boot in ADMA2 mode.

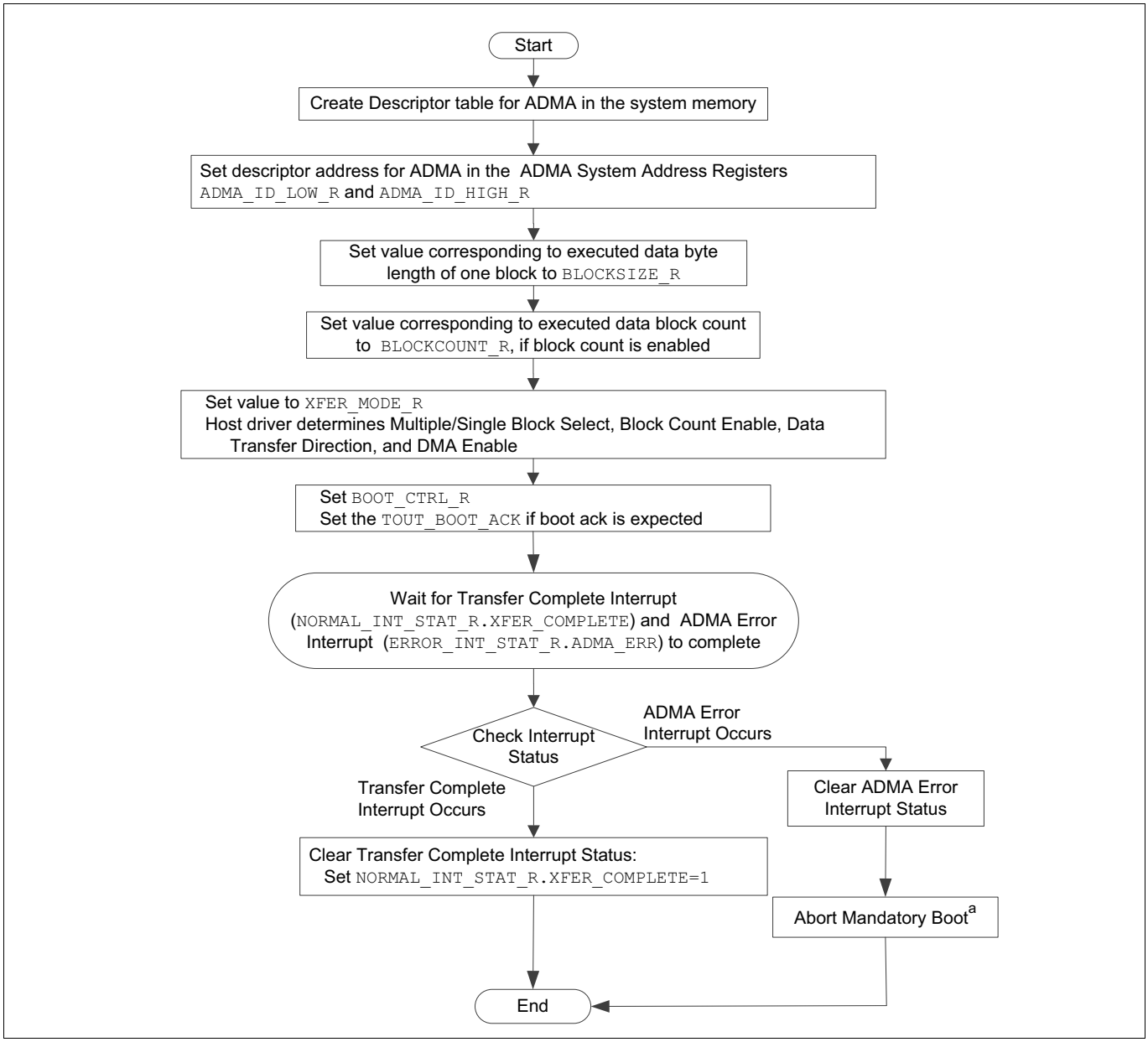


Figure 779 Programming Sequence to Initiate a Mandatory Boot in ADMA2 Mode

[a] Refer programming sequence for "**Abort Mandatory Boot**" on **Page 41**.

46.2.2.11.5 Abort Mandatory Boot

A mandatory boot is executed when an application wants to exit the boot prematurely due to an error or due to receipt of sufficient boot data.

Figure 780 shows the programming sequence for ing a mandatory boot. A mandatory boot is usually exited by pulling up the CMD line to high automatically by DWC_mshc if the boot data is successfully transmitted to the system memory and if the transfer complete interrupt is generated. Note that in **Figure 780**, command inhibit shall be asserted while the CMD line is pulled low and command complete interrupt shall be generated while exiting the mandatory boot.

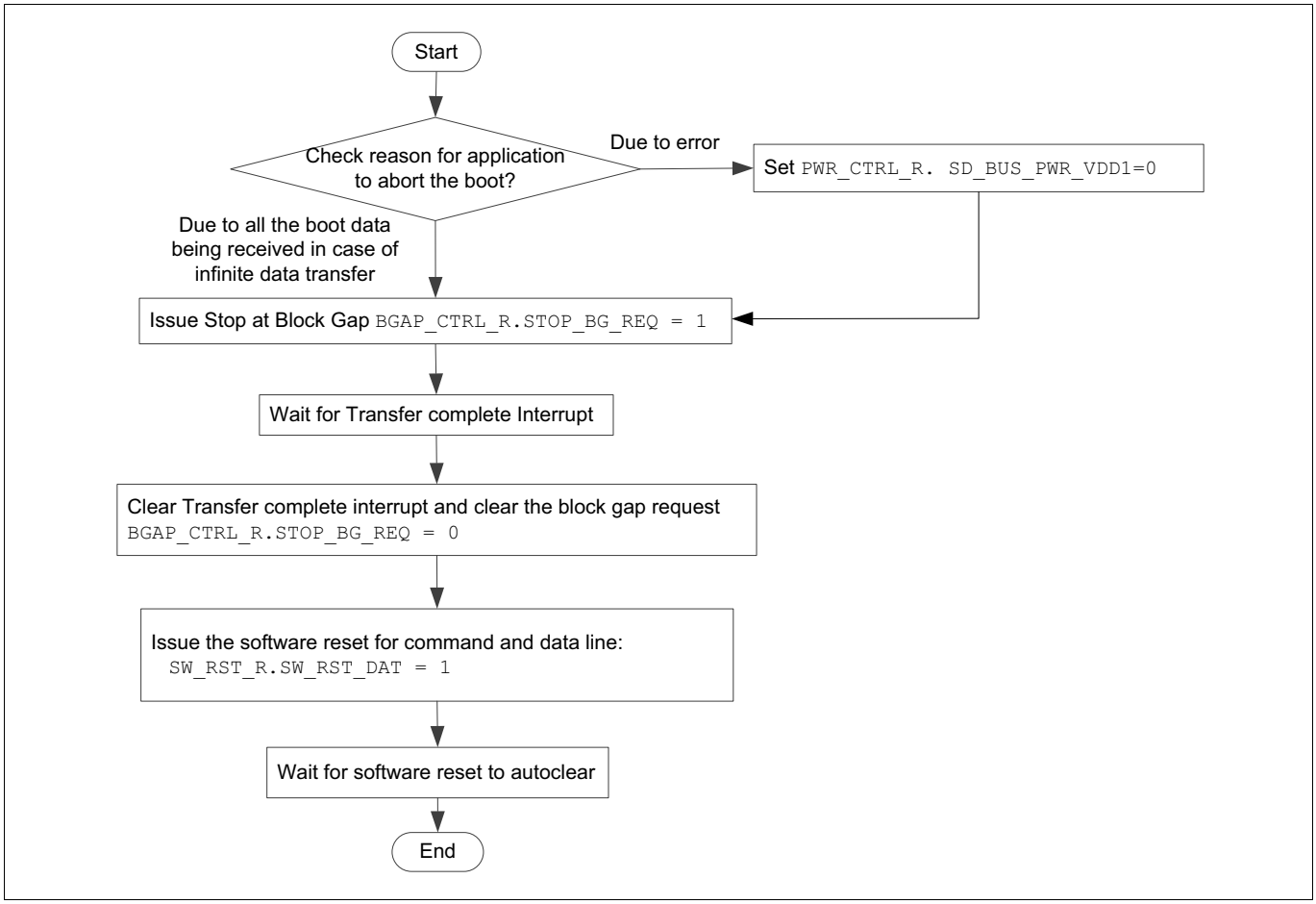


Figure 780 Programming Sequence to a Abort Mandatory Boot

46.2.2.11.6 Initiating Alternate Boot

Alternate Boot is a command-based boot operation and can be initiated by issuing CMD0.

For the programming sequence for initiating an alternate book, refer programming sequences for “Issuing CMD with Data Transfer”, “Issuing CMD with Data Transfer (Using SDMA)”, and “Issuing CMD with Data Transfer (Using ADMA2)” with the following exceptions:

- If boot acknowledgment is expected, BOOT_CTRL_R shall be programmed to set the timeout counter for boot acknowledgment before setting the CMD_R register.
- Response check cannot be performed as CMD0 has no response.
- ing the alternate boot operation follows the “ Alternate Boot” programming sequence instead of CMD12 for other data transfer operation.

When the boot is initiated, device state is assumed to be in pre-boot state and 74 clock cycles has expired after power is stable before issuing CMD0.

46.2.2.11.7 Alternate Boot

An alternate boot is ed if an application wants to exit the boot prematurely due to error or sufficient boot data has been received. However, an alternate boot can only be exited by the application by issuing CMD0/reset. The application exits the boot after transfer compete interrupt is received.

Figure 781 shows the programming sequence for ing an alternate boot.

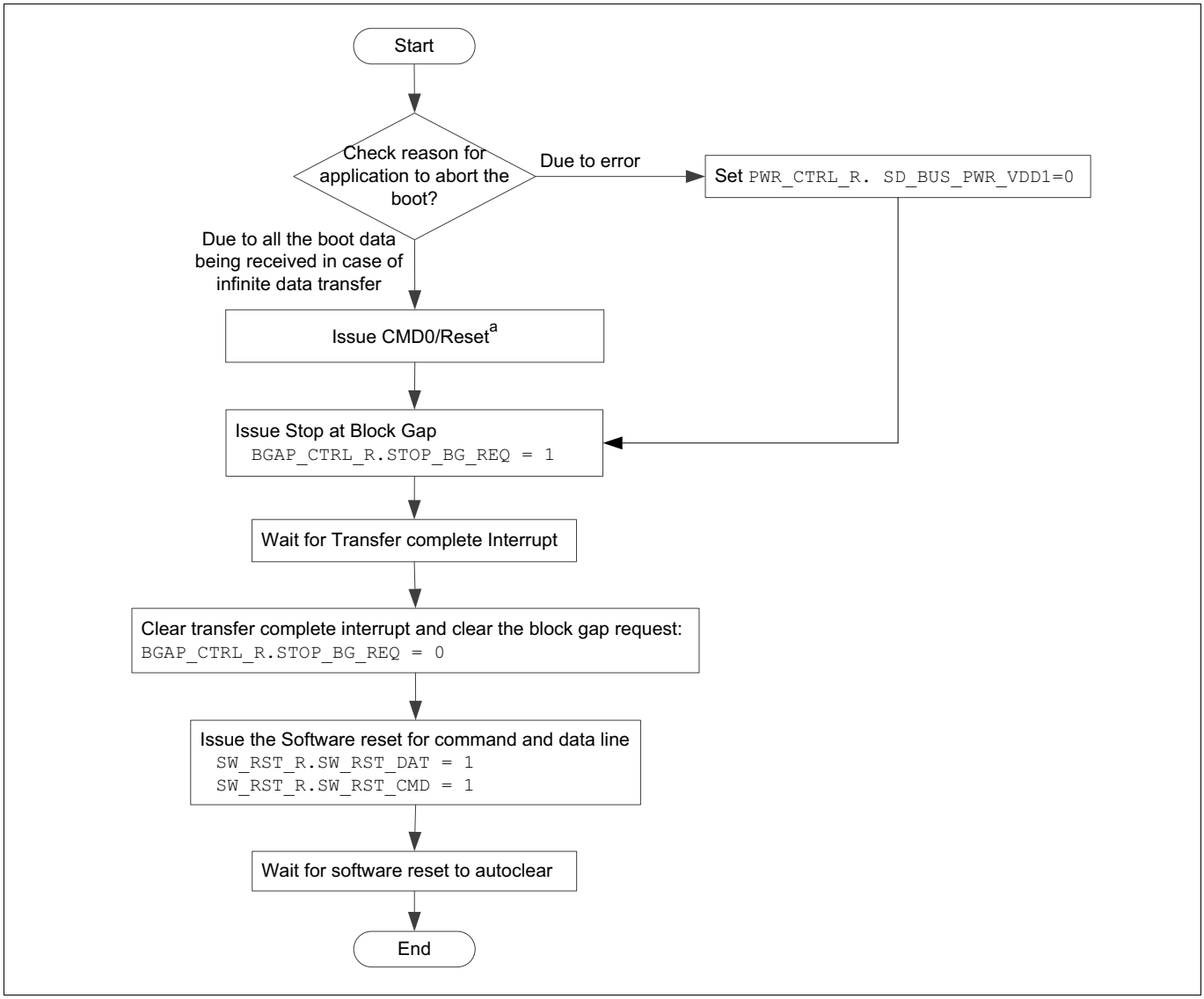


Figure 781 Programming Sequence for ing an Alternate Boot

[a] Refer programming sequence for "**Initializing and Identifying an eMMC Device**" on **Page 34**.

46.2.2.12 Error Recovery in SD/eMMC Mode

An error recovery procedure is required when an error interrupt is indicated by the Error Interrupt Status register and when errors exist in Auto CMD12.

Figure 782 and **Figure 783** show the error recovery sequence for SD/eMMC mode.

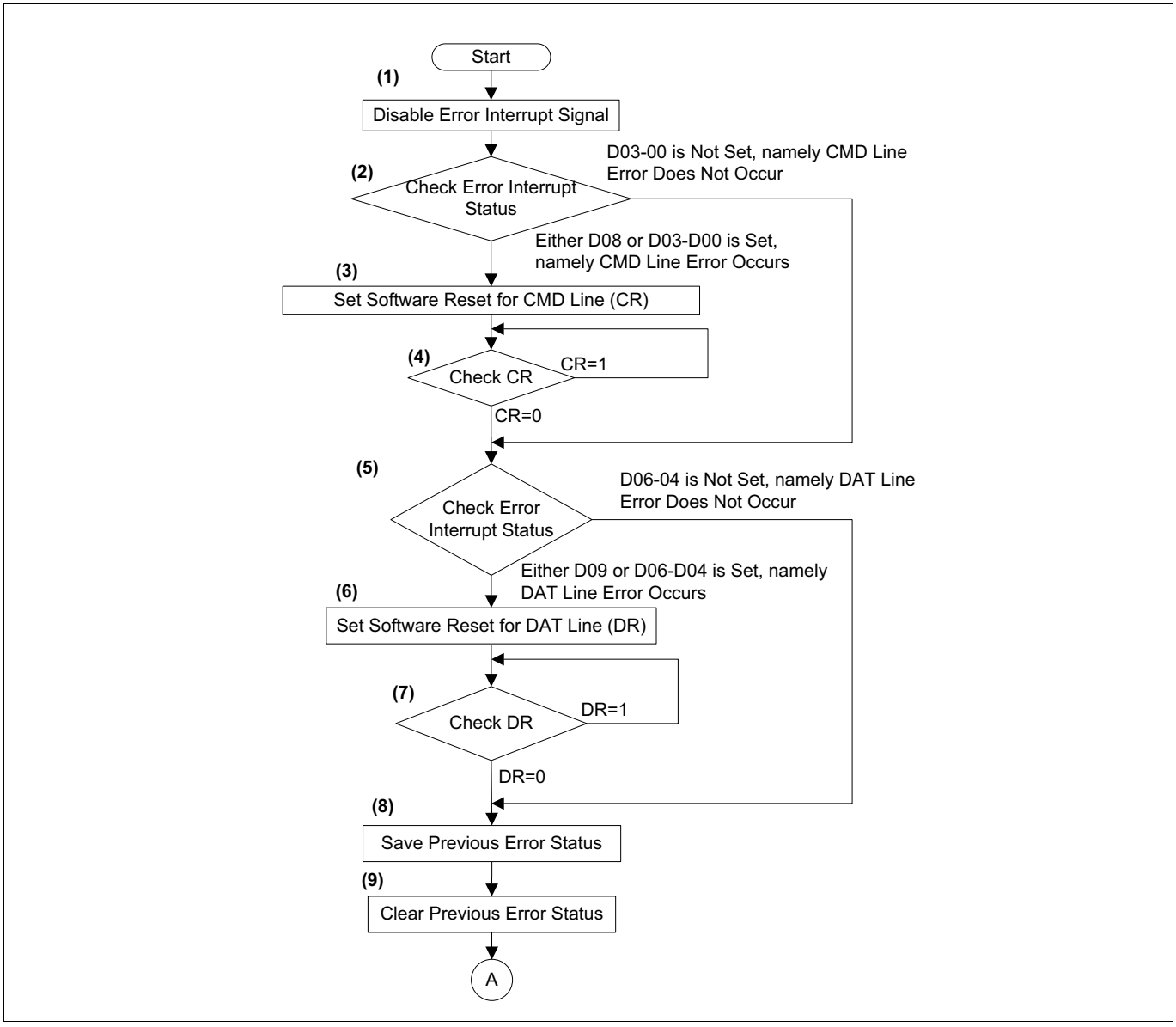


Figure 782 Programming Sequence for Error Recovery in SD/eMMC Mode Part 1

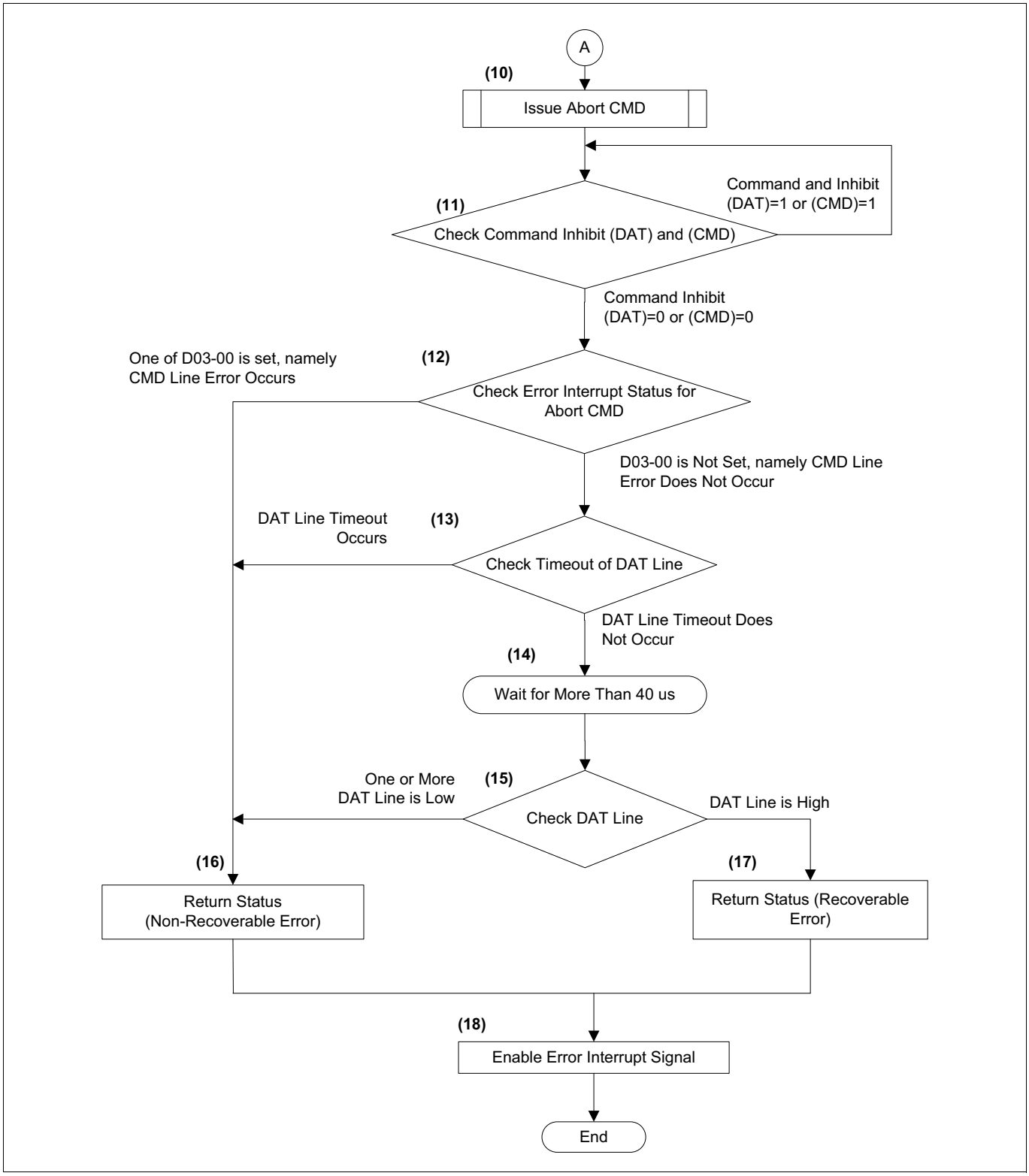


Figure 783 Programming Sequence for Error Recovery in SD/eMMC Mode Part 2

46.3 Registers

Accessing non-listed addresses will result in an bus error.

Table 540 Register Overview - SDMMC (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
SDMASA	SDMA System Address register	000_H	SV,U	U,SV,P	Application Reset	48
BLOCKSIZE	Block Size register	004_H	SV,U	U,SV,P	Application Reset	50
BLOCKCOUNT	16-bit Block Count register	006_H	SV,U	U,SV,P	Application Reset	51
ARGUMENT	Argument register	008_H	SV,U	U,SV,P	Application Reset	51
XFER_MODE	Transfer Mode register	$00C_H$	SV,U	U,SV,P	Application Reset	52
CMD	Command register	$00E_H$	SV,U	U,SV,P	Application Reset	55
RESP01	Response Register 01	010_H	SV,U	BE	Application Reset	57
RESP23	Response Register 23	014_H	SV,U	BE	Application Reset	57
RESP45	Response Register 45	018_H	SV,U	BE	Application Reset	58
RESP67	Response Register 67	$01C_H$	SV,U	BE	Application Reset	58
BUF_DATA	Buffer Data Port Register	020_H	SV,U	U,SV,P	Application Reset	59
PSTATE_REG	Present State Register	024_H	SV,U	BE	Application Reset	59
HOST_CTRL1	Host Control 1 Register	028_H	SV,U	U,SV,P	Application Reset	63
PWR_CTRL	Power Control Register	029_H	SV,U	U,SV,P	Application Reset	64
BGAP_CTRL	Block Gap Control Register	$02A_H$	SV,U	U,SV,P	Application Reset	65
WUP_CTRL	Wakeup Control Register	$02B_H$	SV,U	U,SV,P	Application Reset	66
CLK_CTRL	Clock Control Register	$02C_H$	SV,U	U,SV,P	Application Reset	67
TOUT_CTRL	Timeout Control Register	$02E_H$	SV,U	U,SV,P	Application Reset	69
SW_RST	Software Reset Register	$02F_H$	SV,U	U,SV,P	Application Reset	70

Table 540 Register Overview - SDMMC (ascending Offset Address) (cont'd)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
NORMAL_INT_STAT	Normal Interrupt Status Register	030_H	SV,U	U,SV,P	Application Reset	72
ERROR_INT_STAT	Error Interrupt Status Register	032_H	SV,U	U,SV,P	Application Reset	75
NORMAL_INT_STAT_EN	Normal Interrupt Status Enable Register	034_H	SV,U	U,SV,P	Application Reset	79
ERROR_INT_STAT_EN	Error Interrupt Status Enable Register	036_H	SV,U	U,SV,P	Application Reset	81
NORMAL_INT_SIGNAL_EN	Normal Interrupt Signal Enable Register	038_H	SV,U	U,SV,P	Application Reset	83
ERROR_INT_SIGNAL_EN	Error Interrupt Signal Enable Register	$03A_H$	SV,U	U,SV,P	Application Reset	85
AUTO_CMD_STAT	Auto CMD Status Register	$03C_H$	SV,U	U,SV,P	Application Reset	87
HOST_CTRL2	Host Control 2 Register	$03E_H$	SV,U	U,SV,P	Application Reset	88
CAPABILITIES1	Capabilities 1 Register 0 to 31	040_H	SV,U	BE	Application Reset	90
CAPABILITIES2	Capabilities Register 32 to 63	044_H	SV,U	BE	Application Reset	94
CURR_CAPABILITIES1	Maximum Current Capabilities Register 0 to 31	048_H	SV,U	BE	Application Reset	97
CURR_CAPABILITIES2	Maximum Current Capabilities Register 32 to 63	$04C_H$	SV,U	BE	Application Reset	98
FORCE_AUTO_CMD_STAT	Force Event Register for Auto CMD Error Status register	050_H	SV,U	U,SV,P	Application Reset	98
FORCE_ERROR_INT_STAT	Force Event Register for Error Interrupt Status	052_H	SV,U	U,SV,P	Application Reset	100
ADMA_ERR_STAT	ADMA Error Status Register	054_H	SV,U	BE	Application Reset	101
ADMA_SA_LOW	ADMA System Address Register Low	058_H	SV,U	U,SV,P	Application Reset	102
PRESET_INIT	Preset Value for Initialization	060_H	SV,U	BE	Application Reset	103
PRESET_DS	Preset Value for Default Speed	062_H	SV,U	BE	Application Reset	104
PRESET_HS	Preset Value for High Speed	064_H	SV,U	BE	Application Reset	105

Table 540 Register Overview - SDMMC (ascending Offset Address) (cont'd)

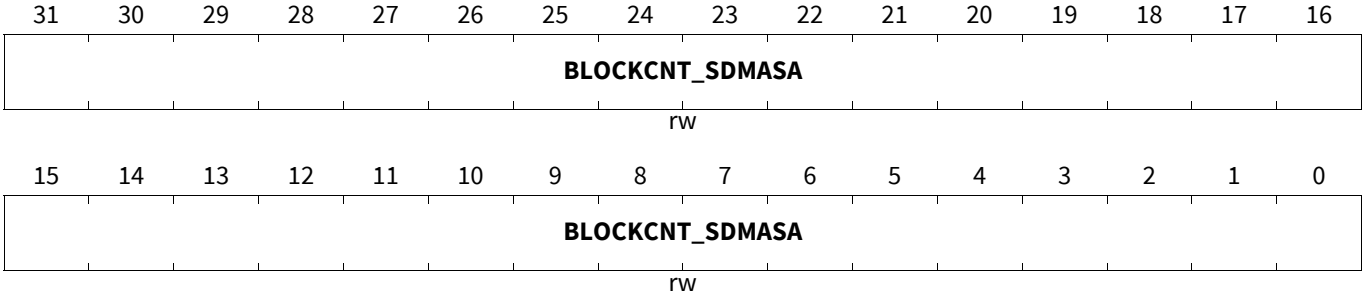
Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
ADMA_ID_LOW	ADMA3 Integrated Descriptor Address Register - Low	078_H	SV,U	U,SV,P	Application Reset	106
P_VENDOR_SPECIFIC_AREA	Pointer for Vendor Specific Area 1	$0E8_H$	SV,U	BE	Application Reset	106
P_VENDOR2_SPECIFIC_AREA	Pointer for Vendor Specific Area 2	$0EA_H$	SV,U	BE	Application Reset	106
SLOT_INTR_STATUS	Slot Interrupt Status Register	$0FC_H$	SV,U	BE	Application Reset	107
HOST_CNTRL_VERS	Host Controller Version	$0FE_H$	SV,U	BE	Application Reset	108
MSHC_VER_ID	MSHC version	180_H	SV,U	BE	Application Reset	108
MSHC_VER_TYPE	MSHC version type	184_H	SV,U	BE	Application Reset	109
MBIU_CTRL	DMA burst control register	190_H	SV,U	U,SV,P	Application Reset	109
EMMC_CTRL	eMMC Control register	$1AC_H$	SV,U	U,SV,P	Application Reset	110
BOOT_CTRL	eMMC Boot Control register	$1AE_H$	SV,U	U,SV,P	Application Reset	111
EMBEDDED_CTRL	Embedded Control register	280_H	SV,U	U,SV,P	Application Reset	112
CLC	Clock Control Register	300_H	SV,U	SV,E,P	Application Reset	114
ID	Module Identification Register	304_H	SV,U	BE	Application Reset	114
ACCEN0	Access Enable Register 0	$30C_H$	SV,U	SV,SE	Application Reset	115
ACCEN1	Access Enable Register 1	310_H	SV,U	SV,SE	Application Reset	115
KRST0	Kernel Reset Register 0	314_H	SV,U	SV,E,P	Application Reset	116
KRST1	Kernel Reset Register 1	318_H	SV,U	SV,E,P	Application Reset	117
KRSTCLR	Kernel Reset Status Clear Register	$31C_H$	SV,U	SV,E,P	Application Reset	118

SDMA System Address register

This register is used to configure 32-bit Block Count or SDMA System Address based on the Host Version 4 Enable bit in the Host Control 2 register. This register is applicable for both SD and eMMC modes.

SDMASA

SDMA System Address register **(000_H)** **Application Reset Value: 0000 0000_H**

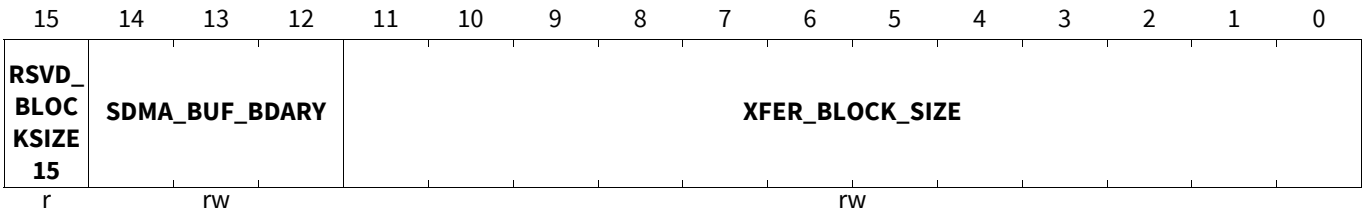


Field	Bits	Type	Description
BLOCKCNT_SDMASA	31:0	rw	**32-bit Block Count (SDMA System Address)** • SDMA System Address (Host Version 4 Enable = 0) - This register contains the system memory address for SDMA transfer in the 32-bit addressing mode. When the Host Controller stops an SDMA transfer, this register shall point to the system address of the next contiguous data position. It can be accessed only if no transaction is executing. Reading this register during data transfers may return invalid value. • 32-bit Block Count (Host Version 4 Enable = 1) - From Host Controller Version 4.10, this register is redefined as 32-bit Block Count. The Host Controller decrements the block count of this register for every block transfer and the data transfer stops when the count reaches zero. This register should be accessed when no transaction is executing. Reading this register during data transfers may return invalid value. • FFFF_FFFFh - 4G - 1 block • ... - ... • 0000_0002h - 2 blocks • 0000_0001h - 1 block • 0000_0000h - Stop Count *Note: For Host Version 4 Enable = 0, Host driver shall not program system address in this register while operating in ADMA mode. It should be programmed in ADMA System Address register.* *Note: For Host Version 4 Enable = 0, Host driver shall program 32-bit block count in this register when Auto CMD23 is enabled for non-DMA and ADMA mode.* **Volatile:** true

Block Size register

This register is used to configure SDMA buffer boundary and number of bytes in a data block. This register is applicable for both SD and eMMC modes.

BLOCKSIZE
Block Size register **(004_H)** **Application Reset Value: 0000_H**



Field	Bits	Type	Description
XFER_BLOCK_SIZE	11:0	rw	**Transfer Block Size** This register specifies the block size of data transfers. In case of memory, it shall be set up to 512 bytes. It can be accessed only if no transaction is executing. Read operations during transfers may return an invalid value, and write operations shall be ignored. 000_H No Data Transfer 001_H 1 Byte 002_H 2 Bytes … 200_H 512 Bytes 201_H reserved … FFF_H reserved
SDMA_BUF_BDARY	14:12	rw	**SDMA Buffer Boundary** These bits specify the size of contiguous buffer in system memory. The SDMA transfer shall wait at every boundary specified by these fields and the Host Controller generates the DMA interrupt to request the Host Driver to update the SDMA System Address register. 000_B 4K Bytes 001_B 8K Bytes 010_B 16K Bytes 011_B 32K Bytes 100_B 64K Bytes 101_B 128K Bytes 110_B 256K Bytes 111_B 512K Bytes
RSVD_BLOCKSIZE15	15	r	**Reserved** This bit is reserved. It will always return 0.

16-bit Block Count register

This register is used to configure the number of data blocks. This register is applicable for both SD and eMMC modes.

BLOCKCOUNT
16-bit Block Count register **(006_H)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLOCK_CNT															
rw															

Field	Bits	Type	Description
BLOCK_CNT	15:0	rw	**16-bit Block Count** • If Host Version 4 Enable is set 0 or 16-bit Block Count register is set to non-zero, 16-bit Block Count register is selected. • If Host Version 4 Enable is set 1 and 16-bit Block Count register is set to zero, 32-bit Block Count register is selected. *Note: For Host Version 4 Enable = 0, this register should be set to 0000h before programming 32-bit block count register when Auto CMD23 is enabled for non-DMA and ADMA mode.* **Volatile:** true 0000_HStop Count 0001_H1 Block 0002_H2 Blocks … $FFFF_H$65535 Blocks

Argument register

This register is used to configure the SD/eMMC command argument.

ARGUMENT
Argument register **(008_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARGUMENT															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARGUMENT															
rw															

Field	Bits	Type	Description
ARGUMENT	31:0	rw	**Command Argument** The SD/eMMC command argument is specified as bit 39-8 of the command format.

Transfer Mode register

This register is used to control the operation of data transfers for an SD/eMMC mode. The Host driver shall set this register before issuing a command that transfers data.

XFER_MODE

Transfer Mode register **($00C_H$)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD							RESP_INT_DISABLE	RESP_ERR_CHK_ENABLE	RESP_TYPE	MULTI_BLK_SEL	DATA_XFER_DIR	AUTO_CMD_ENABLE		BLOCK_COUNT_ENABLE	DMA_ENABLE
r							rw	rw	rw	rw	rw	rw		rw	rw

Field	Bits	Type	Description
DMA_ENABLE	0	rw	**DMA Enable** This bit enables the DMA functionality. If this bit is set to 1, a DMA operation begins when Host Driver writes to Command register. You can select one of the DMA modes by using DMA Select in the Host Control 1 register. The DMA supports only transfers of 32bit data. The addresses have to be 32bit aligned for 32bit addressing. 0_B No data transfer or Non DMA data transfer 1_B DMA Data transfer
BLOCK_COUNT_ENABLE	1	rw	**Block Count Enable** This bit is used to enable the Block Count register, which is relevant for multiple block transfers. When this bit is 0, the Block Count register is disabled, which is useful in executing an infinite transfer. Host Driver should set this bit to 0 when ADMA is used. 0_B Disable 1_B Enable

Field	Bits	Type	Description
AUTO_CMD_ENABLE	3:2	rw	**Auto Command Enable** This field determines use of auto command functions. *Note: In SDIO, this field should be set as 00b (Auto Command Disabled).* 00_B Auto Command Disabled 01_B Auto CMD12 Enable 10_B Auto CMD23 Enable 11_B Auto CMD Auto Select
DATA_XFER_DIR	4	rw	**Data Transfer Direction Select** This bit defines the direction of DAT line data transfers. This bit is set to 1 by the Host Driver to transfer data from the SD/eMMC card to the Host Controller and it is set to 0 for all other commands. 0_B Write (Host to Card) 1_B Read (Card to Host)
MULTI_BLK_SEL	5	rw	**Multi/Single Block Select** This bit is set when issuing multiple-block transfer commands using DAT line. If this bit is set to 0, it is not necessary to set Block Count register. 0_B Write (Single Block) 1_B Read (Multiple Block)

Field	Bits	Type	Description
RESP_TYPE	6	rw	**Response Type R1/R5** When response error check is enabled, this bit selects either R1 or R5 response types. Error statuses checked in R1: • OUT_OF_RANGE • ADDRESS_ERROR • BLOCK_LEN_ERROR • WP_VIOLATION • CARD_IS_LOCKED • COM_CRC_ERROR • CARD_ECC_FAILED • CC_ERROR • ERROR Response Flags checked in R5: • COM_CRC_ERROR • ERROR • FUNCTION_NUMBER • OUT_OF_RANGE 0_B R1 (Memory) 1_B R5 (SDIO)
RESP_ERR_CHK_ENABLE	7	rw	**Response Error Check Enable** Host Controller supports response check function to avoid overhead of response error check by Host driver. Only R1 and R5 can be checked by the controller. If Host Controller checks the response error, set this bit to 1 and set Response Interrupt Disable to 1. If an error is detected, Response Error interrupt is generated in the Error Interrupt Status register. *Note: Response error check should not be enabled for any reponse type other than R1 and R5.* *Note: Response check should not be enabled for tuning command.* 0_B Response Error Check is disabled 1_B Response Error Check is enabled

Field	Bits	Type	Description
RESP_INT_DISABLE	8	rw	**Response Interrupt Disable** Host Controller supports response check function to avoid overhead of response error check by Host driver. Only R1 and R5 can be checked by the controller. If Host Driver checks response error, set this bit to 0 and wait for Command Complete Interrupt and then check the response register. If Host Controller checks response error, set this bit to 1 and set Response Error Check Enable to 1. Command Complete Interrupt is disabled by this bit regardless of Command Complete Signal Enable. *Note: While carrying out tuning (When Execute Tuning bit in Host Control2 register is set), command complete interrupt is not generated irrespective of Response Interrupt Disable setting.* 0_B Response Interrupt enabled 1_B Response Interrupt is disabled
RSVD	15:9	r	**Reserved** These bits are reserved. They will always return 0.

Command register

This register is used to provide the information related to command and response packet. This register is applicable for an SD/eMMC mode.

CMD
Command register **($00E_H$)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD		CMD_INDEX						CMD_TYPE		DATA_PRESENT_SEL	CMD_IDX_CHK_ENABLE	CMD_CRC_CHK_ENABLE	SUB_CMD_FLAG	RESP_TYPE_SELECT	
r		rw						rw		rw	rw	rw	rw	rw	

Field	Bits	Type	Description
RESP_TYPE_SELECT	1:0	rw	**Response Type Select** These bits indicate the type of response expected from the card. 00_B No Response 01_B Response Length 136 10_B Response Length 48 11_B Response Length 48 (check Busy after response)
SUB_CMD_FLAG	2	rw	**Sub Command Flag** This bit distinguishes a main command or a sub command. 0_B Main Command 1_B Sub Command

Field	Bits	Type	Description
CMD_CRC_CHK_ENABLE	3	rw	**Command CRC Check Enable** If this bit is set, Host Controller shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC error. *Note: CRC Check enable should be set to 0 for the command with no response, R3 response and R4 response.* 0_B Disable 1_B Enable
CMD_IDX_CHK_ENABLE	4	rw	**Command Index Check Enable** If this bit is set, Host Controller shall check the index field in the response to verify if it has the same value as the command index. If it is not, it is reported as a Command Index error. *Note: Index Check enable should be set to 0 for the command with no response, R2 response, R3 response and R4 response.* 0_B Disable 1_B Enable
DATA_PRESENT_SEL	5	rw	**Data Present Select** This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. This bit is set to 0 in the following instances: • Command using CMD line • Command with no data transfer but using busy signal on DAT[0] line • Resume Command 0_B No Data Present 1_B Data Present
CMD_TYPE	7:6	rw	**Command Type** These bits indicate the command type. 00_B Normal Command 01_B Suspend Command 10_B Resume Command 11_B Abort Command
CMD_INDEX	13:8	rw	**Command Index** These bits shall be set to the command number that is specified in bits 45-40 of the Command Format.
RSVD	15:14	r	**Reserved** These bits are reserved. They will always return 0.

Response Register 01

This register is used to store the response from the cards. The response can be a maximum of 128 bits. These 128 bits are segregated into four 32-bit registers: RESP01_R, RESP23_R, RESP45_R and RESP67_R.

RESP01_R stores 39-08 bits of Response Field for an SD/eMMC mode. Writing this register has no effect.

RESP01

Response Register 01 **(010_H)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESP01															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESP01															
r															

Field	Bits	Type	Description
RESP01	31:0	r	**Command Response** These bits reflect bits 39-8 of the Response Field. *Note: For Auto CMD, the 32-bit response (bits 39-8 of the Response Field) is updated in RESP67_R register.* **Volatile:** true

Response Register 23

This register is used to store the response from the cards. The response can be of maximum 128 bits. These 128 bits are segregated into four 32-bit registers RESP01_R, RESP23_R, RESP45_R and RESP67_R.

RESP23_R stores 71-40 bits of Response Field for an SD/eMMC mode. Writing this register has no effect.

RESP23

Response Register 23 **(014_H)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESP23															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESP23															
r															

Field	Bits	Type	Description
RESP23	31:0	r	**Command Response** These bits reflect bits 71-40 of the Response Field. **Volatile:** true

Response Register 45

This register is used to store the response from the cards. The response can be of maximum 128 bits. These 128 bits are segregated into four 32-bit registers RESP01_R, RESP23_R, RESP45_R and RESP67_R.

RESP45_R stores 103-72 bits of Response Field for an SD/eMMC mode. Writing this register has no effect.

RESP45

Response Register 45 **(018_H)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESP45															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESP45															
r															

Field	Bits	Type	Description
RESP45	31:0	r	**Command Response** These bits reflect bits 103-72 of the Response Field. **Volatile:** true

Response Register 67

This register is used to store the response from the cards. The response can be of maximum 128 bits. These 128 bits are segregated into four 32-bit registers RESP01_R, RESP23_R, RESP45_R and RESP67_R. RESP67_R stores 135-104 bits of Response Field for an SD/eMMC mode. Writing this register has no effect.

RESP67

Response Register 67 **($01C_H$)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESP67															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESP67															
r															

Field	Bits	Type	Description
RESP67	31:0	r	**Command Response** These bits reflect bits 135-104 of the Response Field. *Note: For Auto CMD, this register also reflects the 32-bit response (bits 39-8 of the Response Field).* **Volatile:** true

Buffer Data Port Register

This register is used to access the packet buffer. This register is applicable for an SD/eMMC mode.

BUF_DATA
Buffer Data Port Register **(020_H)** **Application Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BUF_DATA															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUF_DATA															
rw															

Field	Bits	Type	Description
BUF_DATA	31:0	rw	**Buffer Data** The Host Controller packet buffer can be accessed through this 32-bit Buffer Data Port register. **Volatile:** true

Present State Register

This register indicates the present status of the Host Controller. This register is applicable for an SD/eMMC mode. Writing this register has no effect.

PSTATE_REG
Present State Register **(024_H)** **Application Reset Value: 0XX4 00X0$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UHS2_IF_DETECT	LANE_SYNC	IN_DORMANT_ST	SUB_CMD_STAT	CMD_ISSUE_ERR	RSVD_26	HOST_REG_VOL	CMD_LINE_LVL	DAT_3_0				WR_PROTECT_SW_LVL	CARD_DETECT_LEVEL	CARD_STABLE	CARD_INSERTED
r	r	r	r	r	r	r	r	r				r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_15_12				BUF_RD_ENABLE	BUF_WR_ENABLE	RD_XFER_ACTIVE	WR_XFER_ACTIVE	DAT_7_4				RE_TUNE_REQ	DAT_LINE_ACTIVE	CMD_INHIBIT_DAT	CMD_INHIBIT
r				r	r	r	r	r				r	r	r	r

Field	Bits	Type	Description
CMD_INHIBIT	0	r	**Command Inhibit (CMD)** In an SD/eMMC mode, if this bit is 0, it indicates that the CMD line is not in use and the Host controller can issue an SD/eMMC command using the CMD line. This bit is set when the command register is written. This bit is cleared when the command response is received. This bit is not cleared by the response of auto CMD12/23 but cleared by the response of read/write command. **Volatile:** true 0_B Host Controller is not ready to issue a command 1_B Host Controller is ready to issue a command
CMD_INHIBIT_DAT	1	r	**Command Inhibit (DAT)(SD/eMMC Mode only)** This status bit is generated if either DAT line active or Read transfer active is set to 1. If this bit is 0, it indicates that Host Controller can issue subsequent SD/eMMC commands. **Volatile:** true 0_B Can issue command which used DAT line 1_B Cannot issue command which used DAT line
DAT_LINE_ACTIVE	2	r	**DAT Line Active (SD/eMMC Mode only)** This bit indicates whether one of the DAT lines on the SD/eMMC bus is in use. In the case of read transactions, this status indicates whether a read transfer is executing on the SD/eMMC bus. In the case of write transactions, this status indicates whether a write transfer is executing on the SD/eMMC bus. For command with busy, this status indicates whether command executing busy is executing on the SD/eMMC bus. **Volatile:** true 0_B DAT Line Inactive 1_B DAT Line Active
RE_TUNE_REQ	3	r	**Re-Tuning Request** This bit can be ignored.
DAT_7_4	7:4	r	**DAT[7:4] Line Signal Level (Embedded only)** This status is used to check the DAT line level to recover from errors and for debugging. **Volatile:** true
WR_XFER_ACTIVE	8	r	**Write Transfer Active (SD/eMMC Mode only)** This status indicates whether a write transfer is active. **Volatile:** true 0_B No valid data 1_B Transferring data

Field	Bits	Type	Description
RD_XFER_ACTIVE	9	r	**Read Transfer Active (SD/eMMC Mode only)** This status indicates whether a read transfer is active. **Volatile:** true 0_B No valid data 1_B Transferring data
BUF_WR_ENABLE	10	r	**Buffer Write Enable** This status is used for non-DMA transfers. This bit is set if space is available for writing data. **Volatile:** true 0_B Write disable 1_B Write enable
BUF_RD_ENABLE	11	r	**Buffer Read Enable** This status is used for non-DMA transfers. This bit is set if valid data exists in Host buffer. **Volatile:** true 0_B Read disable 1_B Read enable
RSVD_15_12	15:12	r	**Reserved** These bits are reserved. They will always return 0. **Volatile:** true
CARD_INSERTED	16	r	**Card Inserted** This bit indicates whether a card has been inserted. The Host Controller shall debounce this signal so that Host Driver will not need to wait for it to stabilize. **Volatile:** true 0_B Reset or Debouncing or No card 1_B Card Inserted
CARD_STABLE	17	r	**Card Stable** If this bit is set, it means the Card Detect Level is stable. No card is detected if this bit is set to 1 and card inserted is 0 **Volatile:** true 0_B Reset or Debouncing 1_B No Card or Inserted
CARD_DETECT_LEVEL	18	r	**Card Detect Pin Level - CARD_DETECT_PIN_LEVEL** This bit reflects the card detect stats. **Volatile:** true 0_B No card present 1_B Card Present

Field	Bits	Type	Description
WR_PROTECT_SW_LVL	19	r	**Write Protect Switch Pin Level** The write Protect Switch is supported only for memory and combo card. **Volatile:** true 0_B Write protected 1_B Write enabled
DAT_3_0	23:20	r	**DAT[3:0] Line Signal Level (SD/eMMC Mode only)** This status is used to check the DAT line level to recover from errors and for debugging. **Volatile:** true
CMD_LINE_LVL	24	r	**Command Line Signal Level (SD/eMMC Mode only)** This status is used to check the CMD line level to recover from errors and for debugging. **Volatile:** true
HOST_REG_VOL	25	r	**Host Regulator Voltage Stable** This bit can be ignored. **Volatile:** true
RSVD_26	26	r	**Reserved** This bit is reserved. It will always return 0. **Volatile:** true
CMD_ISSUE_ERR	27	r	**Command Not Issued by Error** This bit is set if a command cannot be issued after setting the command register due to error except Auto CMD12 error **Volatile:** true 0_B No error for issuing a command 1_B Command cannot be issued
SUB_CMD_STAT	28	r	**Sub Command Status** This status is used to distinguish main command and sub command status. **Volatile:** true 0_B Main Command Status 1_B Sub Command Status
IN_DORMANT_ST	29	r	**In Dormant Status** This bit can be ignored. **Volatile:** true
LANE_SYNC	30	r	**Lane Synchronization** This bit can be ignored. **Volatile:** true
UHS2_IF_DETECT	31	r	**UHS-II IF Detection** This bit can be ignored. **Volatile:** true

Host Control 1 Register

This register is used to control the operation of Host Controller. This register is applicable for an SD/eMMC mode.

HOST_CTRL1
Host Control 1 Register **(028$_H$)** **Application Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
CARD_DETECT_SIG_LVL	CARD_DETECT_TEST_LVL	EXT_DAT_XFER	DMA_SEL		HIGH_SPEED_EN	DAT_XFER_WIDTH	LED_CTRL
rw	rw	rw	rw		rw	rw	rw

Field	Bits	Type	Description
LED_CTRL	0	rw	**LED Control** This bit can be ignored. Must be written with '0'. 0_B LED off 1_B LED on
DAT_XFER_WIDTH	1	rw	**Data Transfer Width (SD/eMMC Mode only)** This bit selects the data width of the Host Controller. The Host Driver shall set it to match the data width of the SD/eMMC card. 0_B 1-bit mode 1_B 4-bit mode
HIGH_SPEED_EN	2	rw	**High Speed Enable (SD/eMMC Mode only)** Before setting this bit, the Host Driver shall check the High Speed Support in the Capabilities register. This bit is used to determine the selection of preset value for High Speed mode. *Note: DWC_MSHC always outputs the **sd_cmd_out and sd_dat_out** lines at the rising edge of **cclk_tx** clock irrespective of this bit.* 0_B Normal Speed mode 1_B High Speed mode

Field	Bits	Type	Description
DMA_SEL	4:3	rw	**DMA Select** This field is used to select the DMA type. When Host Version 4 Enable is 1 in Host Control 2 register: • 00b - SDMA is selected • 01b - Reserved • 10b - ADMA2 is selected • 11b - ADMA2 or ADMA3 is selected When Host Version 4 Enable is 0 in Host Control 2 register: • 00b - SDMA is selected • 01b - Reserved • 10b - 32-bit Address ADMA2 is selected • 11b - Reserved 00_B SDMA is selected 01_B Reserved 10_B ADMA2 is selected (+32 bit address if Host Version 4 enable is 1 in Host Control 2 register) 11_B ADMA2 or ADMA3 is selected
EXT_DAT_XFER	5	rw	**Extended Data Transfer Width (Embedded and an SD/eMMC Mode only)** This bit controls 8-bit bus width mode of embedded device. 0_B Bus Width is selected by Data Transfer Width 1_B 8-bit Bus Width
CARD_DETECT_TEST_LVL	6	rw	**Card Detect Test Level** This bit is enabled while the Card Detect Signal Selection is set to 1 and it indicates card inserted or not 0_B No Card 1_B Card Inserted
CARD_DETECT_SIG_LVL	7	rw	**Card Detect Signal Selection** This bit selects source for card detection 0_B SDCD# is selected (for normal use) 1_B The Card Detect Test Level is selected (for test purpose)

Power Control Register

Control the bus power for the Card. This register is applicable for an SD/eMMC mode.

PWR_CTRL
Power Control Register **(029$_H$)** **Application Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
SD_BUS_VOL_VDD2			SD_BUS_PWR_VDD2	SD_BUS_VOL_VDD1			SD_BUS_PWR_VDD1
rw			rw	rw			rw

Field	Bits	Type	Description
SD_BUS_PWR_VDD1	0	rw	**SD Bus Power for VDD1** Before setting this bit, the SD Host Driver shall set SD Bus Voltage Select. If the Host Controller detects the No Card state, this bit shall be cleared. In SD mode, if this bit is cleared, the Host Controller shall stop SD Clock by clearing SD_CLK_IN bit in CLK_CTRL_R register. 0_B Power off 1_B Power on
SD_BUS_VOL_VDD1	3:1	rw	**SD Bus Voltage Select for VDD1/eMMC Bus Voltage Select for VDD.** By setting these bits, the Host Driver selects the voltage level for the SD/eMMC card. Before setting this register, the Host Driver shall check the Voltage Support bits in the capabilities register. If an unsupported voltage is selected, the Host System shall not supply SD Bus voltage. The value set in this field is available on the DWC_mshc output signal (sd_vdd1_sel), which will be used by the voltage switching circuitry. 000_B Reserved … 110_B Reserved 111_B SD 3.3 V (Typ.) / eMMC 3.0 V (Typ.)
SD_BUS_PWR_VDD2	4	rw	**SD Bus Power for VDD2.** This bit needs to be written with '1'
SD_BUS_VOL_VDD2	7:5	rw	**SD Bus Voltage Select for VDD2.** This field needs to be written with 0b000 000_B VDD2 Not Supported 001_B Reserved … 101_B Reserved 110_B Not used 111_B Not used

Block Gap Control Register

The host driver shall use this register to control any operation related to Block Gap. This register is applicable for an SD/eMMC mode.

BGAP_CTRL
Block Gap Control Register **(02A$_H$)** **Application Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
RSVD_7_4				INT_AT_BGAP	RD_WAIT_CTRL	CONTINUE_REQ	STOP_BG_REQ
r				rw	rw	rw	rw

Field	Bits	Type	Description
STOP_BG_REQ	0	rw	**Stop At Block Gap Request** This bit is used to stop executing read and write transaction at the next block gap for non-DMA, SDMA and ADMA transfers. 0_B Transfer 1_B Stop
CONTINUE_REQ	1	rw	**Continue Request** This bit is used to restart the transaction, which was stopped using the Stop At Block Gap Request. The Host Controller automatically clears this bit when the transaction re-starts. If stop at block gap request is set to 1, any write to this bit is ignored. **Volatile:** true 0_B Not Affected 1_B Restart
RD_WAIT_CTRL	2	rw	**Read Wait Control (SD Mode only)** This bit is used to enable the read wait protocol to stop read data using DAT[2] line if the card supports read wait. Otherwise, the Host Controller has to stop the card clock to hold the read data. 0_B Disable Read Wait Control 1_B Enable Read Wait Control
INT_AT_BGAP	3	rw	**Interrupt At Block Gap (SD Mode only)** This bit is valid only in 4-bit mode of SDIO card and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. 0_B Disabled 1_B Enabled
RSVD_7_4	7:4	r	**Reserved** These bits of Block Gap Control register are reserved. They will always return 0.

Wakeup Control Register

This register is mandatory for the Host Controller, but wakeup functionality depends on the Host Controller system hardware and software. The Host Driver shall maintain voltage on the SD Bus, by setting SD Bus Power to 1 in the Power Control Register, when wakeup event through Card Interrupt is desired.

WUP_CTRL
Wakeup Control Register **(02B$_H$)** **Application Reset Value: 00$_H$**

7	6	5	4	3	2	1	0
RSVD_7_3					CARD_REMOVAL	CARD_INSERT	CARD_INT
r					rw	rw	rw

Field	Bits	Type	Description
CARD_INT	0	rw	**Wakeup Event Enable on Card Interrupt** This bit enables wakeup event through Card Interrupt assertion in the Normal Interrupt Status register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. 0_B Disable 1_B Enable
CARD_INSERT	1	rw	**Wakeup Event Enable on SD Card Insertion** This bit enables wakeup event through Card Insertion assertion in the Normal Interrupt Status register. FN_WUS (Wake Up Support) in CIS does not affect this bit. 0_B Disable 1_B Enable
CARD_REMOVAL	2	rw	**Wakeup Event Enable on SD Card Removal** This bit enables wakeup event through Card Removal assertion in the Normal Interrupt Status register.FN_WUS (Wake Up Support) in CIS does not affect this bit. 0_B Disable 1_B Enable
RSVD_7_3	7:3	r	**Reserved** These bits are reserved. They will always return 0.

Clock Control Register

This register controls SDCLK (card clock) in an SD/eMMC mode. This register is applicable for an SD/eMMC mode.

CLK_CTRL
Clock Control Register **(02C$_H$)** **Application Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREQ_SEL								UPPER_FREQ_SEL		CLK_GEN_SELECT	RSVD_4	PLL_ENABLE	SD_CLK_EN	INTERNAL_CLK_STABLE	INTERNAL_CLK_EN
rw								rw		rw	r	rw	rw	r	rw

Field	Bits	Type	Description
INTERNAL_CLK_EN	0	rw	**Internal Clock Enable** This bit is set to 0 when the Host Driver is not using the Host Controller or the Host Controller awaits a wakeup interrupt. The Host Controller should stop its internal clock to go to a very low power state. However, registers can still be able read and written to. 0_B Stop 1_B Oscillate
INTERNAL_CLK_STABLE	1	r	**Internal Clock Stable** Host Driver checks clock stability by this status twice after Internal Clock Enable is set and after PLL Enable is set. **Volatile:** true 0_B Not Ready 1_B Ready
SD_CLK_EN	2	rw	**SD/eMMC Clock Enable** This bit stops the SDCLK or RCLK when this bit is set to 0. SDCLK/RCLK Frequency Select can be changed when this bit is 0. 0_B Disable 1_B Enable
PLL_ENABLE	3	rw	**PLL Enable** This bit can be ignored. Must be written with '0'. 0_B PLL is in low power mode 1_B PLL is enabled
RSVD_4	4	r	**Reserved** This bit is reserved. It will always return 0.
CLK_GEN_SELECT	5	rw	**Clock Generator Select** This bit is used to select the clock generator mode in SDCLK/RCLK Frequency Select. If Preset Value Enable = 0, this field is set by Host Driver. If Preset Value Enable = 1, this field is automatically set to a value specified in one of the Preset Value registers. **Volatile:** true 0_B Divided Clock Mode 1_B Programmable Clock Mode
UPPER_FREQ_SEL	7:6	rw	**SDCLK/RCLK Frequency Select - FREQ_SEL** These bits together with FREQ_SEL are used to select the frequency of the SDCLK pin. This field depends on setting of Preset Value Enable in Host Control 2 register. If Preset Value Enable = 0, this field is set by Host Driver. If Preset Value Enable = 1, this field is automatically set to a value specified in one of the Preset Value register. SPB clock frequency is calculated with the formula =fSPB * 1/((N+1)*2). **Volatile:** true

Field	Bits	Type	Description
FREQ_SEL	15:8	rw	**SDCLK/RCLK Frequency Select** These bits together with UPPER_FREQ_SEL are used to select the frequency of the SDCLK pin. This field depends on setting of Preset Value Enable in Host Control 2 register. If Preset Value Enable = 0, this field is set by Host Driver. If Preset Value Enable = 1, this field is automatically set to a value specified in one of the Preset Value register. SPB clock frequency is calculated with the formula =fSPB * 1/((N+1)*2). **Volatile:** true **10-bit Divided Clock Mode:** • 3FFh - 1/2048 SPB Clock • ... - ... • N - 1/2(N+1) SPB Clock • ... - ... • 002h - 1/6 SPB Clock • 001h - 1/4 SPB Clock • 000h - 1/2 SPB Clock

Timeout Control Register

At the initialization of the Host Controller, the Host Driver shall set the Data Timeout Counter value for an SD/eMMC mode according to the timer clock defined in the Capabilities register.

TOUT_CTRL

Timeout Control Register **($02E_H$)** **Application Reset Value: 00_H**

7	6	5	4	3	2	1	0
RSVD_7_4				TOUT_CNT			
r				rw			

Field	Bits	Type	Description
TOUT_CNT	3:0	rw	**Data Timeout Counter Value.** This value determines the interval by which DAT line timeouts are detected. The Timeout is based on the base clock TMCLK, which is fSPB devided by 32. Timeout clock frequency will be generated by dividing the base clock TMCLK value by this value. When setting this register, prevent inadvertent timeout events by clearing the Data Timeout Error Status Enable (in the Error Interrupt Status Enable register) *Note: During a boot operating in an eMMC mode, an application should configure the boot data timeout value (approximately 1 sec) in this field.* 0_H TMCLK * 2^13 1_H TMCLK * 2^14 2_H TMCLK * 2^15 … E_H TMCLK * 2^27 F_H Reserved
RSVD_7_4	7:4	r	**Reserved** These bits are reserved. They will always return 0.

Software Reset Register

A reset pulse is generated when writing 1 to each bit of this register. After completing the reset, the Host Controller shall clear each bit. As it takes some time to complete software reset, the Host Driver shall confirm that these bits are 0. This register is applicable for an SD/eMMC mode.

SW_RST
Software Reset Register **($02F_H$)** **Application Reset Value: 00_H**

7	6	5	4	3	2	1	0
RSVD_7_3					SW_RST_DAT	SW_RST_CMD	SW_RST_ALL
r					rw	rw	rw

Field	Bits	Type	Description
SW_RST_ALL	0	rw	**Software Reset For All** This reset affects the entire Host Controller except for the card detection circuit. During its initialization, the Host Driver shall set this bit to 1 to reset the Host Controller. All registers are reset except the capabilities register. If this bit is set to 1, the Host Driver should issue reset command and reinitialize the SD/eMMC card. **Volatile:** true 0_B Work 1_B Reset

Field	Bits	Type	Description
SW_RST_CMD	1	rw	**Software Reset For CMD line** Only part of command circuit is reset to be able to issue a command. This reset is effective only for a command issuing circuit (including response error statuses related to Command Inhibit (CMD) control) and does not affect the data transfer circuit. Host Controller can continue data transfer even after this reset is executed during handling of subcommand response errors. The following registers and bits are cleared by this bit: - Present State register Command Inhibit (CMD) - Normal Interrupt Status register Command Complete - Error Interrupt Status Response error statuses related to Command Inhibit (CMD) **Volatile:** true 0_B Work 1_B Reset

Field	Bits	Type	Description
SW_RST_DAT	2	rw	**Software Reset For DAT line (SD/eMMC Mode only)** This bit resets only a part of the data circuit and the DMA circuit is also reset. The following registers and bits are cleared by this bit: • Buffer Data Port register Buffer is cleared and initialized. – Buffer is cleared and initialized. • Present state register Buffer Read Enable Buffer Write Enable Read Transfer Active Write Transfer Active DAT Line Active Command Inhibit (DAT) – Buffer Read Enable – Buffer Write Enable – Read Transfer Active – Write Transfer Active – DAT Line Active – Command Inhibit (DAT) • Block Gap Control register Continue Request Stop At Block Gap Request – Continue Request – Stop At Block Gap Request • Normal Interrupt status register Buffer Read Ready Buffer Write Ready DMA Interrupt Block Gap Event Transfer Complete – Buffer Read Ready – Buffer Write Ready – DMA Interrupt – Block Gap Event – Transfer Complete **Volatile:** true 0_B Work 1_B Reset
RSVD_7_3	7:3	r	**Reserved** These bits are reserved. They will always return 0. **Volatile:** true

Normal Interrupt Status Register

This register reflects the status of Normal Interrupt. This register is applicable for an SD/eMMC mode.

NORMAL_INT_STAT

Normal Interrupt Status Register **(030_H)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERR_INTERRUPT	CQE_EVENT	FX_EVENT	RE_TUNE_EVENT	INT_C	INT_B	INT_A	CARD_INTERRUPT	CARD_REMOVAL	CARD_INSERTION	BUF_RD_READY	BUF_WR_READY	DMA_INTERRUPT	BGAP_EVENT	XFER_COMPLETE	CMD_COMPLETE
r	rwh	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
CMD_COMPLETE	0	rwh	**Command Complete** In an SD/eMMC Mode, this bit is set when the end bit of a response except for Auto CMD12 and Auto CMD23. This interrupt is not generated when the Response Interrupt Disable in Transfer Mode Register is set to 1. A write of 1 clears this register field. **Volatile:** true 0_B No command complete 1_B Command Complete
XFER_COMPLETE	1	rwh	**Transfer Complete** This bit is set when a read/write transfer and a command with status busy is completed. A write of 1 clears this register field. **Volatile:** true 0_B Not complete 1_B Command execution is completed
BGAP_EVENT	2	rwh	**Block Gap Event** This bit is set when both read/write transaction is stopped at block gap due to Stop at Block Gap Request. A write of 1 clears this register field. **Volatile:** true 0_B No Block Gap Event 1_B Transaction stopped
DMA_INTERRUPT	3	rwh	**DMA Interrupt** This bit is set if the Host Controller detects the SDMA Buffer Boundary during transfer. In case of ADMA, by setting Int field in the descriptor table, Host controller generates this interrupt. This interrupt shall not be generated after Transfer Complete. A write of 1 clears this register field. **Volatile:** true 0_B No DMA Interrupt 1_B DMA Interrupt is generated

Field	Bits	Type	Description
BUF_WR_READY	4	rwh	**Buffer Write Ready** This bit is set if the Buffer Write Enable changes from 0 to 1. A write of 1 clears this register field. **Volatile:** true 0_B Not ready to write buffer 1_B Ready to write buffer
BUF_RD_READY	5	rwh	**Buffer Read Ready** This bit is set if the Buffer Read Enable changes from 0 to 1. A write of 1 clears this register field. **Volatile:** true 0_B Not ready to read buffer 1_B Ready to read buffer
CARD_INSERTION	6	rwh	**Card Insertion** This bit is set if the Card Inserted in the Present State register changes from 0 to 1. A write of 1 clears this register field. **Volatile:** true 0_B Card state stable or Debouncing 1_B Card Inserted
CARD_REMOVAL	7	rwh	**Card Removal** This bit is set if the Card Inserted in the Present State register changes from 1 to 0. A write of 1 clears this register field. **Volatile:** true 0_B Card state stable or Debouncing 1_B Card Removed
CARD_INTERRUPT	8	r	**Card Interrupt** This bit reflects the synchronized value of: DAT[1]. *Note: This reflection changes after the interrupt ocurred. Clearing the interrupt is only possible by clearing it inside the card while the CARD_INTERRUPT is disabled inside SDMMC.* **Volatile:** true 0_B No Card Interrupt 1_B Generate Card Interrupt
INT_A	9	r	**INT_A (Embedded)** This bit is set if INT_A is enabled. **Volatile:** true 0_B No Interrupt is detected 1_B INT_A is detected

Field	Bits	Type	Description
INT_B	10	r	**INT_B (Embedded)** This bit is set if INT_B is enabled. **Volatile:** true 0_B No Interrupt is detected 1_B INT_B is detected
INT_C	11	r	**INT_C (Embedded)** This bit is set if INT_C is enabled. **Volatile:** true 0_B No Interrupt is detected 1_B INT_C is detected
RE_TUNE_EVENT	12	r	**Re-tuning Event** This bit can be ignored. **Volatile:** true
FX_EVENT	13	r	**FX Event** This status is set when r[14] of response register is set to 1 and Response Type R1/R5 is set to 0 in Transfer Mode register. This interrupt is used with response check function. **Volatile:** true 0_B No Event 1_B FX Event is detected
CQE_EVENT	14	rwh	**Command Queuing Event** This bit can be ignored. 0_B No Event 1_B Command Queuing Event is detected
ERR_INTERRUPT	15	r	**Error Interrupt** If any of the bits in the Error Interrupt Status register are set, then this bit is set. **Volatile:** true 0_B No Error 1_B Error

Error Interrupt Status Register

Signals defined in this register can be enabled by the Error Interrupt Status Enable register, but not by the Error Interrupt Signal Enable register. The interrupt is generated when the Error Interrupt Status Enable is enabled and at least one of the statuses is set to 1. Writing to 1 clears the bit and writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. This register is applicable for an SD/eMMC mode.

ERROR_INT_STAT

Error Interrupt Status Register **(032_H)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VENDOR_ERR3	VENDOR_ERR2	VENDOR_ERR1	BOOT_ACK_ERR	RESP_ERR	TUNING_ERR	ADMA_ERR	AUTO_CMD_ERR	CUR_LMT_ERR	DATA_END_BIT_ERR	DATA_CRC_ERR	DATA_TOUT_ERR	CMD_IDX_ERR	CMD_END_BIT_ERR	CMD_CRC_ERR	CMD_TOUT_ERR
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_TOUT_ERR	0	rw	**Command Timeout Error (SD/eMMC Mode only)** This bit is set only if no response is returned within 64 SD clock cycles from the end bit of the command. If the Host Controller detects a CMD line conflict, clock cycles. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Time out
CMD_CRC_ERR	1	rw	**Command CRC Error (SD/eMMC Mode only)** Command CRC Error is generated in two cases. If a response is returned and the Command Timeout Error is set to 0 (indicating no timeout), this bit is set to 1 when detecting a CRC error in the command response. The Host Controller detects a CMD line conflict by monitoring the CMD line when a command is issued. If the Host Controller drives the CMD line to 1 level, but detects 0 level on the CMD line at the next SD clock edge, then the Host Controller shall abort the command (stop driving CMD line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B CRC Error Generated
CMD_END_BIT_ERR	2	rw	**Command End Bit Error (SD/eMMC Mode only)** This bit is set when detecting that the end bit of a command response is 0. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B End Bit Error Generated
CMD_IDX_ERR	3	rw	**Command Index Error (SD/eMMC Mode only)** This bit is set if a Command Index error occurs in the commandresponse. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Error

Field	Bits	Type	Description
DATA_TOUT_ERR	4	rw	**Data Timeout Error (SD/eMMC Mode only)** This bit is set when detecting one of the following timeout conditions: - Busy timeout for R1b, R5b type - Busy timeout after Write CRC status - Write CRC Status timeout - Read Data timeout A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Time out
DATA_CRC_ERR	5	rw	**Data CRC Error (SD/eMMC Mode only).** This error occurs when detecting CRC error when transferring read data which uses the DAT line, when detecting the Write CRC status having a value of other than 010 or when write CRC status timeout. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Error
DATA_END_BIT_ERR	6	rw	**Data End Bit Error (SD/eMMC Mode only)** This error occurs either when detecting 0 at the end bit position of read data that uses the DAT line or at the end bit position of the CRC status. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Error
CUR_LMT_ERR	7	rw	**Current Limit Error** By setting the SD Bus Power bit in the Power Control register, the Host Controller is requested to supply power for the SD Bus. If the Host Controller supports the Current Limit function, it can be protected from an illegal card by stopping power supply to the card in which case this bit indicates a failure status. A reading of 1 for this bit means that the Host Controller is not supplying power to the SD card due to some failure. A reading of 0 for this bit means that the Host Controller is supplying power and no error has occurred. The Host Controller may require some sampling time to detect the current limit. If the Host Controller does not support this function, this bit shall always be set to 0. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Power Fail

Field	Bits	Type	Description
AUTO_CMD_ERR	8	rw	**Auto CMD Error (SD/eMMC Mode only)** This error status is used by Auto CMD12 and Auto CMD23. This bit is set when detecting that any of the bits D00 to D05 in Auto CMD Error Status register has changed from 0 to 1. D07 is effective in case of Auto CMD12. Auto CMD Error Status register is valid while this bit is set to 1 and may be cleared by clearing of this bit (another implementation is also allowed). A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Error
ADMA_ERR	9	rw	**ADMA Error** This bit is set when the Host Controller detects error during ADMA-based data transfer. The error could be due to following reasons: • Error response received from System bus (Master I/F) • ADMA3,ADMA2 Descriptors invalid • CQE Task or Transfer descriptors invalid When the error occurs, the state of the ADMA is saved in the ADMA Error Status register. In eMMC CQE mode: The Host Controller generates this Interrupt when it detects an invalid descriptor data (Valid=0) at the ST_FDS state. ADMA Error State in the ADMA Error Status indicates that an error has occurred in ST_FDS state. The Host Driver may find that Valid bit is not set at the error descriptor. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Error
TUNING_ERR	10	rw	**Tuning Error** This bit can be ignored and needs to be written to with '0'. **Volatile:** true
RESP_ERR	11	rw	**Response Error (SD Mode only)** Host Controller Version 4.00 supports response error check function to avoid overhead of response error check by Host Driver during DMA execution. If Response Error Check Enable is set to 1 in the Transfer Mode register, Host Controller Checks R1 or R5 response. If an error is detected in a response, this bit is set to 1. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Error

Field	Bits	Type	Description
BOOT_ACK_ERR	12	rw	**Boot Acknowledgement Error (eMMC Mode only)** This bit is set when there is a timeout for boot acknowledgement or when detecting boot ack status having a value other than 010. This is applicable only when boot acknowledgement is expected and boot feature in DWC_mshc is enabled with DWC_MSHC_EMMC_BOOT_EN = 1. A write of 1 clears this register field. **Volatile:** true 0_B No Error 1_B Error
VENDOR_ERR1	13	rw	**Reserved** This bit can be ignored. Must be written with '0'. 0_B No Error 1_B Error
VENDOR_ERR2	14	rw	**Reserved** This bit can be ignored. Must be written with '0'. 0_B No Error 1_B Error
VENDOR_ERR3	15	rw	**Reserved** This bit can be ignored. Must be written with '0'. 0_B No Error 1_B Error

Normal Interrupt Status Enable Register

This register enables the Interrupt Status for Normal Interrupt Status register (NORMAL_INT_STAT_R) when NORMAL_INT_STAT_R is set to 1. This register is applicable for an SD/eMMC mode.

NORMAL_INT_STAT_EN
Normal Interrupt Status Enable Register **(034_H)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_15	**CQE_EVENT_STAT_EN**	**FX_EVENT_STAT_EN**	**RE_TUNE_EVENT_STAT_E**	**INT_C_STAT_EN**	**INT_B_STAT_EN**	**INT_A_STAT_EN**	**CARD_INTERRUPT_STAT_**	**CARD_REMOVAL_STAT_E**	**CARD_INSERTION_STAT_**	**BUF_RD_READY_STAT_EN**	**BUF_WR_READY_STAT_**	**DMA_INTERRUPT_STAT_**	**BGAP_EVENT_STAT_EN**	**XFER_COMPLETE_STAT_**	**CMD_COMPLETE_STAT_**
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_COMPLETE_STAT_EN	0	rw	**Command Complete Status Enable** 0_B Masked 1_B Enabled
XFER_COMPLETE_STAT_EN	1	rw	**Transfer Complete Status Enable** 0_B Masked 1_B Enabled

Field	Bits	Type	Description
BGAP_EVENT_STAT_EN	2	rw	**Block Gap Event Status Enable** 0_B Masked 1_B Enabled
DMA_INTERRUPT_STAT_EN	3	rw	**DMA Interrupt Status Enable** 0_B Masked 1_B Enabled
BUF_WR_READY_STAT_EN	4	rw	**Buffer Write Ready Status Enable** 0_B Masked 1_B Enabled
BUF_RD_READY_STAT_EN	5	rw	**Buffer Read Ready Status Enable** 0_B Masked 1_B Enabled
CARD_INSERTION_STAT_EN	6	rw	**Card Insertion Status Enable** 0_B Masked 1_B Enabled
CARD_REMOVAL_STAT_EN	7	rw	**Card Removal Status Enable** 0_B Masked 1_B Enabled
CARD_INTERRUPT_STAT_EN	8	rw	**Card Interrupt Status Enable** If this bit is set to 0, the Host Controller shall clear the interrupt request to the System. The Card Interrupt detection is stopped when this bit is cleared and restarted when this bit is set to 1. The Host Driver may clear the Card Interrupt Status Enable before servicing the Card Interrupt and may set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. By setting this bit to 0, interrupt input should be masked by implementation so that the interrupt input is not affected by external signal in any state (for example, floating). 0_B Masked 1_B Enabled
INT_A_STAT_EN	9	rw	**INT_A (Embedded) Status Enable** If this bit is set to 0, the Host Controller shall clear the interrupt request to the System. The Host Driver may clear this bit before servicing the INT_A and may set this bit again after all interrupt requests to INT_A pin are cleared to prevent inadvertent interrupts. 0_B Masked 1_B Enabled

Field	Bits	Type	Description
INT_B_STAT_EN	10	rw	**INT_B (Embedded) Status Enable.** If this bit is set to 0, the Host Controller shall clear the interrupt request to the System. The Host Driver may clear this bit before servicing the INT_B and may set this bit again after all interrupt requests to INT_B pin are cleared to prevent inadvertent interrupts. 0_B Masked 1_B Enabled
INT_C_STAT_EN	11	rw	**INT_C (Embedded) Status Enable.** If this bit is set to 0, the Host Controller shall clear the interrupt request to the System. The Host Driver may clear this bit before servicing the INT_C and may set this bit again after all interrupt requests to INT_C pin are cleared to prevent inadvertent interrupts. 0_B Masked 1_B Enabled
RE_TUNE_EVENT_STAT_EN	12	rw	**Re-Tuning Event Status Enable** This bit must always remain '0'.
FX_EVENT_STAT_EN	13	rw	**FX Event Status Enable.** This bit is added from Version 4.10. 0_B Masked 1_B Enabled
CQE_EVENT_STAT_EN	14	rw	**CQE Event Status Enable.** This bit must always remain '0'.
RSVD_15	15	r	**Reserved** This bit is reserved. It will always return 0.

Error Interrupt Status Enable Register

This register sets the Interrupt Status for Error Interrupt Status register (ERROR_INT_STAT_R), when ERROR_INT_STAT_EN_R is set to 1. This register is applicable for an SD/eMMC mode.

ERROR_INT_STAT_EN
Error Interrupt Status Enable Register **(036_H)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VENDOR_ER_STAT_EN3	VENDOR_ER_STAT_EN2	VENDOR_ER_STAT_EN1	BOOT_ACK_ERR_STAT_E	RESP_ERR_STAT_EN	TUNING_ERR_STAT_EN	ADMA_ERR_STAT_EN	AUTO_CMD_ERR_STAT_E	CUR_LMT_ER_STAT_EN	DATA_END_BIT_ER_STA	DATA_CRC_ERR_STAT_EN	DATA_TOUT_ERR_STAT_E	CMD_IDX_ER_STAT_EN	CMD_END_BIT_ER_STA	CMD_CRC_ERR_STAT_EN	CMD_TOUT_ERR_STAT_E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_TOUT_ERR_STAT_EN	0	rw	**Command Timeout Error Status Enable (SD/eMMC Mode only).** 0_B Masked 1_B Enabled
CMD_CRC_ERR_STAT_EN	1	rw	**Command CRC Error Status Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
CMD_END_BIT_ERR_STAT_EN	2	rw	**Command End Bit Error Status Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
CMD_IDX_ERR_STAT_EN	3	rw	**Command Index Error Status Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
DATA_TOUT_ERR_STAT_EN	4	rw	**Data Timeout Error Status Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
DATA_CRC_ERR_STAT_EN	5	rw	**Data CRC Error Status Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
DATA_END_BIT_ERR_STAT_EN	6	rw	**Data End Bit Error Status Enable (SD/eMMC Mode only).** 0_B Masked 1_B Enabled
CUR_LMT_ERR_STAT_EN	7	rw	**Current Limit Error Status Enable** 0_B Masked 1_B Enabled
AUTO_CMD_ERR_STAT_EN	8	rw	**Auto CMD Error Status Enable (SD/eMMC Mode only).** 0_B Masked 1_B Enabled
ADMA_ERR_STAT_EN	9	rw	**ADMA Error Status Enable.** 0_B Masked 1_B Enabled
TUNING_ERR_STAT_EN	10	rw	**Tuning Error Status Enable.** This bit must always remain '0'.

Field	Bits	Type	Description
RESP_ERR_STAT_EN	11	rw	**Response Error Status Enable (SD Mode only)** 0_B Masked 1_B Enabled
BOOT_ACK_ERR_STAT_EN	12	rw	**Boot Acknowledgment Error (eMMC Mode only).** Setting this bit to 1 enables setting of Boot Acknowledgment Error in Error Interrupt Status register (ERROR_INT_STAT_R). 0_B Masked 1_B Enabled
VENDOR_ERR_STAT_EN1	13	rw	**Reserved** This bit can be ignored. Must be written with '0'. 0_B Masked 1_B Enabled
VENDOR_ERR_STAT_EN2	14	rw	**Reserved** This bit can be ignored. Must be written with '0'. 0_B Masked 1_B Enabled
VENDOR_ERR_STAT_EN3	15	rw	**Reserved** This bit can be ignored. Must be written with '0'. 0_B Masked 1_B Enabled

Normal Interrupt Signal Enable Register

This register is used to select the interrupt status that is indicated to the Host System as the interrupt. All these status bits share the same 1-bit interrupt line. Setting any of these bits to 1, enables interrupt generation. This register is applicable for an SD/eMMC mode.

NORMAL_INT_SIGNAL_EN
Normal Interrupt Signal Enable Register **(038_H)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_15	**CQE_EVENT_SIGNAL_EN**	**FX_EVENT_SIGNAL_EN**	**RE_TUNE_EVENT_SIGNAL**	**INT_C_SIGNAL_EN**	**INT_B_SIGNAL_EN**	**INT_A_SIGNAL_EN**	**CARD_INTERRUPT_SIGNA**	**CARD_REMOVAL_SIGNAL**	**CARD_INSERTION_SIGNA**	**BUF_RD_READY_SIGNAL_**	**BUF_WR_READY_SIGNA**	**DMA_INTERRUPT_SIGNA**	**BGAP_EVENT_SIGNAL_EN**	**XFER_COMPLETE_SIGNA**	**CMD_COMPLETE_SIGNA**
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_COMPLETE_SIGNAL_EN	0	rw	**Command Complete Signal Enable** 0_B Masked 1_B Enabled
XFER_COMPLETE_SIGNAL_EN	1	rw	**Transfer Complete Signal Enable** 0_B Masked 1_B Enabled

Field	Bits	Type	Description
BGAP_EVENT_SIGNAL_EN	2	rw	**Block Gap Event Signal Enable** 0_B Masked 1_B Enabled
DMA_INTERRUPT_SIGNAL_EN	3	rw	**DMA Interrupt Signal Enable** 0_B Masked 1_B Enabled
BUF_WR_READY_SIGNAL_EN	4	rw	**Buffer Write Ready Signal Enable** 0_B Masked 1_B Enabled
BUF_RD_READY_SIGNAL_EN	5	rw	**Buffer Read Ready Signal Enable** 0_B Masked 1_B Enabled
CARD_INSERTION_SIGNAL_EN	6	rw	**Card Insertion Signal Enable** 0_B Masked 1_B Enabled
CARD_REMOVAL_SIGNAL_EN	7	rw	**Card Removal Signal Enable** 0_B Masked 1_B Enabled
CARD_INTERRUPT_SIGNAL_EN	8	rw	**Card Interrupt Signal Enable** 0_B Masked 1_B Enabled
INT_A_SIGNAL_EN	9	rw	**INT_A (Embedded) Signal Enable** 0_B Masked 1_B Enabled
INT_B_SIGNAL_EN	10	rw	**INT_B (Embedded) Signal Enable** 0_B Masked 1_B Enabled
INT_C_SIGNAL_EN	11	rw	**INT_C (Embedded) Signal Enable** 0_B Masked 1_B Enabled
RE_TUNE_EVENT_SIGNAL_EN	12	rw	**Re-Tuning Event Signal Enable.** This bit must always remain '0'.

Field	Bits	Type	Description
FX_EVENT_SIGNAL_EN	13	rw	**FX Event Signal Enable** 0_B Masked 1_B Enabled
CQE_EVENT_SIGNAL_EN	14	rw	**Command Queuing Engine Event Signal Enable** This bit must always remain '0'.
RSVD_15	15	r	**Reserved** This bit is reserved. It will always return 0.

Error Interrupt Signal Enable Register

This register is used to select the interrupt status that is notified to the Host System as an interrupt. All these status bits share the same 1-bit interrupt line. Setting any of these bits to 1 enables interrupt generation. This register is applicable for an SD/eMMC mode.

ERROR_INT_SIGNAL_EN
Error Interrupt Signal Enable Register **($03A_H$)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VENDOR_ERR_SIGNAL_E	VENDOR_ERR_SIGNAL_E	VENDOR_ERR_SIGNAL_E	BOOT_ACK_ERR_SIGNAL	RESP_ERR_SIGNAL_EN	TUNING_ERR_SIGNAL_EN	ADMA_ERR_SIGNAL_EN	AUTO_CMD_ERR_SIGNAL	CUR_LMT_ERR_SIG NAL_E	DATA_END_BIT_ERR_SIG	DATA_CRC_ERR_SIGNAL_	DATA_TOUT_ERR_SIGNAL	CMD_IDX_ERR_SIG NAL_E	CMD_END_BIT_ERR_SIG	CMD_CRC_ERR_SIGNAL_	CMD_TOUT_ERR_SIGNAL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CMD_TOUT_ERR_SIGNAL_EN	0	rw	**Command Timeout Error Signal Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
CMD_CRC_ERR_SIGNAL_EN	1	rw	**Command CRC Error Signal Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
CMD_END_BIT_ERR_SIGNAL_EN	2	rw	**Command End Bit Error Signal Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
CMD_IDX_ERR_SIGNAL_EN	3	rw	**Command Index Error Signal Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
DATA_TOUT_ERR_SIGNAL_EN	4	rw	**Data Timeout Error Signal Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled

Field	Bits	Type	Description
DATA_CRC_ERR_SIGNAL_EN	5	rw	**Data CRC Error Signal Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
DATA_END_BIT_ERR_SIGNAL_EN	6	rw	**Data End Bit Error Signal Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
CUR_LMT_ERR_SIGNAL_EN	7	rw	**Current Limit Error Signal Enable** 0_B Masked 1_B Enabled
AUTO_CMD_ERR_SIGNAL_EN	8	rw	**Auto CMD Error Signal Enable (SD/eMMC Mode only)** 0_B Masked 1_B Enabled
ADMA_ERR_SIGNAL_EN	9	rw	**ADMA Error Signal Enable** 0_B Masked 1_B Enabled
TUNING_ERR_SIGNAL_EN	10	rw	**Tuning Error Signal Enable** This bit must always remain '0'.
RESP_ERR_SIGNAL_EN	11	rw	**Response Error Signal Enable (SD Mode only)** 0_B Masked 1_B Enabled
BOOT_ACK_ERR_SIGNAL_EN	12	rw	**Boot Acknowledgment Error (eMMC Mode only).** Setting this bit to 1 enables generating interrupt signal when Boot Acknowledgement Error in Error Interrupt Status register is set. 0_B Masked 1_B Enabled
VENDOR_ERR_SIGNAL_EN1	13	rw	**Reserved** This bit can be ignored. Must be written with '0'. 0_B Masked 1_B Enabled
VENDOR_ERR_SIGNAL_EN2	14	rw	**Reserved** This bit can be ignored. Must be written with '0'. 0_B Masked 1_B Enabled
VENDOR_ERR_SIGNAL_EN3	15	rw	**Reserved** This bit can be ignored. Must be written with '0'. 0_B Masked 1_B Enabled

Auto CMD Status Register

This register is used to indicate the CMD12 response error of Auto CMD12, and the CMD23 response error of Auto CMD23. The Host driver can determine the kind of Auto CMD12/CMD23 errors that can occur in this register. Auto CMD23 errors are indicated in bit 04-01. This register is valid only when Auto CMD Error is set. This register is applicable for an SD/eMMC mode. Writing this register has no effect.

AUTO_CMD_STAT
Auto CMD Status Register **($03C_H$)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_15_8								CMD_NOT_ISSUED_AUTO	RSVD_6	AUTO_CMD_RESP_ERR	AUTO_CMD_IDX_ERR	AUTO_CMD_EBIT_ERR	AUTO_CMD_CRC_ERR	AUTO_CMD_TOUT_ERR	AUTO_CMD12_NOT_EXEC
r								r	r	r	r	r	r	r	r

Field	Bits	Type	Description
AUTO_CMD12_NOT_EXEC	0	r	**Auto CMD12 Not Executed** If multiple memory block data transfer is not started due to a command error, this bit is not set because it is not necessary to issue an Auto CMD12. Setting this bit to 1 means that the Host Controller cannot issue Auto CMD12 to stop multiple memory block data transfer, due to some error. If this bit is set to 1, error status bits (D04-D01) is meaningless. This bit is set to 0 when Auto CMD Error is generated by Auto CMD23. **Volatile:** true 0_B Executed 1_B Not Executed
AUTO_CMD_TOUT_ERR	1	r	**Auto CMD Timeout Error** This bit is set if no response is returned with 64 SDCLK cycles from the end bit of the command. If this bit is set to 1, error status bits (D04 - D01) are meaningless. **Volatile:** true 0_B No Error 1_B Time out
AUTO_CMD_CRC_ERR	2	r	**Auto CMD CRC Error** This bit is set when detecting a CRC error in the command response. **Volatile:** true 0_B No Error 1_B CRC Error Generated
AUTO_CMD_EBIT_ERR	3	r	**Auto CMD End Bit Error** This bit is set when detecting that the end bit of command response is 0. **Volatile:** true 0_B No Error 1_B End Bit Error Generated

Field	Bits	Type	Description
AUTO_CMD_IDX_ERR	4	r	**Auto CMD Index Error** This bit is set if the command index error occurs in response to a command. **Volatile:** true 0_B No Error 1_B Error
AUTO_CMD_RESP_ERR	5	r	**Auto CMD Response Error** This bit is set when Response Error Check Enable in the Transfer Mode register is set to 1 and an error is detected in R1 response of either Auto CMD12 or CMD13. This status is ignored if any bit between D00 to D04 is set to 1. **Volatile:** true 0_B No Error 1_B Error
RSVD_6	6	r	**Reserved** This bit is reserved. It will always return 0. **Volatile:** true
CMD_NOT_ISSUED_AUTO_CMD12	7	r	**Command Not Issued By Auto CMD12 Error** If this bit is set to 1, CMD_wo_DAT is not executed due to an Auto CMD12 Error (D04-D01) in this register. This bit is set to 0 when Auto CMD Error is generated by Auto CMD23. **Volatile:** true 0_B No Error 1_B Not Issued
RSVD_15_8	15:8	r	**Reserved** These bits are reserved. They will always return 0. **Volatile:** true

Host Control 2 Register

This register is used to control how the Host Controller operates. This register is applicable for an SD/eMMC mode.

HOST_CTRL2
Host Control 2 Register **($03E_H$)** **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRESET_VAL_ENABLE	ASYNC_INT_ENABLE	ADDRESSING	HOST_VER4_ENABLE	CMD23_ENABLE	ADMA2_LEN_MODE	RSVD_9	UHS2_IF_ENABLE	SAMPLE_CLK_SEL	EXEC_TUNING	DRV_STRENGTH_SEL		SIGNALING_EN	UHS_MODE_SEL		
rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw		rw	rw		

Field	Bits	Type	Description
UHS_MODE_SEL	2:0	rw	**eMMC Speed Mode Select - MODE_SEL** This field is reserved and needs to be written to with '000'. 000_B Standard setting 001_B reserved 010_B Reserved … 111_B Reserved
SIGNALING_EN	3	rw	**1.8 V Signaling Enable** This bit must remain '0'.
DRV_STRENGTH_SEL	5:4	rw	**Driver Strength Select** This bitfield can be ignored.
EXEC_TUNING	6	rw	**Execute Tuning** This bit must remain '0'.
SAMPLE_CLK_SEL	7	rw	**Sampling Clock Select** This bit must remain '0'.
UHS2_IF_ENABLE	8	rw	**UHS-II Interface Enable** This bit must remain '0'.
RSVD_9	9	r	**Reserved** This bit is reserved. It will always return 0.
ADMA2_LEN_MODE	10	rw	**ADMA2 Length Mode** This bit selects ADMA2 Length mode to be either 16-bit or 26-bit. 0_B 16-bit Data Length Mode 1_B 26-bit Data Length Mode
CMD23_ENABLE	11	rw	**CMD23 Enable** If the card supports CMD23, this bit is set to 1. This bit is used to select Auto CMD23 or Auto CMD12 for ADMA3 data transfer. 0_B Auto CMD23 is disabled 1_B Auto CMD23 is enabled

Field	Bits	Type	Description
HOST_VER4_ENABLE	12	rw	**Host Version 4 Enable** This bit selects either Version 3.00 compatible mode or Version 4 mode. Functions of following fields are modified for Host Version 4 mode: • SDMA Address: SDMA uses ADMA System Address ($05F_H$-058_H) instead of SDMA System Address register (003_H-000_H) • ADMA2/ADMA3 selection: ADMA3 is selected by DMA select in Host Control 1 register • 32-bit Block Count: SDMA System Address register (003_H-000_H) is modified to 32-bit Block Count register *Note: It is recommended not to program ADMA3 Integrated Descriptor Address registers while operating in Host version less than 4 mode (Host Version 4 Enable = 0).* 0_B Version 3.00 compatible mode 1_B Version 4 mode
ADDRESSING	13	rw	**64-bit Addressing** This bit is effective when Host Version 4 Enable is set to 1. 0_B 32 bits addressing 1_B Reserved
ASYNC_INT_ENABLE	14	rw	**Asynchronous Interrupt Enable** This bit can be set if a card supports asynchronous interrupts and Asynchronous Interrupt Support is set to 1 in the Capabilities register. 0_B Disabled 1_B Enabled
PRESET_VAL_ENABLE	15	rw	**Preset Value Enable** This bit enables automatic selection of SDCLK frequency and Driver strength Preset Value registers. When Preset Value Enable is set, SDCLK frequency generation (Frequency Select and Clock Generator Select) amd driver strength selection is performed by the controller. These values are selected from set of Preset Value registers based on selected speed mode. During initialization, the host driver needs to manually set the clock preset value to SDCLK/Rclk frequency select. Preset value enable can be set after intialization completed, in accordance with PartA2_SD Host_controller_simplified_specification. 0_B SDCLK and Driver Strength are controlled by Host Driver 1_B Automatic Selection of SDCLK frequency and Driver Strength as Preset Values are Enabled

Capabilities 1 Register 0 to 31

This register provides the Host Driver with information specific to the Host Controller implementation. The host controller implements these values as fixed. Capabilities register is segregated into two 32-bit registers:

CAPABILITIES1_R and CAPABILITIES2_R. The CAPABILITIES1_R register is the lower part of Capabilities register. Writing this register has no effect.

CAPABILITIES1

Capabilities 1 Register 0 to 31 **(040_H)** **Application Reset Value: 216C 6483$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOT_TYPE		ASYNC_INT_SUPPORT	SYS_ADDR_64_V3	SYS_ADDR_64_V4	VOLT_18	VOLT_30	VOLT_33	SUS_RES_SUPPORT	SDMA_SUPPORT	HIGH_SPEED_SUPPORT	RSVD_20	ADMA2_SUPPORT	Embedded_8_BIT	MAX_BLK_LEN	
r		r	r	r	r	r	r	r	r	r	r	r	r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASE_CLK_FREQ								TOUT_CLK_UNIT	RSVD_6	TOUT_CLK_FREQ					
r								r	r	r					

Field	Bits	Type	Description
TOUT_CLK_FREQ	5:0	r	**Timeout Clock Frequency** This bit shows the base clock frequency used to detect Data Timeout Error. The Timeout Clock unit defines the unit of the field value: • Timeout Clock Unit = 0 [KHz] unit: 1 KHz to 63 KHz • Timeout Clock Unit = 1 [MHz] unit: 1 MHz to 63 MHz • Not 0 unit: 1 KHz to 63 KHz or 1 MHz to 63 MHz • 00 0000b - Get information through another method
RSVD_6	6	r	**Reserved** This bit is reserved. It will always return 0.
TOUT_CLK_UNIT	7	r	**Timeout Clock Unit** This bit shows the unit of base clock frequency used to detect Data TImeout Error. 0_B kHz 1_B MHz

Field	Bits	Type	Description
BASE_CLK_FREQ	15:8	r	**Base Clock Frequency for SD clock** This value indicates the base (maximum) clock frequency for SD Clock. The definition of this field depends on Host Controller Version. • 6-Bit Base Clock Frequency: This mode is supported by the Host Controller version 1.00 and 2.00. The upper 2 bits are not effective and are always 0. The unit values are 1 MHz. The supported clock range is 10 MHz to 63 MHz. 11xx xxxb - Not Supported 0011 1111b - 63 MHz ..... - ..... 0000 0010 - 2 MHz 0000 001b - 1 MHz 0000 0000 - Get information through another method • 8-Bit Base Clock Frequency: This mode is supported by the Host Controller version 3.00. The unit values are 1 MHz. The supported clock range is 10 MHz to 255 MHz. FFh - 255 MHz ..... - ..... 02h - 2 MHz 01h - 1 MHz 00h - Get information through another method If the frequency is 16.5 MHz, the larger value shall be set to 0001001b (17 MHz) because the Host Driver uses this value to calculate the clock divider value and it shall not exceed the upper limit of the SD Clock frequency. If these bits are all 0, the Host system has to get information using a different method.
MAX_BLK_LEN	17:16	r	**Maximum Block Length** This bit indicates the maximum block size that the Host driver can read and write to the buffer in the Host Controller. The buffer transfers this block size without wait cycles. The transfer block length is always 512 bytes for the SD Memory irrespective of this bit. 00_B 512 Byte 01_B 1024 Byte 10_B 2048 Byte 11_B Reserved
Embedded_8_BIT	18	r	**8-bit Support for Embedded Device** This bit indicates whether the Host Controller is capable of using an 8-bit bus width mode. This bit is not effective when the Slot Type is set to 10b. 0_B 8-bit Bus Width not Supported 1_B 8-bit Bus Width Supported

Field	Bits	Type	Description
ADMA2_SUPPORT	19	r	**ADMA2 Support** This bit indicates whether the Host Controller is capable of using ADMA2. 0_B ADMA2 not Supported 1_B ADMA2 Supported
RSVD_20	20	r	**Reserved** This bit is reserved. It will always return 0.
HIGH_SPEED_SUPPORT	21	r	**High Speed Support** This bit indicates whether the Host Controller and the Host System supports High Speed mode and they can supply the SD Clock frequency from 25 MHz to 50 MHz. 0_B High Speed not Supported 1_B High Speed Supported
SDMA_SUPPORT	22	r	**SDMA Support** This bit indicates whether the Host Controller is capable of using SDMA to transfer data between the system memory and the Host Controller directly. 0_B SDMA not Supported 1_B SDMA Supported
SUS_RES_SUPPORT	23	r	**Suspense/Resume Support** This bit indicates whether the Host Controller supports Suspend/Resume functionality. If this bit is 0, the Host Driver shall not issue either Suspend or Resume commands because the Suspend and Resume mechanism is not supported. 0_B Not Supported 1_B Supported
VOLT_33	24	r	**Voltage Support 3.3 V** 0_B 3.3 V Not Supported 1_B 3.3 V Supported
VOLT_30	25	r	**Voltage Support 3.0 V** 0_B 3.0 V Not Supported 1_B 3.0 V Supported
VOLT_18	26	r	**Voltage Support 1.8 V** 0_B 1.8 V Not Supported 1_B 1.8 V Supported

Field	Bits	Type	Description
SYS_ADDR_64_V4	27	r	**64-bit System Address Support for V4** This bit sets the Host Controller to support 64-bit System Addressing of V4 mode. 0_B 64-bit System Address for V4 is Not Supported 1_B 64-bit System Address for V4 is Supported
SYS_ADDR_64_V3	28	r	**64-bit System Address Support for V3** This bit sets Host Controller to support 64-bit System Addressing of V3 mode. SDMA cannot be used in 64-bit Addressing in Version 3 Mode. 0_B 64-bit System Address for V3 is Not Supported 1_B 64-bit System Address for V3 is Supported
ASYNC_INT_SUPPORT	29	r	**Asynchronous Interrupt Support (SD Mode only)** 0_B Asynchronous Interrupt Not Supported 1_B Asynchronous Interrupt Supported
SLOT_TYPE	31:30	r	**Slot Type** This field indicates usage of a slot by a specific Host System. 00_B Removable Card Slot 01_B Embedded Slot for one Device 10_B Shared Bus Slot (SD mode) 11_B Reserved

Capabilities Register 32 to 63

This register provides the Host Driver with information specific to the Host Controller implementation. The host controller implements these values as fixed. Capabilities register is segregated into two 32-bit registers, namely CAPABILITIES1_R and CAPABILITIES2_R. The CAPABILITIES2_R register is upper part of Capabilities register. Writing this register has no effect.

CAPABILITIES2
Capabilities Register 32 to 63 **(044_H)** **Application Reset Value: $0800\ 0007_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD_62_63		RSVD_61	VDD2_18V_SUPPORT	ADMA3_SUPPORT	RSVD_56_58			CLK_MUL							
r		r	r	r	r			r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RE_TUNING_MODES		USE_TUNING_SDR50	RSVD_44	RETUNE_CNT				RSVD_39	DRV_TYPED	DRV_TYPEC	DRV_TYPEA	UHS2_SUPPORT	DDR50_SUPPORT	SDR104_SUPPORT	SDR50_SUPPORT
r		r	r	r				r	r	r	r	r	r	r	r

Field	Bits	Type	Description
SDR50_SUPPORT	0	r	**SDR50 Support** If SDR104 is supported, this bit shall be set to 1. Bit 45 indicates whether SDR50 requires tuning or not. 0_B SDR50 is not supported 1_B SDR50 is supported
SDR104_SUPPORT	1	r	**SDR104 Support** SDR104 requires tuning. 0_B SDR104 is not supported 1_B SDR104 is supported
DDR50_SUPPORT	2	r	**DDR50 Support** 0_B DDR50 is not supported 1_B DDR50 is supported
UHS2_SUPPORT	3	r	**UHS-II Support** This bit indicates whether Host Controller supports UHS-II. 0_B UHS-II is not supported 1_B UHS-II is supported
DRV_TYPEA	4	r	**Driver Type A Support** This bit indicates support of Driver Type A for 1.8V Signaling. 0_B Driver Type A is not supported 1_B Driver Type A is supported
DRV_TYPEC	5	r	**Driver Type C Support** This bit indicates support of Driver Type C for 1.8 V Signaling. 0_B Driver Type C is not supported 1_B Driver Type C is supported
DRV_TYPED	6	r	**Driver Type D Support** This bit indicates support of Driver Type D for 1.8 V Signaling. 0_B Driver Type D is not supported 1_B Driver Type D is supported
RSVD_39	7	r	**Reserved** This bit is reserved. It will always return 0.

Field	Bits	Type	Description
RETUNE_CNT	11:8	r	**Timer Count for Re-Tuning** 0_H Re-Tuning Timer disabled 1_H 1 seconds 2_H 2 seconds 3_H 2^2 seconds … B_H 2^10 seconds C_H Reserved … E_H Reserved F_H Get information from other source
RSVD_44	12	r	**Reserved** This bit is reserved. It will always return 0.
USE_TUNING_SDR50	13	r	**Use Tuning for SDR50** 0_B SDR50 does not require tuning 1_B SDR50 requires tuning
RE_TUNING_MODES	15:14	r	**Re-Tuning Modes** This field selects re-tuning method and limits the maximum data length. 00_B Mode 1 - Timer 01_B Mode 2 - Timer and Re-Tuning Request 10_B Mode 3 - Auto Re-Tuning (for transfer) Timer and Re-Tuning Request 11_B Reserved
CLK_MUL	23:16	r	**Clock Multiplier** This field indicates clock multiplier of programmable clock generator. Setting to 0 means that Host Controller does not support programmable clock generator. 00_H Clock Multiplier is not Supported 01_H Clock Multiplier M = 2 02_H Clock Multiplier M = 3 … FF_H Clock Multiplier M = 256
RSVD_56_58	26:24	r	**Reserved** These bits are reserved. They will always return 0.
ADMA3_SUPPORT	27	r	**ADMA3 Support** This bit indicates whether the Host Controller is capable of using ADMA3. 0_B ADMA3 not Supported 1_B ADMA3 Supported

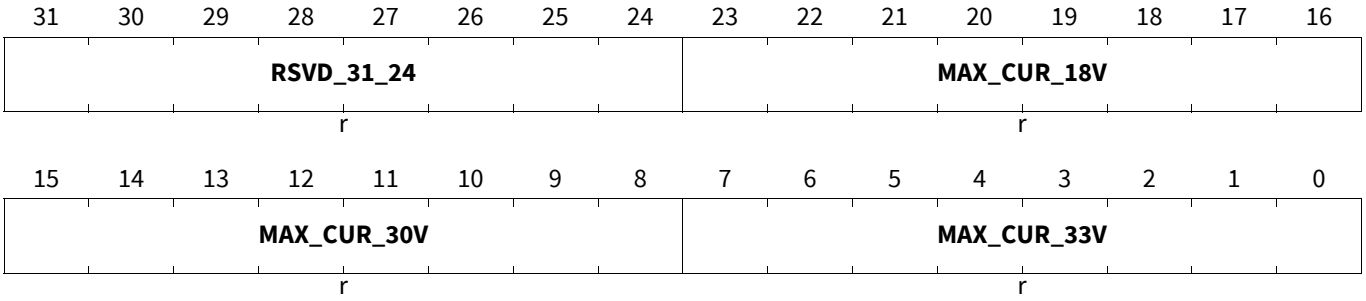
Field	Bits	Type	Description
VDD2_18V_SUPPORT	28	r	**1.8 V VDD2 Support** This bit indicates support of VDD2 for Host System. 0_B 1.8 V VDD2 is not Supported 1_B 1.8 V VDD2 is Supported
RSVD_61	29	r	**Reserved** This bit is reserved. It will always return 0.
RSVD_62_63	31:30	r	**Reserved** These bits are reserved. They will always return 0.

Maximum Current Capabilities Register 0 to 31

For VDD1, these registers indicate maximum current capability for each voltage. The value is meaningful if Voltage Support is set in the Capabilities register. If this information is supplied by the Host System through another method, all Maximum Current Capabilities register shall be 0. Writing this register has no effect.

CURR_CAPABILITIES1

Maximum Current Capabilities Register 0 to 31 (048_H) **Application Reset Value: 0000 0002_H**



Field	Bits	Type	Description
MAX_CUR_33V	7:0	r	**Maximum Current for 3.3 V** 00_H Get Information through another method 01_H 4 mA 02_H 8 mA … FF_H 1020 mA
MAX_CUR_30V	15:8	r	**Maximum Current for 3.0 V** 00_H Get Information through another method 01_H 4 mA 02_H 8 mA … FF_H 1020 mA

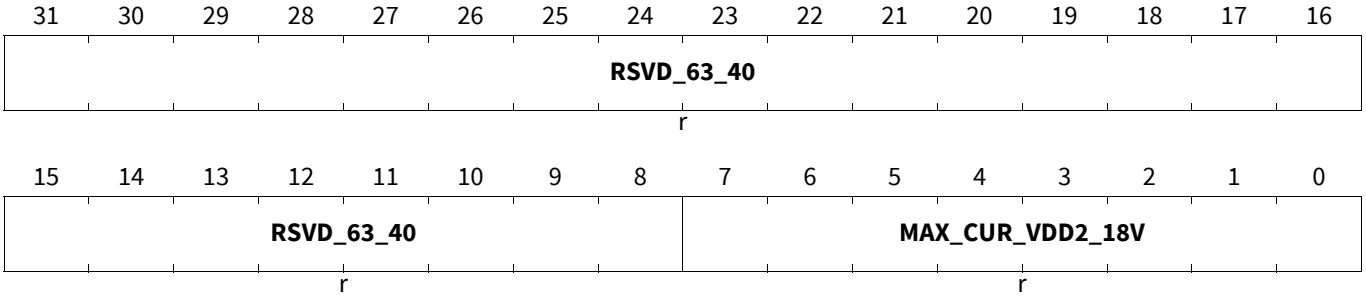
Field	Bits	Type	Description
MAX_CUR_18V	23:16	r	**Maximum Current for 1.8 V** 00_H Get Information through another method 01_H 4 mA 02_H 8 mA … FF_H 1020 mA
RSVD_31_24	31:24	r	**Reserved** These bits are reserved. They will always return 0.

Maximum Current Capabilities Register 32 to 63

For VDD2, these registers indicate maximum current capability for each voltage. The value is meaningful if Voltage Support is set in the Capabilities register. If this information is supplied by the Host System through another method, all Maximum Current Capabilities register shall be 0. Writing this register has no effect.

CURR_CAPABILITIES2

Maximum Current Capabilities Register 32 to 63 ($04C_H$) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
MAX_CUR_VDD2_18V	7:0	r	**Maximum Current for 1.8 V VDD2** 00_H Get Information through another method 01_H 4 mA 02_H 8 mA … FF_H 1020 mA
RSVD_63_40	31:8	r	**Reserved** These bits are reserved. They will always return 0.

Force Event Register for Auto CMD Error Status register

The register is not physically implemented, but is an address at which the Auto CMD Error Status register can be written. This register is applicable for an SD/eMMC mode.

- 1 : Sets each bit of the Auto CMD Error Status register
- 0 : No effect

FORCE_AUTO_CMD_STAT

Force Event Register for Auto CMD Error Status register(050_H) **Application Reset Value: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_15_8								FORCE_CMD_NOT_ISSUED	RSVD_6	FORCE_AUTO_CMD_RESP_	FORCE_AUTO_CMD_IDX_E	FORCE_AUTO_CMD_EBIT_	FORCE_AUTO_CMD_CRC_E	FORCE_AUTO_CMD_TOUT_	FORCE_AUTO_CMD12_NO
w								w	w	w	w	w	w	w	w

Field	Bits	Type	Description
FORCE_AUTO_CMD12_NOT_EXEC	0	w	**Force Event for Auto CMD12 Not Executed** 0_B Not Affected 1_B Auto CMD12 Not Executed Status is set
FORCE_AUTO_CMD_TOUT_ERR	1	w	**Force Event for Auto CMD Timeout Error** 0_B Not Affected 1_B Force Event for Auto CMD Timeout Error Status is set
FORCE_AUTO_CMD_CRC_ERR	2	w	**Force Event for Auto CMD CRC Error** 0_B Not Affected 1_B Force Event for Auto CMD CRC Error Status is set
FORCE_AUTO_CMD_EBIT_ERR	3	w	**Force Event for Auto CMD End Bit Error** 0_B Not Affected 1_B Force Event for Auto CMD End Bit Error Status is set
FORCE_AUTO_CMD_IDX_ERR	4	w	**Force Event for Auto CMD Index Error** 0_B Not Affected 1_B Force Event for Auto CMD Index Error Status is set
FORCE_AUTO_CMD_RESP_ERR	5	w	**Force Event for Auto CMD Response Error** 0_B Not Affected 1_B Force Event for Auto CMD Response Error Status is set
RSVD_6	6	w	**Reserved** This bit is reserved. It will always return 0.
FORCE_CMD_NOT_ISSUED_AUTO_CMD12	7	w	**Force Event for Command Not Issued By Auto CMD12 Error** 0_B Not Affected 1_B Command Not Issued By Auto CMD12 Error Status is set
RSVD_15_8	15:8	w	**Reserved** These bits are reserved. They will always return 0.

Force Event Register for Error Interrupt Status

This register is not physically implemented, but is an address at which the Error Interrupt Status register can be written. The effect of a write to this address will be reflected in the Error Interrupt Status register if the corresponding bit of the Error Interrupt Status Enable register is set. This register is applicable for an SD/eMMC mode.

- 1 - Sets each bit of the Error Interrupt Status register
- 0 - No effect

FORCE_ERROR_INT_STAT
Force Event Register for Error Interrupt Status (052$_H$) **Application Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FORCE_VENDOR_ER R3	FORCE_VENDOR_ER R2	FORCE_VENDOR_ER R1	FORCE_BOOT_ACK_ERR	FORCE_RESP_ERR	FORCE_TUNING_ER R	FORCE_ADMA_ERR	FORCE_AUTO_CMD_ERR	FORCE_CUR_LMT_ERR	FORCE_DATA_END_BIT_E	FORCE_DATA_CRC_ERR	FORCE_DATA_TOUT_ERR	FORCE_CMD_IDX_ERR	FORCE_CMD_END_BIT_ER	FORCE_CMD_CRC_ERR	FORCE_CMD_TOUT_ERR
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Field	Bits	Type	Description
FORCE_CMD_TOUT_ERR	0	w	**Force Event for Command Timeout Error (SD/eMMC Mode only)** 0_B Not Affected 1_B Command Timeout Error Status is set
FORCE_CMD_CRC_ERR	1	w	**Force Event for Command CRC Error (SD/eMMC Mode only)** 0_B Not Affected 1_B Command CRC Error Status is set
FORCE_CMD_END_BIT_ERR	2	w	**Force Event for Command End Bit Error (SD/eMMC Mode only)** 0_B Not Affected 1_B Command End Bit Error Status is set
FORCE_CMD_IDX_ERR	3	w	**Force Event for Command Index Error (SD/eMMC Mode only)** 0_B Not Affected 1_B Command Index Error Status is set
FORCE_DATA_TOUT_ERR	4	w	**Force Event for Data Timeout Error (SD/eMMC Mode only)** 0_B Not Affected 1_B Data Timeout Error Status is set
FORCE_DATA_CRC_ERR	5	w	**Force Event for Data CRC Error (SD/eMMC Mode only)** 0_B Not Affected 1_B Data CRC Error Status is set
FORCE_DATA_END_BIT_ERR	6	w	**Force Event for Data End Bit Error (SD/eMMC Mode only)** 0_B Not Affected 1_B Data End Bit Error Status is set

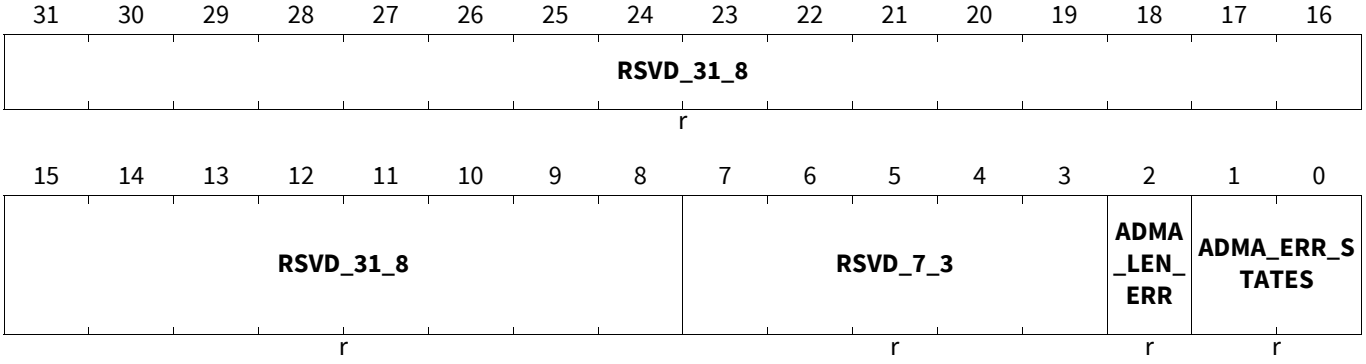
Field	Bits	Type	Description
FORCE_CUR_LMT_ERR	7	w	**Force Event for Current Limit Error** 0_B Not Affected 1_B Current Limit Error Status is set
FORCE_AUTO_CMD_ERR	8	w	**Force Event for Auto CMD Error (SD/eMMC Mode only)** 0_B Not Affected 1_B Auto CMD Error Status is set
FORCE_ADMA_ERR	9	w	**Force Event for ADMA Error** 0_B Not Affected 1_B ADMA Error Status is set
FORCE_TUNING_ERR	10	w	**Force Event for Tuning Error** This bit must remain '0'.
FORCE_RESP_ERR	11	w	**Force Event for Response Error (SD Mode only)** 0_B Not Affected 1_B Response Error Status is set
FORCE_BOOT_ACK_ERR	12	w	**Force Event for Boot Ack error.** 0_B Not Affected 1_B Vendor Specific Error Status is set
FORCE_VENDOR_ERR1	13	w	**Reserved** This bit is reserved. It will always return 0. 0_B Not Affected 1_B Vendor Specific Error Status is set
FORCE_VENDOR_ERR2	14	w	**Reserved** This bit is reserved. It will always return 0. 0_B Not Affected 1_B Vendor Specific Error Status is set
FORCE_VENDOR_ERR3	15	w	**Reserved** This bit is reserved. It will always return 0. 0_B Not Affected 1_B Vendor Specific Error Status is set

ADMA Error Status Register

This register holds the ADMA state during ADMA error. This register is applicable for an SD/eMMC mode. Writing this register has no effect.

ADMA_ERR_STAT

ADMA Error Status Register **(054_H)** **Application Reset Value: $0000\ 0000_H$**



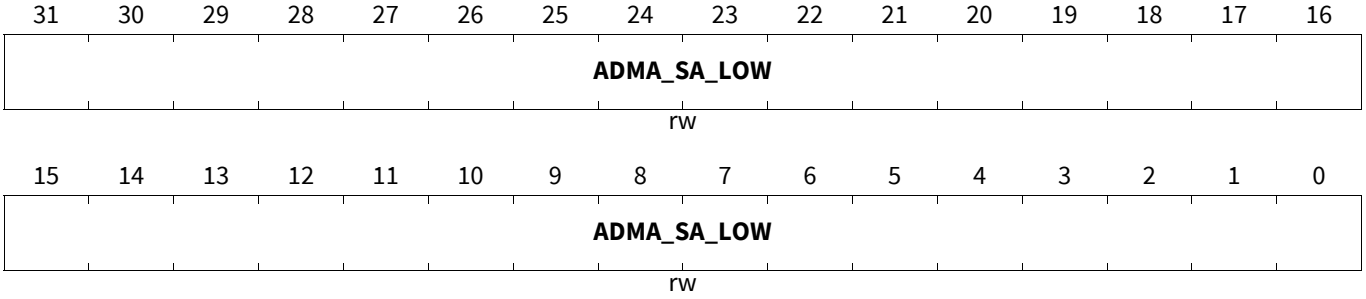
Field	Bits	Type	Description
ADMA_ERR_STATES	1:0	r	**ADMA Error States** These bits indicate the state of ADMA when an error occurs during ADMA data transfer. **Volatile:** true 00_B Stop DMA (SYS_ADR register points to a location next to the error descriptor) 01_B Fetch Descriptor (SYS_ADR register points to the error descriptor) 10_B Never set this state (Not used) 11_B Transfer Data (SYS_ADR register points to a location next to the error descriptor)
ADMA_LEN_ERR	2	r	**ADMA Length Mismatch Error States** This error occurs in the following instances: • While Block Count Enable is being set, the total data length specified by the Descriptor table is different from that specified by the Block Count and Block Length • Total data length cannot be divided by the block length **Volatile:** true 0_B No Error 1_B Error
RSVD_7_3	7:3	r	**Reserved** These bits are reserved. They will always return 0. **Volatile:** true
RSVD_31_8	31:8	r	**Reserved** These bits are reserved. They will always return 0. Volatile: true

ADMA System Address Register Low

This register holds the lower 32-bit system address for DMA transfer. This register is applicable for an SD/eMMC mode.

ADMA_SA_LOW

ADMA System Address Register Low **(058_H)** **Application Reset Value: 0000 0000_H**



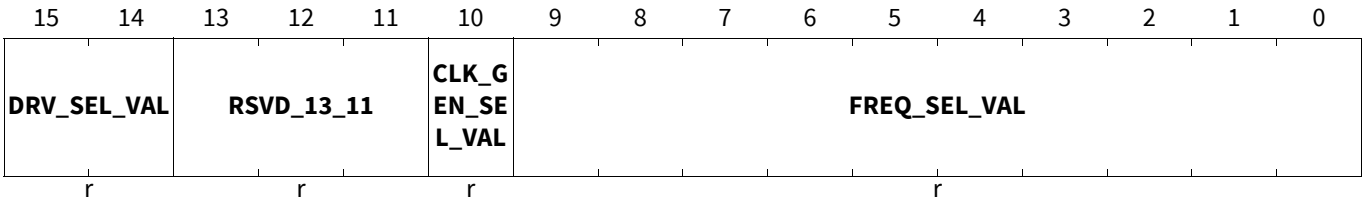
Field	Bits	Type	Description
ADMA_SA_LOW	31:0	rw	**ADMA System Address** These bits indicate the lower 32 bits of the ADMA system address. • SDMA If Host Version 4 Enable is set to 1, this register holds the system address of the data location • ADMA2 This register holds byte address of executing command of the descriptor table • ADMA3 This register is set by ADMA3. ADMA2 increments the address of this register that points to the next line, every time a Descriptor line is fetched. **Volatile:** true

Preset Value for Initialization

This register defines Preset Value for Initialization in SD/eMMC mode. Writing this register has no effect.

PRESET_INIT

Preset Value for Initialization **(060_H)** **Application Reset Value: $007F_H$**



Field	Bits	Type	Description
FREQ_SEL_VAL	9:0	r	**SDCLK/RCLK Frequency Select Value** 10-bit preset value to be set in SDCLK/RCLK Frequency Select field of the Clock Control register described by a Host System.

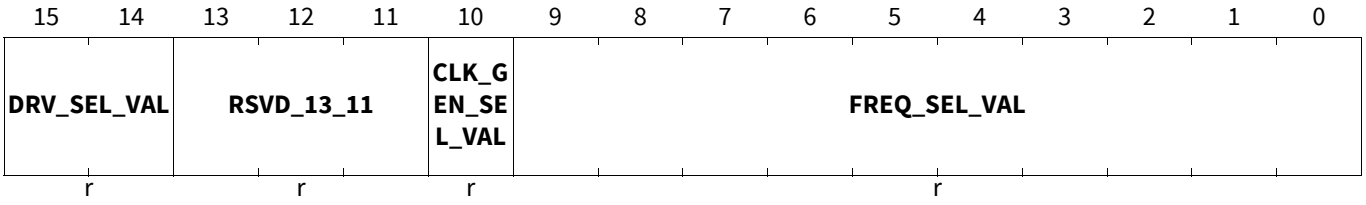
Field	Bits	Type	Description
CLK_GEN_SEL_VAL	10	r	**Clock Generator Select Value** This bit is effective when the Host Controller supports a programmable clock generator. 0_B Host Controller Ver2.0 Compatible Clock Generator 1_B Programmable Clock Generator
RSVD_13_11	13:11	r	**Reserved** These bits are reserved. They will always return 0.
DRV_SEL_VAL	15:14	r	**Driver Strength Select Value** Driver strength is supported by 1.8V signaling bus speed modes. This field is meaningless for 3.3V signaling. 00_B Driver Type B is selected 01_B Driver Type A is selected 10_B Driver Type C is selected 11_B Driver Type D is selected

Preset Value for Default Speed

This register defines Preset Value for Default Speed mode in SD mode. Writing this register has no effect.

PRESET_DS

Preset Value for Default Speed **(062_H)** **Application Reset Value: 0001_H**



Field	Bits	Type	Description
FREQ_SEL_VAL	9:0	r	**SDCLK/RCLK Frequency Select Value** 10-bit preset value to be set in SDCLK/RCLK Frequency Select field of the Clock Control register described by a Host System.
CLK_GEN_SEL_VAL	10	r	**Clock Generator Select Value** This bit is effective when Host Controller supports programmable clock generator. 0_B Host Controller Ver2.0 Compatible Clock Generator 1_B Programmable Clock Generator
RSVD_13_11	13:11	r	**Reserved** These bits are reserved. They will always return 0.

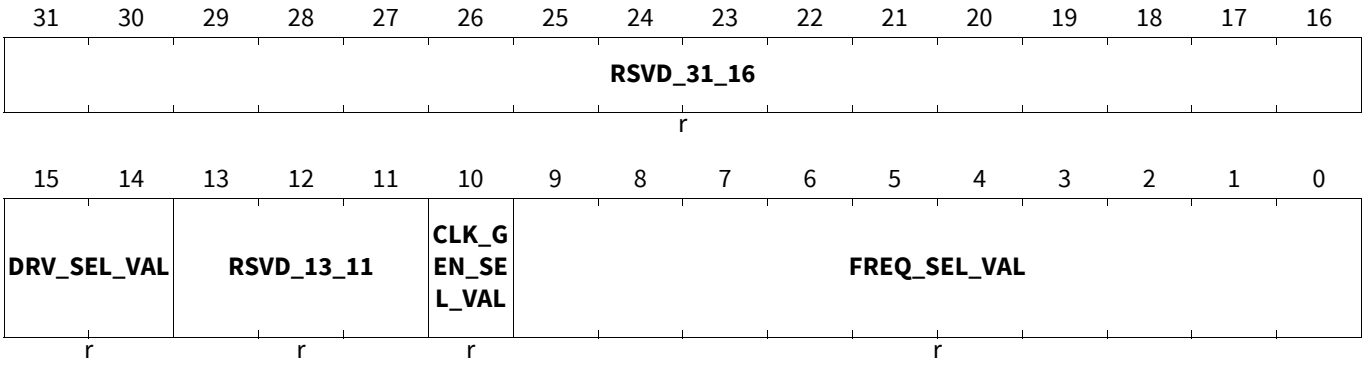
Field	Bits	Type	Description
DRV_SEL_VAL	15:14	r	**Driver Strength Select Value** Driver strength is supported by 1.8 V signaling bus speed modes. This field is meaningless for 3.3 V signaling. **Value After Reset:** 0_H 00_B Driver Type B is selected 01_B Driver Type A is selected 10_B Driver Type C is selected 11_B Driver Type D is selected

Preset Value for High Speed

This register defines Preset Value for High Speed mode in SD mode. Writing this register has no effect.

PRESET_HS

Preset Value for High Speed **(064_H)** **Application Reset Value: 0000 0000$_H$**



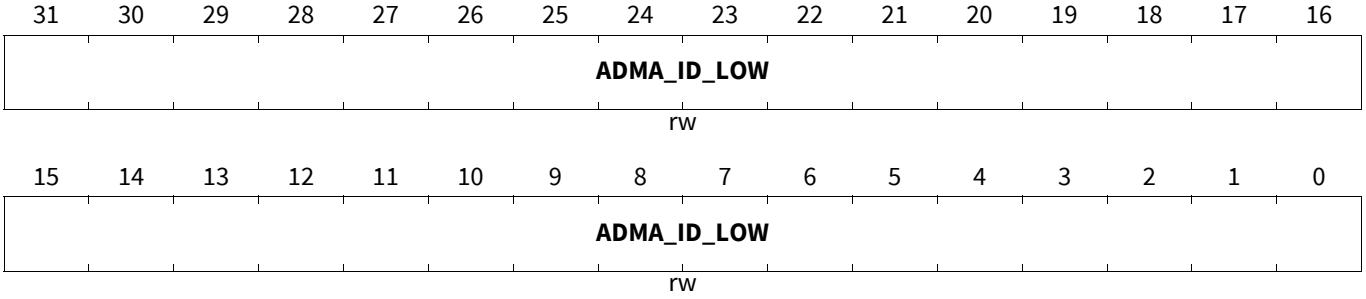
Field	Bits	Type	Description
FREQ_SEL_VAL	9:0	r	**SDCLK/RCLK Frequency Select Value** 10-bit preset value to be set in SDCLK/RCLK Frequency Select field of the Clock Control register described by a Host System.
CLK_GEN_SEL_VAL	10	r	**Clock Generator Select Value** This bit is effective when Host Controller supports programmable clock generator. 0_B Host Controller Ver2.0 Compatible Clock Generator 1_B Programmable Clock Generator
RSVD_13_11	13:11	r	**Reserved** These bits are reserved. They will always return 0.
DRV_SEL_VAL	15:14	r	**Driver Strength Select Value** Driver strength is supported by 1.8 V signaling bus speed modes. This field is meaningless for 3.3 V signaling. 00_B Driver Type B is selected 01_B Driver Type A is selected 10_B Driver Type C is selected 11_B Driver Type D is selected
RSVD_31_16	31:16	r	**Reserved** These bits are reserved. They will always return 0.

ADMA3 Integrated Descriptor Address Register - Low

This register holds the lower 32-bit Integrated Descriptor address.This register is applicable for an SD/eMMC mode.

ADMA_ID_LOW

ADMA3 Integrated Descriptor Address Register - Low(078_H) **Application Reset Value: $0000\ 0000_H$**



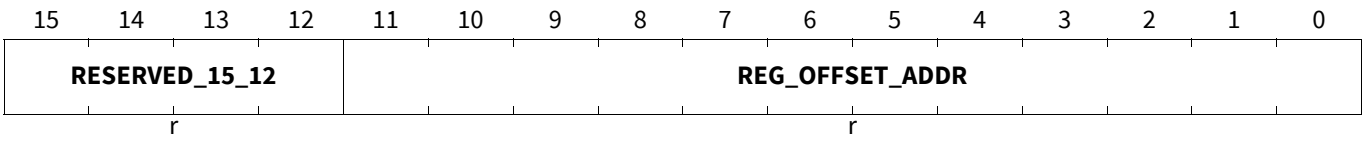
Field	Bits	Type	Description
ADMA_ID_LOW	31:0	rw	**ADMA Integrated Descriptor Address** This bit indicates the lower 32-bit of the ADMA Integrated Descriptor address. The start address of Integrated Descriptor is set to this register. The ADMA3 fetches one Descriptor Address and increments this field to indicate the next Descriptor address. **Volatile:** true

Pointer for Vendor Specific Area 1

This register used as a pointer for the Vendor Specific Area 1. Writing this register has no effect.

P_VENDOR_SPECIFIC_AREA

Pointer for Vendor Specific Area 1 **($0E8_H$)** **Application Reset Value: 0180_H**

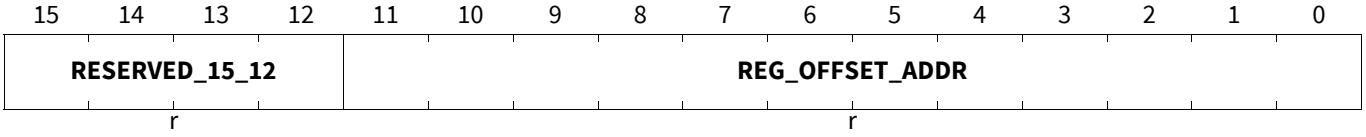


Field	Bits	Type	Description
REG_OFFSET_ADDR	11:0	r	**Base offset Address for Vendor Specific registers.**
RESERVED_15_12	15:12	r	**Reserved** These bits are reserved. They will always return 0.

Pointer for Vendor Specific Area 2

This register is used as a pointer for the Vendor Specific Area 2. Writing this register has no effect.

P_VENDOR2_SPECIFIC_AREA
Pointer for Vendor Specific Area 2 **($0EA_H$)** **Application Reset Value: 0300_H**

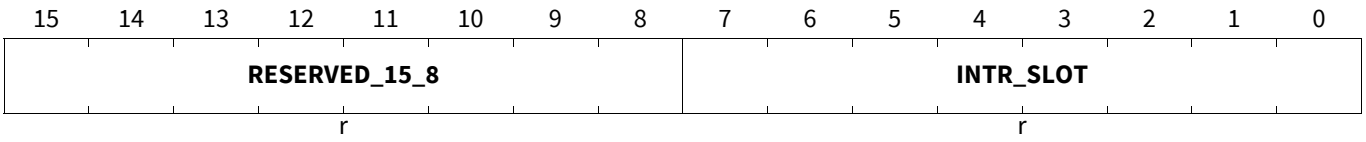


Field	Bits	Type	Description
REG_OFFSET_ADDR	11:0	r	**Base offset Address for Command Queuing registers.**
RESERVED_15_12	15:12	r	**Reserved** These bits are reserved. They will always return 0.

Slot Interrupt Status Register

This register indicate the Interrupt status of each slot. Writing this register has no effect.

SLOT_INTR_STATUS
Slot Interrupt Status Register **($0FC_H$)** **Application Reset Value: 0000_H**



Field	Bits	Type	Description
INTR_SLOT	7:0	r	**Interrupt signal for each Slot** These status bits indicate the logical OR of Interrupt signal and Wakeup signal for each slot. A maximum of 8 slots can be defined. If one interrupt signal is associated with multiple slots, the Host Driver can know which interrupt is generated by reading these status bits. By a power on reset or by setting Software Reset For All, the interrupt signal are de-asserted and this status reads 00h. **Volatile:** true 01_H Bit 0 - Slot 1 02_H Bit 1 - Slot 2 04_H Bit 2 - Slot 3 08_H Bit 3 - Slot 4 10_H Bit 4 - Slot 5 20_H Bit 5 - Slot 6 40_H Bit 6 - Slot 7 80_H Bit 7 - Slot 8
RESERVED_15_8	15:8	r	**Reserved** These bits are reserved. They will always return 0. **Volatile:** true

Host Controller Version

This register is used to indicate the Host Controller Version number. Writing this register has no effect.

HOST_CNTRL_VERS
Host Controller Version **($0FE_H$)** **Application Reset Value: 1005_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VENDOR_VERSION_NUM								SPEC_VERSION_NUM							
r								r							

Field	Bits	Type	Description
SPEC_VERSION_NUM	7:0	r	**Specification Version Number** This field indicates Host controller specification version. The upper and lower 4-bits indicate the version. 00_H SD Host Controller Specification Version 1.00 01_H SD Host Controller Specification Version 2.00 02_H SD Host Controller Specification Version 3.00 03_H SD Host Controller Specification Version 4.00 04_H SD Host Controller Specification Version 4.10 05_H SD Host Controller Specification Version 4.20 **others**, Reserved
VENDOR_VERSION_NUM	15:8	r	**Vendor Version Number** This field is reserved for the vendor version number. Host Driver should not use this status.

MSHC version

This register reflects the current release number of DWC_mshc/DWC_mshc_lite.

MSHC_VER_ID
MSHC version **(180_H)** **Application Reset Value: $3135\ 302A_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSHC_VER_ID															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSHC_VER_ID															
r															

Field	Bits	Type	Description
MSHC_VER_ID	31:0	r	**Current release number** This field indicates the Synopsys DesignWare Cores DWC_mshc/DWC_mshc_lite current release number that is read by an application. Current release is 1.50a which is represented in ASCII as 0x313530. Lower 8 bits read from this register can be ignored by the application. Application reading this register in conjunction with MSHC_VER_TYPE_R will gather details of the current release.

MSHC version type

This register reflects the current release type of DWC_mshc/DWC_mshc_lite.

MSHC_VER_TYPE
MSHC version type **(184_H)** **Application Reset Value: $6C70\ 3032_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSHC_VER_TYPE															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSHC_VER_TYPE															
r															

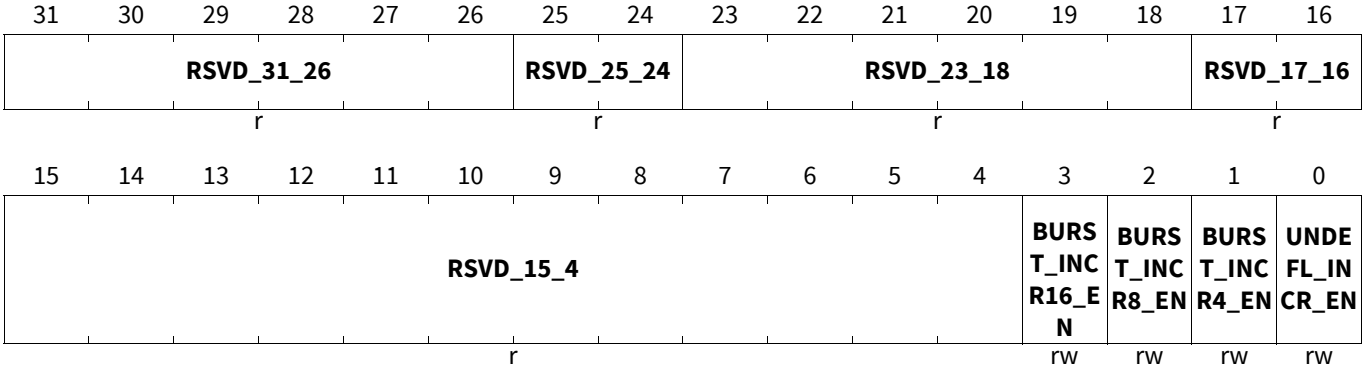
Field	Bits	Type	Description
MSHC_VER_TYPE	31:0	r	**Current release type** This field indicates the Synopsys DesignWare Cores DWC_mshc/DWC_mshc_lite current release type that is read by an application. Current release is of type "ga", which is represented in ASCII as 0x6761. Lower 16 bits read from this register can be ignored by the application. Application reading this register in conjunction with MSHC_VER_ID_R will gather details of the current release.

DMA burst control register

This register is used to control the Master bus interface.

MBIU_CTRL

DMA burst control register **(190_H)** **Application Reset Value: $0303\ 0006_H$**



Field	Bits	Type	Description
UNDEFL_INCR_EN	0	rw	**Controls the generation of undefined length INCR transfer on master interface** Must be disabled for usage of INCR4 and INCR8 bursts. 0_B INCR type bursts are not preferrend on AHB Master interface 1_B INCR type bursts are preferrend on AHB Master interface
BURST_INCR4_EN	1	rw	**Controls generation of INCR4 transfers on master interface** 0_B INCR bursts of length 4 are not generated on the master interface 1_B INCR bursts of length 4 can be generated on the master interface
BURST_INCR8_EN	2	rw	**Controls generation of INCR8 transfers on master interface** 0_B INCR bursts of length 8 are not generated on the master interface 1_B INCR bursts of length 8 can be generated on the master interface
BURST_INCR16_EN	3	rw	**Controls generation of INCR16 transfers on master interface** Not supported. Must be disabled. 0_B INCR bursts of length 16 are not generated on the master interface 1_B INCR bursts of length 16 can be generated on the master interface
RSVD_15_4	15:4	r	**Reserved - RSVD_7_4** These bits are reserved. They will always return 0.
RSVD_17_16	17:16	r	**Reserved** Write '1' to read '1'.
RSVD_23_18	23:18	r	**Reserved - RSVD_31_8** These bits are reserved. They will always return 0.
RSVD_25_24	25:24	r	**Reserved - RSVD_7_4** Write '1' to read '1'.
RSVD_31_26	31:26	r	**Reserved - RSVD_31_8** These bits are reserved. They will always return 0.

eMMC Control register

This register is used to control the eMMC operation.

EMMC_CTRL
eMMC Control register **(1AC$_H$)** **Application Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD						CQE_ALGO_SEL	ENH_STROBE_ENABLE	RSVD_7_2						DISABLE_DATA_CRC_CHK	CARD_IS_EMMC
r						rw	rw	r						rw	rw

Field	Bits	Type	Description
CARD_IS_EMMC	0	rw	**eMMC Card present** This bit indicates the type of card connected. An application program this bit based on the card connected to MSHC. 0_B Card connected to MSHC is non eMMC card 1_B Card connected to MSHC is eMMC card
DISABLE_DATA_CRC_CHK	1	rw	**Disable Data CRC Check** This bit controls masking of CRC16 error for Card Write in eMMC mode. This is useful in bus testing (CMD19) for an eMMC device. In bus testing, an eMMC card does not send CRC status for a block, which may generate CRC error. This CRC error can be masked using this bit during bus testing. 0_B DATA CRC check is enabled 1_B DATA CRC check is disabled
RSVD_7_2	7:2	r	**Reserved** These bits are reserved. They will always return 0.
ENH_STROBE_ENABLE	8	rw	**Enhanced Strobe Enable** This bit will be ignored.
CQE_ALGO_SEL	9	rw	**Scheduler algorithm selected for execution** This bit will be ignored.
RSVD	15:10	r	**Reserved** These bits are reserved. They will always return zero.

eMMC Boot Control register

This register is used to control the eMMC Boot operation.

BOOT_CTRL
eMMC Boot Control register **(1AE$_H$)** **Application Reset Value: 0000$_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT_TOUT_CNT				RSVD_11_9			BOOT_ACK_ENABLE	VALIDATE_BOOT	RSVD_6_1						MAN_BOOT_EN
rw				r			rw	w	r						rw

Field	Bits	Type	Description
MAN_BOOT_EN	0	rw	**Mandatory Boot Enable** This bit is used to initiate the mandatory boot operation. The application shall set this bit along with VALIDATE_BOOT bit. Writing 0 is ignored. The DWC_mshc clears this bit after the boot transfer is completed or terminated. **Testable:** readOnly 0_B Mandatory boot disable 1_B Mandatory boot enable
RSVD_6_1	6:1	r	**Reserved** These bits are reserved. They will always return 0.
VALIDATE_BOOT	7	w	**Validate Mandatory Boot Enable bit** This bit is used to validate the MAN_BOOT_EN bit. 0_B Ignore Mandatory boot Enable bit 1_B Validate Mandatory boot enable bit
BOOT_ACK_ENABLE	8	rw	**Boot Acknowledge Enable** When this bit set, DWC_mshc checks for boot acknowledge start pattern of 0-1-0 during boot operation. This bit is applicable for both mandatory and alternate boot mode. 0_B Boot Ack disable 1_B Boot Ack enable
RSVD_11_9	11:9	r	**Reserved** These bits are reserved. They will always return 0.
BOOT_TOUT_CNT	15:12	rw	**Boot Ack Timeout Counter Value.** This value determines the interval by which boot ack timeout (50 ms) is detected when boot ack is expected during boot operation. 0_H TMCLK * 2^13 1_H TMCLK * 2^14 2_H TMCLK * 2^15 … E_H TMCLK * 2^27 F_H Reserved

Embedded Control register

This register is control embedded device. When host controller is connected to removable device this register is not used.

EMBEDDED_CTRL
Embedded Control register **(280_H)** **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD_31	BACK_END_PWR_CTRL							RSVD_23	INT_PIN_SEL			RSVD_19	CLK_PIN_SEL		
r	rw							r	rw			r	rw		

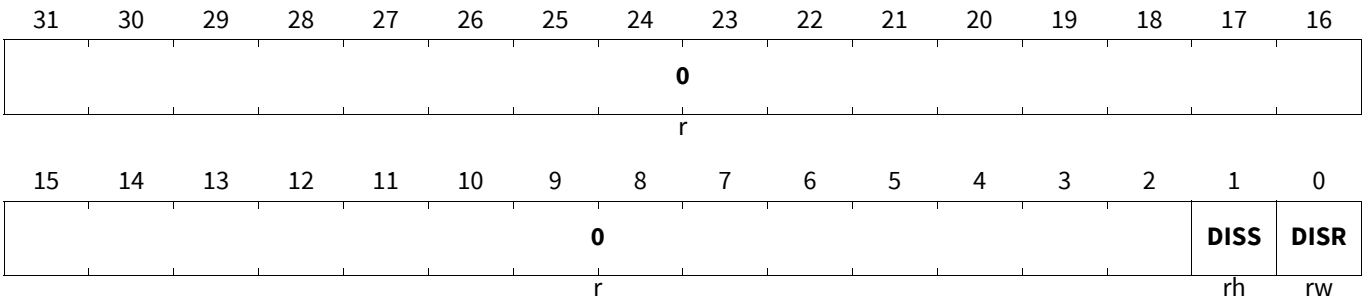
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSVD_15	BUS_WIDTH_PRESET							RSVD_7_6		NUM_INT_PIN		RSVD_3	NUM_CLK_PIN		
r	r							r		r		r	r		

Field	Bits	Type	Description
NUM_CLK_PIN	2:0	r	**Number of Clock Pins (SD Mode)** This field is not used and can be ignored.
RSVD_3	3	r	**Reserved** This bit is reserved. It will always return 0.
NUM_INT_PIN	5:4	r	**Number of Interrupt Input Pins** This field is not used and can be ignored.
RSVD_7_6	7:6	r	**Reserved** These bits are reserved. They will always return 0.
BUS_WIDTH_PRESET	14:8	r	**Bus Width Preset (SD Mode)** This field is not used and can be ignored.
RSVD_15	15	r	**Reserved** This bit is reserved. It will always return 0.
CLK_PIN_SEL	18:16	rw	**Clock Pin Select (SD Mode)** This field is not used and should not be changed.
RSVD_19	19	r	**Reserved** This bit is reserved. It will always return 0.
INT_PIN_SEL	22:20	rw	**Interrupt Pin Select** This field is not used and should not be changed.
RSVD_23	23	r	**Reserved** This bit is reserved. It will always return 0.
BACK_END_PWR_CTRL	30:24	rw	**Back-End Power Control (SD Mode)** This field is not used and should not be changed.
RSVD_31	31	r	**Reserved** This bit is reserved. It will always return 0.

Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality. Where a module kernel is connected to the CLC clock control interface, CLC controls the module clock for the module.

CLC
Clock Control Register **(300_H)** **Application Reset Value: 0000 0003$_H$**

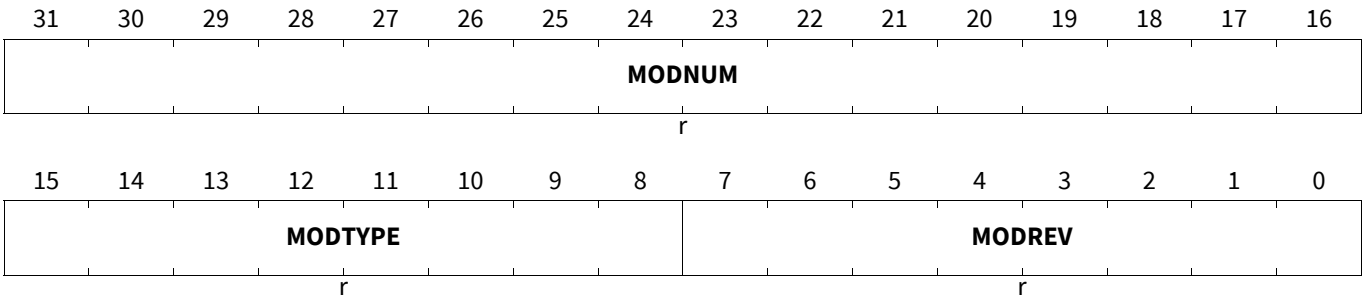


Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. The disable request is delayed internally until all pending transactions on the master and slave interface are completed.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module.
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Module Identification Register

The SDMMC Module Identification Register ID contains read-only information about the module version.

ID
Module Identification Register **(304_H)** **Application Reset Value: 00E9 C0XX$_H$**



Field	Bits	Type	Description
MODREV	7:0	r	**Module Revision Number** This bit field defines the module revision number. The value of a module revision starts with 01_H (first revision).

Field	Bits	Type	Description
MODTYPE	15:8	r	**Module Type** This bit field defines the module as a 32-bit module: $C0_H$
MODNUM	31:16	r	**Module Number Value** This bitfield defines the module identification number for the SDMMC Module: $00E9_H$

Access Enable Register 0

The Access Enable Register 0 controls write access[1)]for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, … , EN31 -> TAG ID 011111B.

ACCEN0

Access Enable Register 0 **($30C_H$)** **Application Reset Value: $FFFF\ FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

Access Enable Register 1

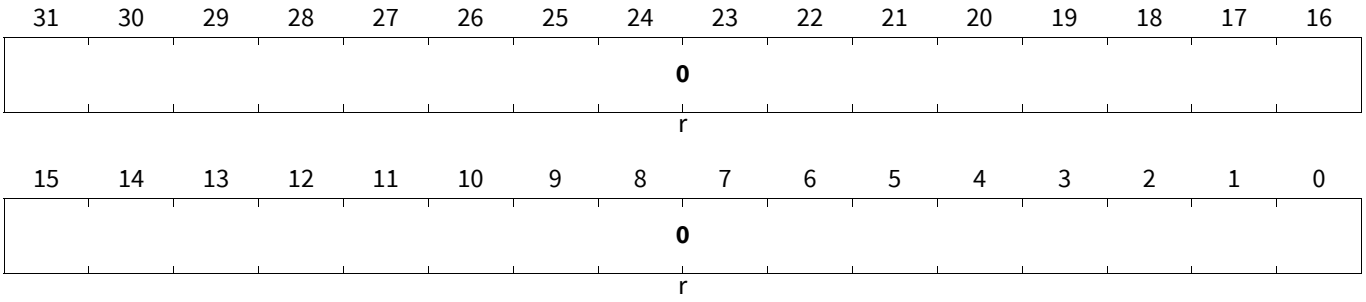
The Access Enable Register 1 controls write access1) for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in the device.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, … , EN31 -> TAG ID 111111B.

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

ACCEN1

Access Enable Register 1 **(310_H)** **Application Reset Value: 0000 0000$_H$**



Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 0

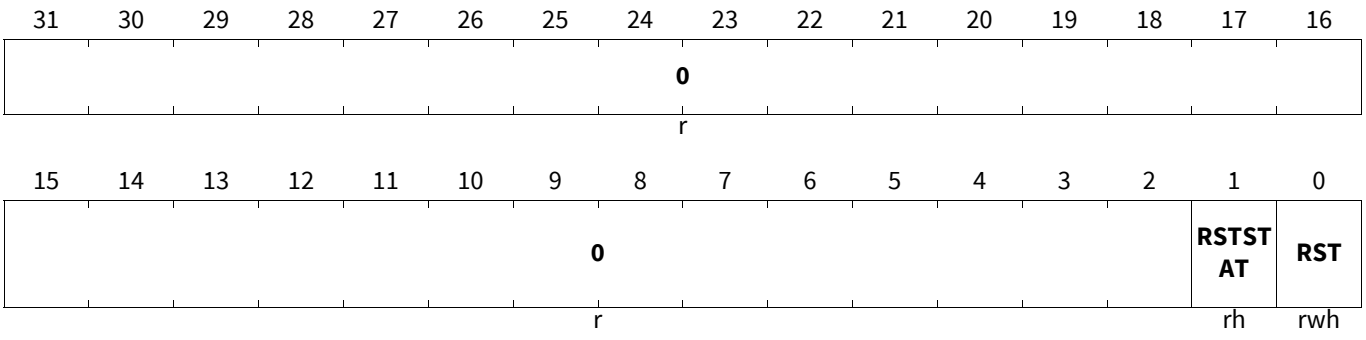
The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLR.CLR register bit.

Note: During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0

Kernel Reset Register 0 **(314_H)** **Application Reset Value: 0000 0000$_H$**

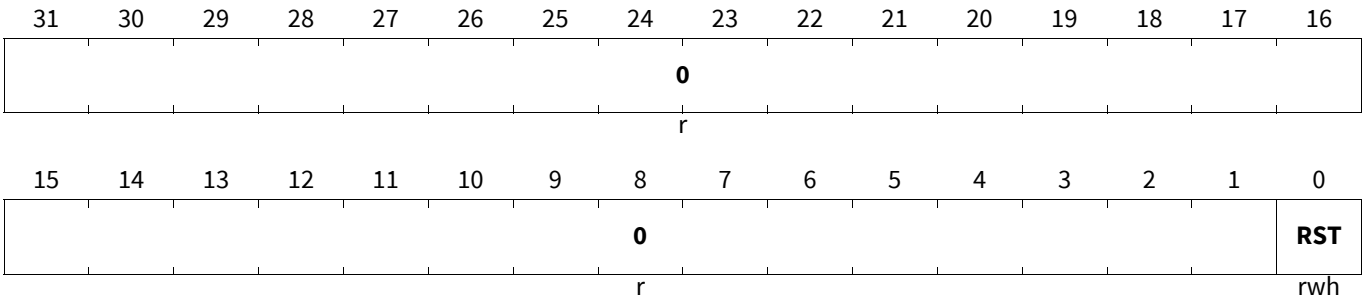


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set and both: master and slave interface are idle. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
Kernel Reset Register 1 **(318_H)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set and both: master and slave interface are idle. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested

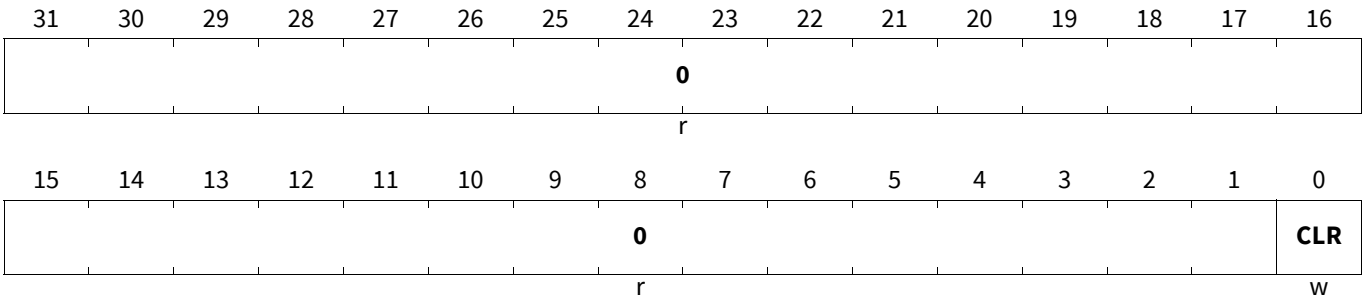
Field	Bits	Type	Description
0	31:1	r	**Reserved** Read as 0; should be written with 0.

Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR

Kernel Reset Status Clear Register **($31C_H$)** **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT
0	31:1	r	**Reserved** Read as 0; should be written with 0.

46.4 IO Interfaces

The table below lists all the interfaces of the SDMMC to other modules.

Table 541 List of SDMMC Interface Signals

Interface Signals	I/O	Description
CMD_IN	in	**command in** This signal reflects the read input from the command line pad.
CMD	out	**command out** This signal reflects the write output to the command line pad.
DAT0_IN	in	**read data in** This signal reflects the read input from the data line pads.
DAT1_IN		
DAT2_IN		
DAT3_IN		
DAT4_IN		
DAT5_IN		
DAT6_IN		
DAT7_IN		

Table 541 List of SDMMC Interface Signals (cont'd)

Interface Signals	I/O	Description
DAT0	out	**write data out** This signal reflects the write output to the data line pads.
DAT1		
DAT2		
DAT3		
DAT4		
DAT5		
DAT6		
DAT7		
CLK	out	**card clock** This signal reflects the output to the card clock pad.

46.5 Revision History

Table 542 Revision History

Reference	Change to Previous Version	Comment
V1.0.17		
Page 11	Footnote "a" added to **Figure 751**.	
Page 13	Footnotes "b" and "c" added to **Figure 753**.	
Page 88	Description of bit field "PRESET_VAL_ENABLE" enhanced and duplicy of value description removed.	
Page 64, **Page 88**, **Page 94**	List of reserved bit field values shortened in description of registers "PWR_CTRL", "HOST_CTRL2" and "CAPABILITIES2" (no functional change).	
V1.0.18		
Page 32	Replaced figure.	

47 Hardware Security Module (HSM)

The HSM is a separate processor subsystem dedicated for security tasks. It is connected as master and slave to the SPB bus.

For security reasons this module is described in a separate documentation. Please contact your Infineon representative for further information.

48 Input Output Monitor (IOM)

Description of the IOM (Input Output Monitor) block and its function within the system.

48.1 Feature List

- 16 Filter & Prescaler Channels, each connected to a specific pad (PIN), the output of which to serve as a monitor or reference signal.
- 16 Logic Analyzer Modules (LAMs), with each channel comprising a multiplexer to independently select monitor and reference signal inputs, with which to undertake the compare. A local event (i.e. internal to IOM) is generated when timing relationships between selected monitor and reference signals are violated. Configurable via register settings.
- 1 EXOR combiner, configurable (via register) to select a range of up to 8 GTM inputs in order to generate a combined signal to serve as a reference.
- Event Combiner Module (ECM), that takes the 16 local event signals (1 from each of the LAM modules) and generates a system event signal (connecting to system Safety Management Unit) from a single, combination or multiple local event(s). Configurable via register settings.
- System Peripheral Bus (SPB) interface for configuration and status register interaction.

48.2 Overview

The Input Output Monitor (IOM) module serves as a smart I/O comparison unit, observing and checking for the correct operation of system peripheral outputs that may serve and/or control externally-attached hardware, as well as the correct operation of the hardware itself, including any sensors whose signals may serve as an input to the monitoring function.

The monitoring function may be achieved in a number of configurable ways depending upon the application and the type of safety measure to be deployed. (Examples of such measures are included later-on within this chapter).

Monitor and reference signals (where needed) are taken from applicable system peripherals (i.e. Capture/Compare Units and General Purpose Timer Units) where they connect to internal pad logic (that feeds/controls I/O Ports), which may or may not be driven to the pads (configuration dependent). In this manner connectivity can be made to a signal that is to be driven externally, and to another signal that may serve as a reference, although may not be driven off-chip. A driven signal may also serve as a reference in order to compare against another that may be provided via a GPIO input, e.g. from a sensor external to the device.

Up to 16 monitoring points can be configured, with the capability to generate a system event that may be driven from one individual, several individual or multiple occurrences of a monitored condition, or a combination of several depending upon configuration.

48.3 Functional Description

48.3.1 Interfaces

The IOM features the following interfaces:

- System Peripheral Bus (SPB) - for the configuration of constituent parts of the IOM and the interrogation of associated status registers/bitfields.
- Multiple port logic/module connectivity, serving as reference or monitor signals.
- Safety Management Unit (SMU) - a single global event signal used to inform the SMU of a generated event within the IOM.

Note: The functionality of the IOM is independent from system debug, and no connectivity or mode of operation exists for this purpose.

48.3.2 Kernel Description

In addition to the Logic Analyzer Modules, Filter & Prescaler Blocks and Event Combiner Module, the IOM also features a System Peripheral Bus (SPB) interface for configuration and status register access. Note: All the configuration registers are protected by the register access protection mechanism (global definition).

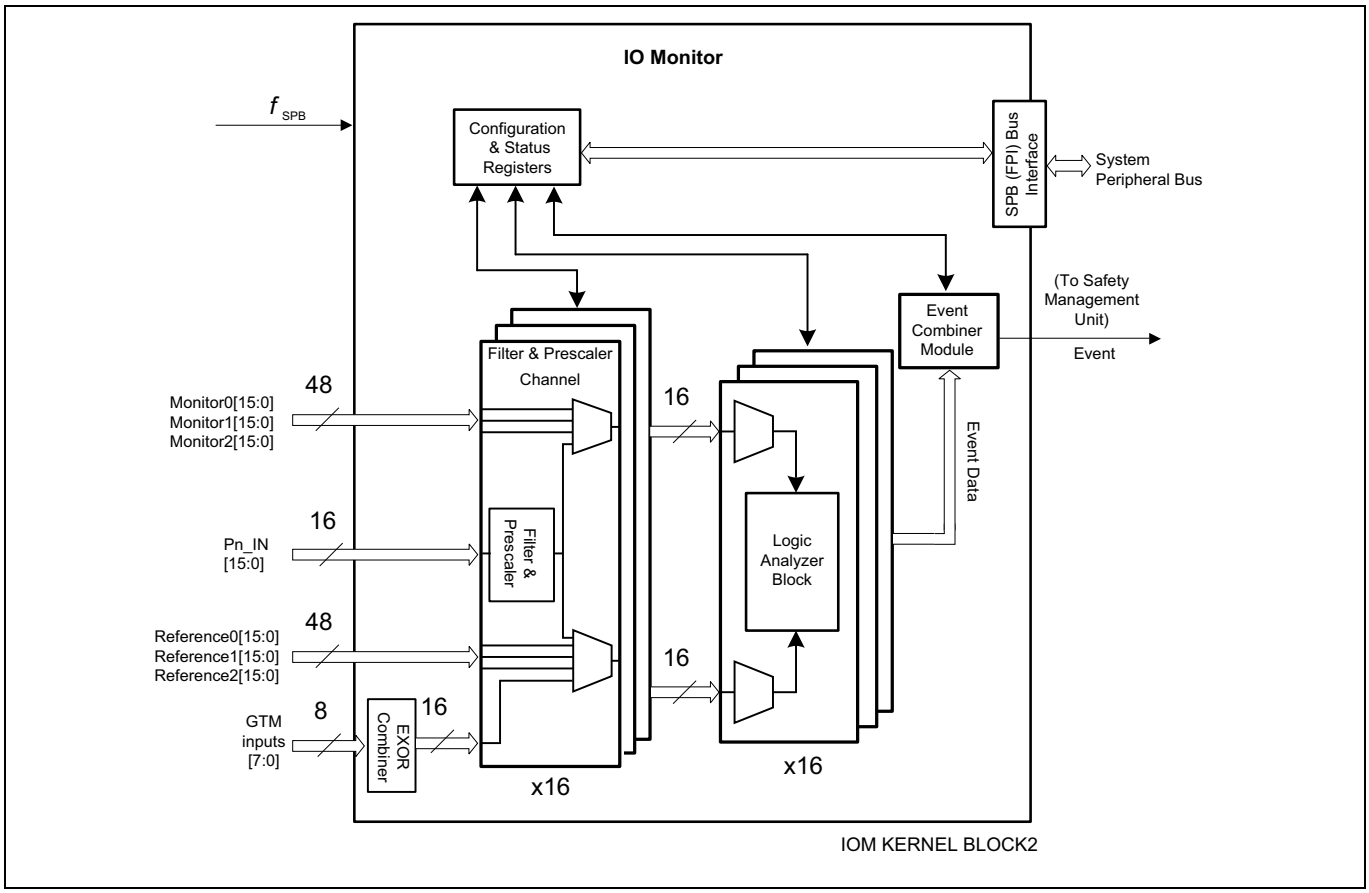


Figure 784 IO Monitor Block Diagram

The operational clock for the IOM (f_{IOM}) is derived form the SPB clock after applying the divider configurable in CLC.RMC bitfield.

48.3.3 Filter & Prescaler Channel (FPC) Description

The IOM instances 16 Filter & Prescaler Channels (FPC's) for the purposes of preconditioning and filtering the signals received from the pads (Pn_IN inputs) that may be used as a reference or monitor.

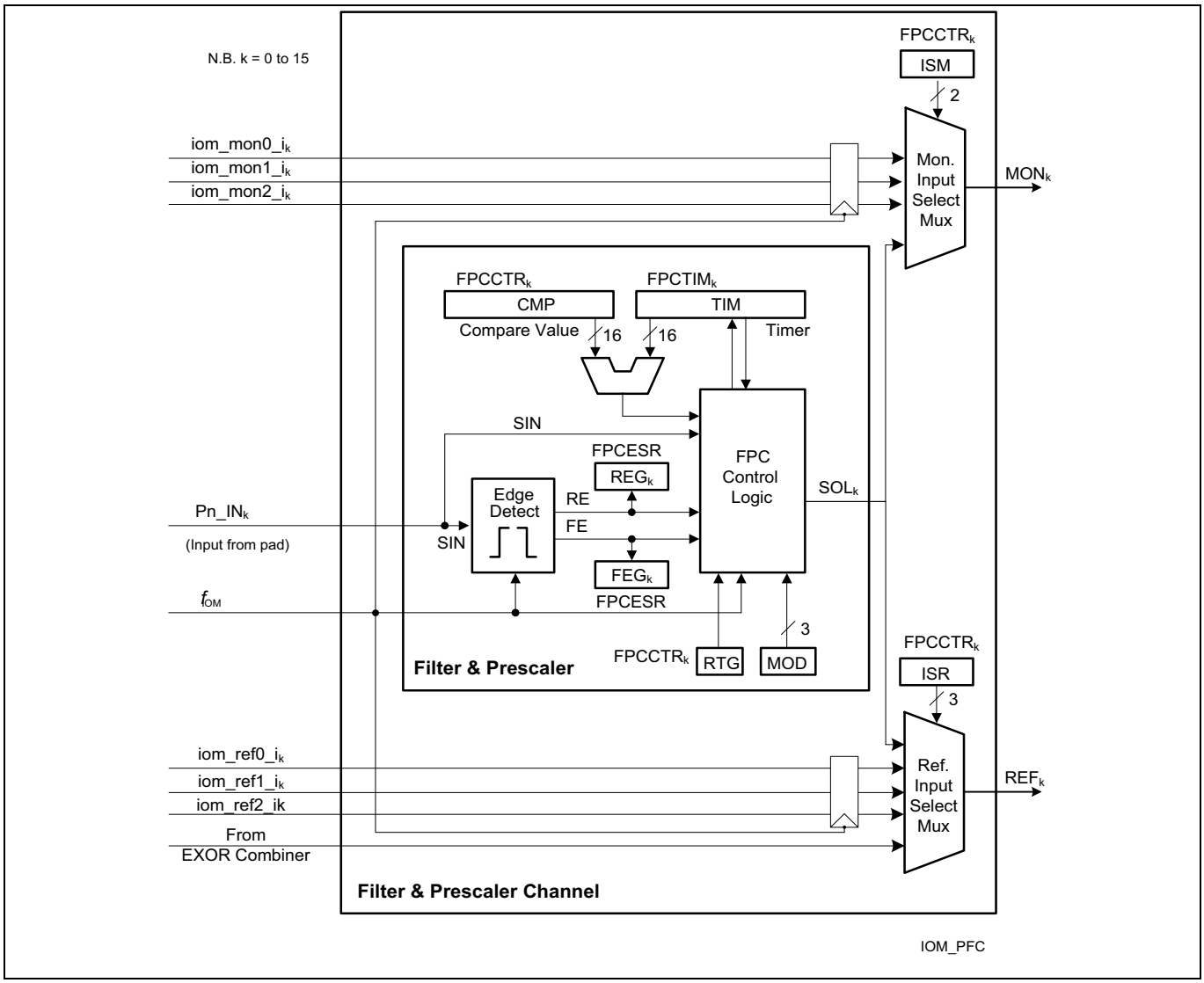


Figure 785 Monitor & Reference Selection logic incorporating Filter and Prescaler channel

As shown in **Figure 785**, each FPC is equipped with a signal input multiplexer, an edge detection circuitry, a 16-bit timer, a 16-bit compare register, a 16-bit comparator, and a FPC control circuitry. The edge detection circuitry detects respective edges for the prescaler modes and detects glitches in all other modes.

FPC Registers

The following registers are assigned to the Filter and Prescaler Channels FPCk (k = 0-15):

- FPCESR = Filter and Prescaler Channels Rising and Falling Edge Status Register.
- FPCCTRk = Filter and Prescaler Channel Control Register k
- FPCTIMk = Filter and Prescaler Channel Timer Register k

FPC Operating Modes

Each filter and prescaler channel can be individually configured to operate in one of the following operating modes:

- Delayed Debounce Filter Mode on both edges
- Immediate Debounce Filter Mode on both edges
- Rising edge: Immediate Debounce Filter Mode, falling edge: No filtering
- Rising edge: No filtering, falling edge: Immediate Debounce Filter Mode
- Rising edge: Delayed Debounce Filter Mode, falling edge: Immediate Debounce Filter Mode
- Rising edge: Immediate Debounce Filter Mode, falling edge: Delayed Debounce Filter Mode
- Prescaler Mode (triggered by edge detection circuitry on rising edge)
- Prescaler Mode (triggered by edge detection circuitry on falling edge)

The operation mode is selected by bit field FPCCTRk.MOD.

FPC Input Signal notes

The FPC input signal sampling frequency is f_{IOM}.

FPC Output Signal notes

A level output, representative signal SOLk, indicating the direction of the detected signal transition, that can be configured to connect to the Logic Analyzer Module.

Delayed Debounce Filter Mode

In Delayed Debounce Filter Mode, the signal input SIN is filtered from all signal transitions and glitches with a width smaller than f_{IOM} multiplied by the compare register value (FPCCTRk.CMP).

The input signal SIN is analyzed at the selected filter clock rate of f_{IOM}. If the state of the input sample differs from the current output signal value, the 16-bit timer is increased by one. When the timer register FPCTIMk is not in its idle state (0000_H) and the state of the input sample matches the current output signal value, the 16-bit timer is decremented by one (see **Figure 786**); if bit FPCCTRk.RTG is set, the timer will be set to idle state again (see **Figure 787**). A rising or falling edge, occurring on the signal input line SIN when the timer is greater than zero but less than the compare value, sets the corresponding glitch flag FPCESR.REG (on rising edge glitch) or FPCESR.FEG (on falling edge glitch). When the timer matches the 16-bit compare value stored in FPCCTRk.CMP (timer threshold), the level output signal line SOLk is inverted, and the timer is reset to 0000_H. The rising/falling edge glitch flags must be reset by software.

The filter is by-passed if the compare value FPCCTRk.CMP is programmed to zero (0000_H). In this case, the input signal is directly copied to the output signal.

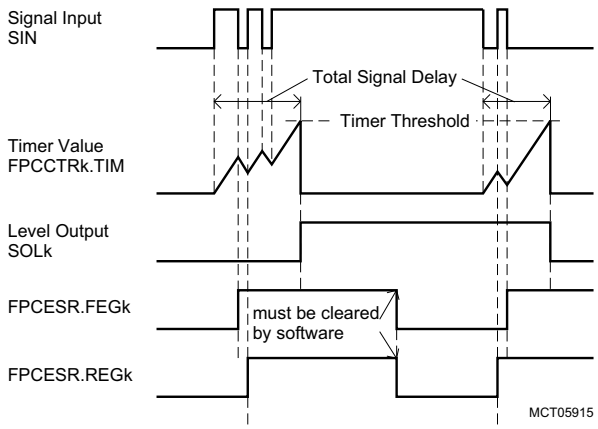


Figure 786 FPC Delayed Debounce Filter Algorithm with Timer Decrement

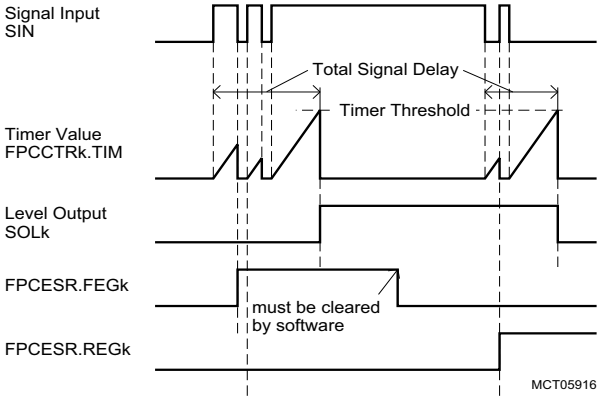


Figure 787 FPC Delayed Debounce Filter Algorithm with Timer Reset

The total signal delay from input to output depends on the programmed compare register value, the number of high-frequency pulses (glitches) during the filter operating time, and the timer behavior in case of a glitch (decrement or reset).

The FPC Delayed Debounce Filter Mode is selected by:

- FPCCTRk.MOD = 000_B

Immediate Debounce Filter Mode

In Immediate Debounce Filter Mode, the input signal is filtered from signal transitions and glitches arriving a programmable time after an input signal edge detection (see **Figure 788**).

The input signal (SIN) is sampled with f_{IOM} and the input signal SIN edge detection is also performed with f_{IOM}. The further analysis (e.g. filter timer increment, glitch detection) is also achieved with f_{IOM}.

As long as the timer is reset, the FPC control circuitry copies the sampled input value directly to the level output signal line SOLk. When a rising or falling edge occurs on SIN and the 16-bit compare value (FPCCTRk.CMP) is not zero, the timer is enabled to be increased by the selected clock and the copy mechanism is disabled. When the timer value FPCTIMk.TIM matches the compare value FPCCTRk.CMP, the timer is reset and the copy mechanism is enabled again. A rising or falling edge, occurring on SIN while the timer is greater than zero but less than the compare value, sets the corresponding glitch flag FPCESR.REG (on rising edge glitch) or FPCESR.FEG (on falling edge glitch). The rising/falling edge glitch flags must be reset by software.

The filter is by-passed if the compare value FPCCTRk.CMP is programmed to zero (0000_H). In this case, the input signal is directly copied to the output signal without any disable periods.

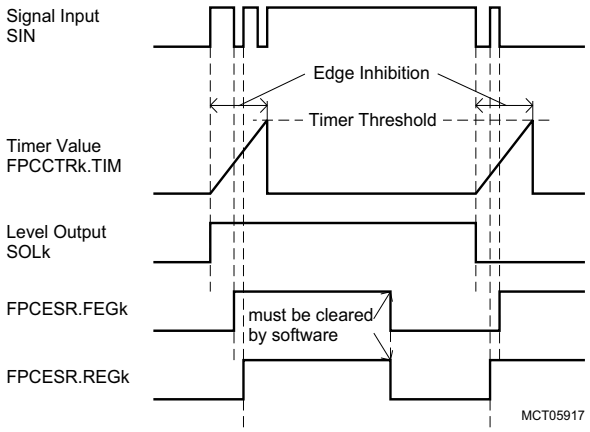


Figure 788 FPC Immediate Debounce Filter Algorithm on Both Edges

Note: During the last clock cycle of edge inhibition time (where timer value is equal to the compare value) an input signal glitch will be filtered but the corresponding glitch status flag in register FPCESR.REG/FPCESR.FEG is not set.

The Immediate Debounce Filter can be enabled for both edges. The signal output follows the signal input value immediately after the timer threshold of the filtered edge is reached (see **Figure 789**).

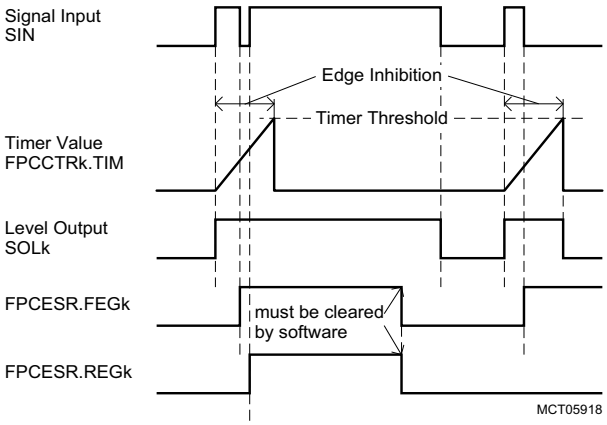


Figure 789 FPC Immediate Debounce Filter Algorithm on Rising Edge only

The FPC Immediate Debounce Filter Modes are selected by:

- FPCCTRk.MOD = 001_B: Immediate Debounce Filter Mode on both edges
- FPCCTRk.MOD = 010_B: Immediate Debounce Filter Mode on rising edge only, no filtering on falling edge.
- FPCCTRk.MOD = 011_B: Immediate Debounce Filter Mode on falling edge only, no filtering on rising edge.

Mixed Filter Modes

In the Mixed Filter Modes, one edge of a signal is filtered in the Delayed Debounce Mode, and the other edge is filtered in the Immediate Debounce Mode. The Debounce Mode is switched when the timer threshold is reached. Note that both filter modes use the same timer threshold in this case (see **Figure 790**, demonstrating Delayed Debounce Mode with Timer Decrement on Rising Edge and Immediate Debounce of on Falling Edge).

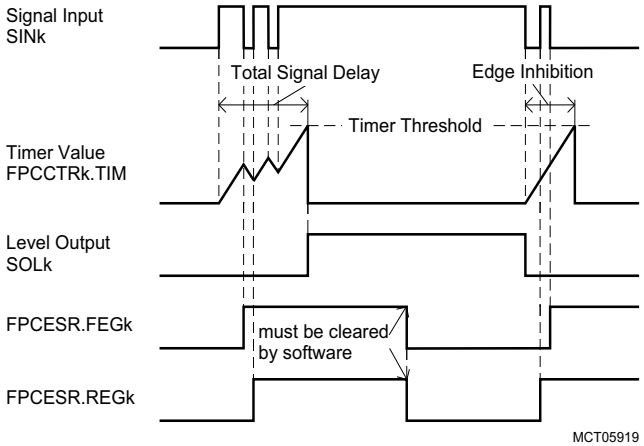


Figure 790 FPC Mixed Filter Algorithm

The FPC Mixed Filter Modes are selected by:

- FPCCTRk.MOD = 100_B: Delayed Debounce Filter Mode on rising edge, Immediate Debounce Filter Mode on falling edge
- FPCCTRk.MOD = 101_B: Immediate Debounce Filter Mode on rising edge, Delayed Debounce Filter Mode on falling edge

Prescaler Mode

In Prescaler Mode, the input signal is sampled and analyzed with f_{IOM}. The FPC control circuitry counts each rising (or falling) edge of the input signal. When the timer value matches the compare value:

- one IOM module clock pulse is generated at the level output signal SOLk
- the timer FPCTIMk.TIM is reset to 0000_H

Figure 791 shows a divide-by-6 operation using the FPC in Prescaler Mode with trigger on rising edge selected.

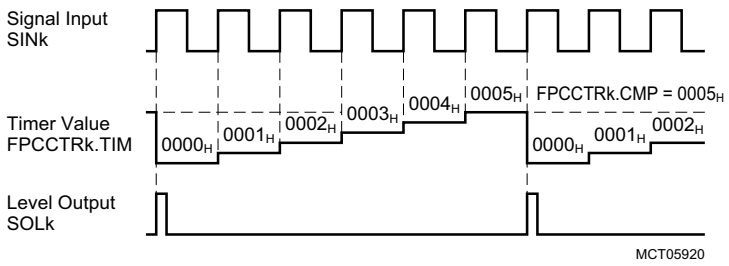


Figure 791 FPC Prescaler Mode

For a divide-by-n operation, the compare value FPCCTRk.CMP must be set to n - 1.

The FPC Prescaler Modes are selected by:

- FPCCTRk.MOD = 110_B: Prescaler Mode triggered by edge detection circuitry on rising edge
- FPCCTRk.MOD = 111_B: Prescaler Mode triggered by edge detection circuitry on falling edge

48.3.4 EXOR Combiner Description

The EXOR Combiner can be configured to accepts up to 8 external inputs from GTM module outputs.The output from which can then be used as a reference to the Logic Analyzer Module.

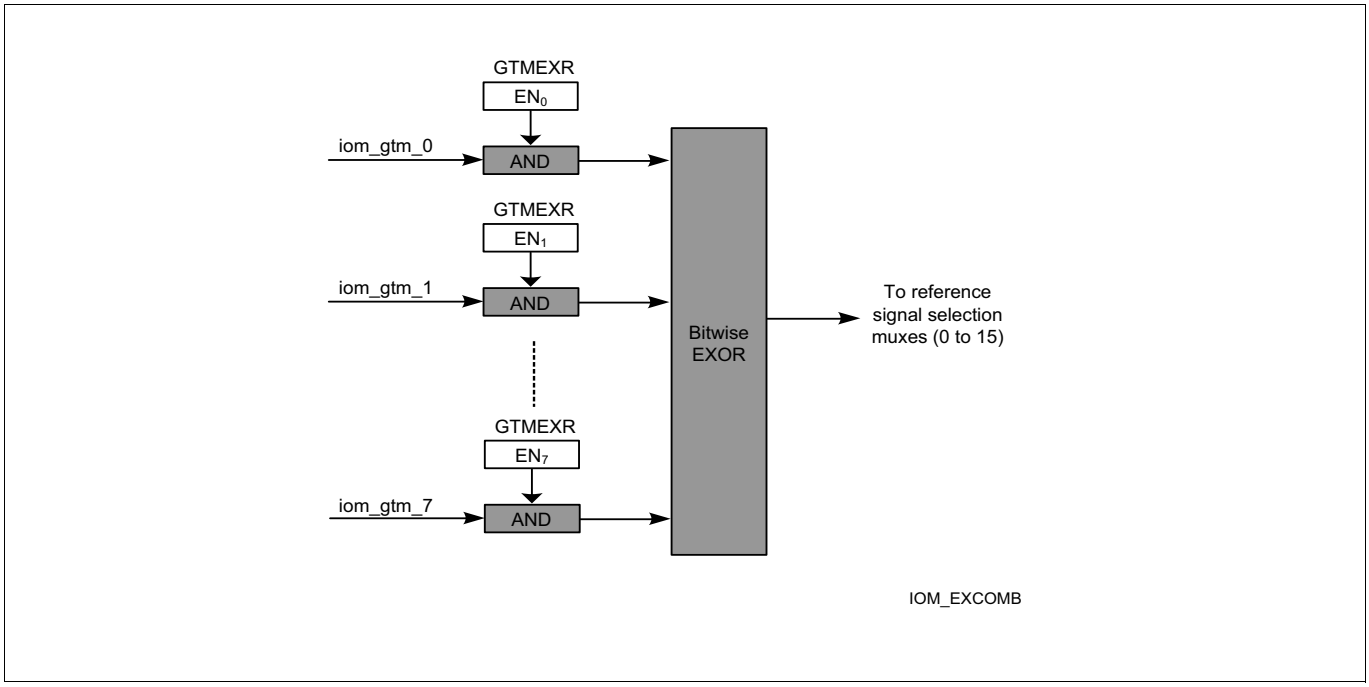


Figure 792 EXOR Combiner block diagram

EXOR Combiner Registers

The following registers are assigned to the block:

- GTMEXR = GTM Input EXOR Combiner Selection Configuration Register.

EXOR Combiner Input Signals

- 8 inputs from the GTM module(s).

EXOR Combiner Output Signals

- 1 output signal reflecting the combined logical EXOR function on the associated (and enabled input(s)).

48.3.5 Logic Analyzer Module (LAM) Description

The IOM instances 16 Logic Analyzer Modules (LAM's) for the purposes of monitor and reference signal comparison. Each block accepts one monitor and one reference level signal from the 16 monitor & 16 reference signals available, from the FPC/Input selection block. The behavior of each LAM is set by configuring the associated register settings. All sub-blocks of every LAM run at IOM frequency (f_{IOM}), for instance this is the frequency of the event window counter.

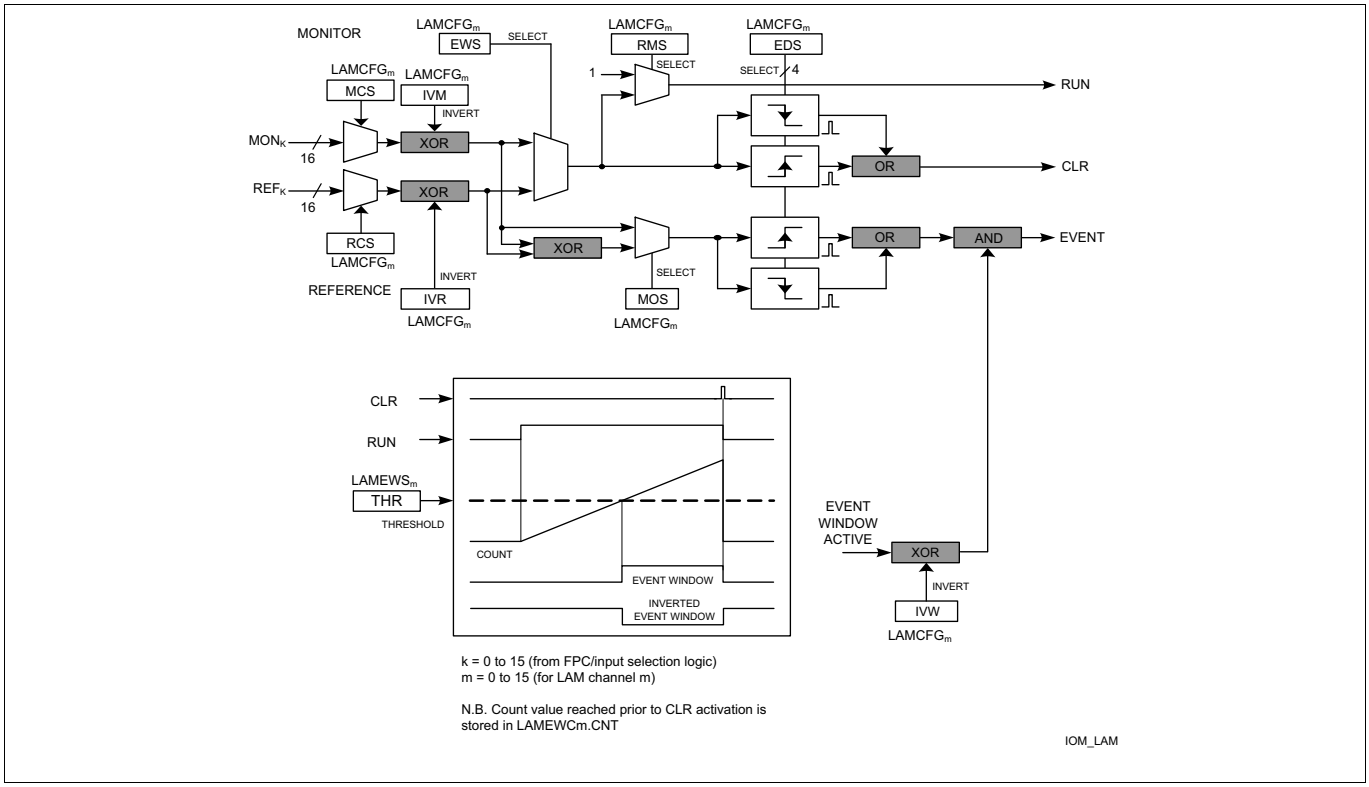


Figure 793 Logic Analyzer Module (LAM) block diagram

As shown in **Figure 793**, each Logic Analyzer Module features inputs for the Reference & Monitor points, fed from the requisite Filter & Prescaler block. Depending upon the Logic Analyzer Module configuration register (LAMCFG), and the event window settings register (LAMEWS), the block may be configured to monitor a particular type of signal behavior, input from sensors external to the device in order to monitor, govern and/or control external application behavior with particular regard to any given safety requirements.

If the 24-bit event window counter reaches its maximum value ($FFFFFF_H$), it will keep the maximum value until a clear condition arrives. This counter status (counter frozen at $FFFFFF_H$) will be notified by the LAMEWCm.CNTO flag.

LAM Registers

The following registers are assigned to the Logic Analyzer Module LAM m (m = 0-15):

- LAMCFGm = Logic Analyzer Module Configuration Register m
- LAMEWSm = Logic Analyzer Module Event Window Settings Register m
- LAMEWCm = Logic Analyzer Module Event Window Count Status Register m

After reset the counter for channel 'm' will be kept constant until a write operation to one of the 3 LAM registers: LAMCFGm, LAMEWSm or LAMEWCm.

Writing to one of the 3 LAM registers clears the LAMEWCm register and resets the internal counter for channel m.

LAM Input Signals

Two inputs exist for each LAM:

- Reference signal input, fed from the requisite Filter & Prescaler block.
- Monitor signal input, fed from the requisite Filter & Prescaler block.

Depending upon the required function, both inputs or just the Monitor input may be utilized.

LAM Output Signals

One output is provided by each LAM:

- Event trigger output, configured to be active when a monitored signal falls outside the intended timed and/or periodic behavior. When LAMCFGm.DISEV is set to 1, the LAM will suppress alarm outputs to the ECM. Except for sending alarms to the ECM, all other effects of an alarm condition being detected (for instance, setting LAMEWCm.CNT to the value of the counter) will still take place inside the LAM.

48.3.6 Event Combiner Module (ECM) Description

The IOM instances one Event Combiner Module, which takes the event outputs from each of the 16 Logic Analyzer Modules (LAM's). Some or all of these events can then be combined in a variety of configurable ways in order to generate a single system event.

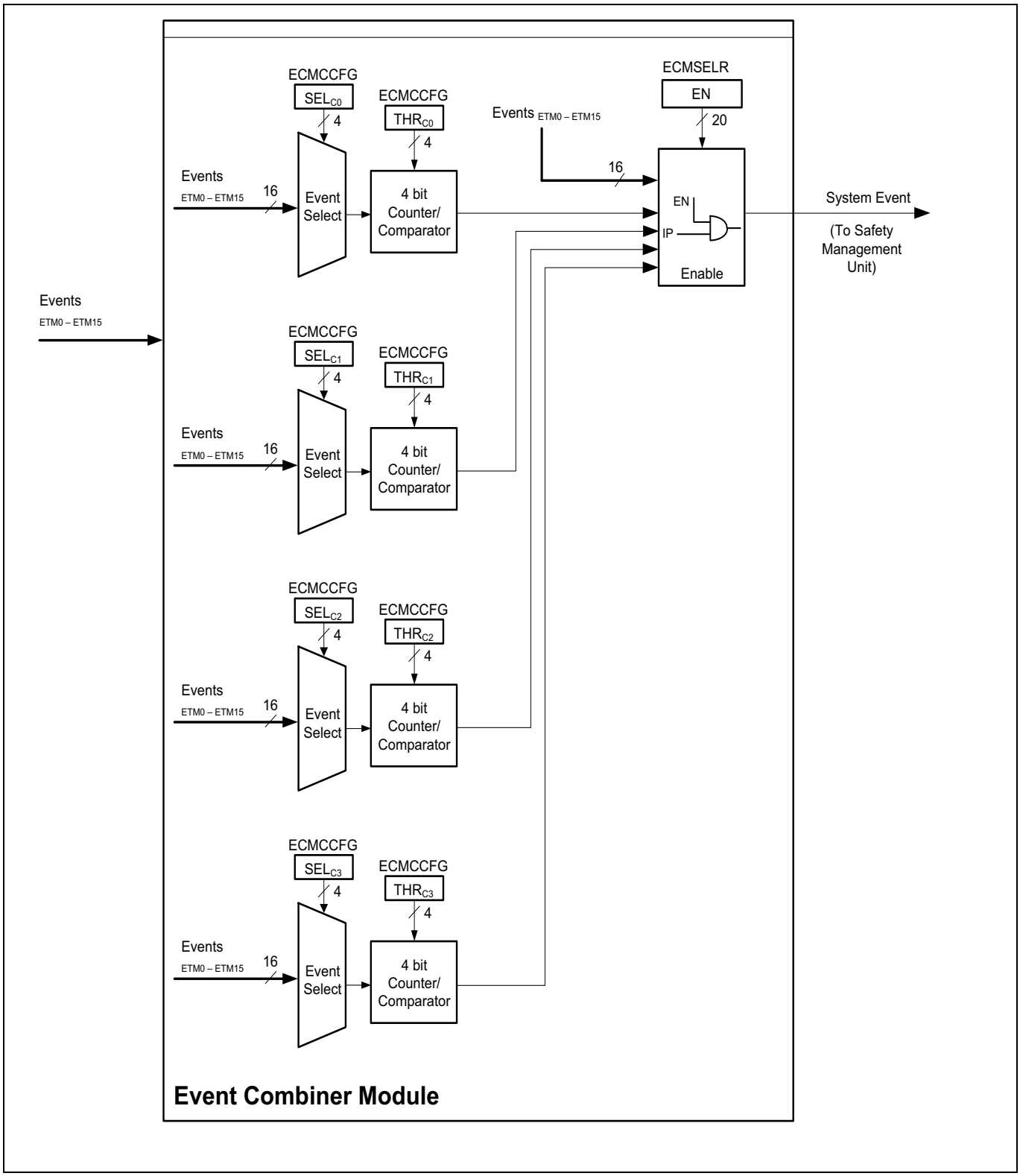


Figure 794 Event Combiner Module (ECM) block diagram

Four separate 4bit counters with configurable thresholds are also available to be assigned to count multiple occurring events from any of the 16 available LAM's. The output of these counter units (active once a count threshold has been met/surpassed) can also be included (via configuration) within the generation of the system event.

The ECM also includes two Event Trigger History (ETMETH0/1) registers, updated with each system event, used to record the status of the 16 LAM event outputs, thereby identifying which triggers were responsible.

All sub-blocks of the ECM run at IOM frequency (f_{IOM}). The IOM ensures that the System Event generated last as least one SPB clock cycle. (Note: the alarm output adapted to the f_{SPB} will be connected to the Safety Management Unit, SMU.EMM).

ECM Registers

The following registers are assigned to the ECM:

- ECMCCFG = ECM Counter Configuration Register (for the four 4bit local event counters).
- ECMSELR = ECM Global Event Selection Register (for the selection of local events, or multiples thereof, required to generate the global event signal).
- ECMETH0/1 =ECM Event Trigger History Register.

ECM Input Signals

- From Logic Analyzer Modules (LAM's)
  - Local event triggers from LAM blocks (0 to 15).

ECM Output Signals

- Global event trigger output, intended to be connected to the Safety Management Unit.

48.3.7 Configuration Sequence

In order to avoid undefined behavior due to the initialization order of the IOM submodules, the configuration sequence for an application that is using the IOM should be the following:

1st Step: Configure SMU reaction for the IOM as no alarm.

2nd Step: Enable the IOM and its clock by setting CLC.DISR to 0 and CLC.RMC to 1.

3rd Step: Configure the EXOR Combiner if required.

4th Step: If needed, configure each FPC:

- Configure each channel "m" (with "m" form 0 to 15) using the FPCCTRm register.
- Clear the existing rising and falling edge status flags by writing to the FPCESR register.

5th Step: Configure each LAM block "m" (with "m" form 0 to 15) :

- Set the desired event window count threshold by writing to LAMEWSm.THR.
- Configure the application specific settings for each LAM block in LAMCFGm register.

6th Step: Configure the ECM:

- Select the different events from the LAM to be combined in the ECMSELR register.
- Configure each of the four event counter submodules in ECMCCFG register.
- Write to ECMETH0 or ECMETH1 to clear all event trigger active flags.

7th Step: If required, configure the desired clock divider in CLC.RMC.

8th Step: Configure SMU reaction for the IOM to a specific alarm reaction.

48.3.8 Example Monitor/Safety Measures

For each channel the following measurements shall be possible:

- Duty cycle measurement
- Set-up and/or hold time measurement
- Delay window (min,max) with respect to rising or falling edge

The following waveforms provide more examples of typical signal monitoring functionality that could be deployed using the IOM. Some typical configuration settings are also provided on some of the examples.

48.3.8.1 Example 1 - Pulse or duty cycle too short

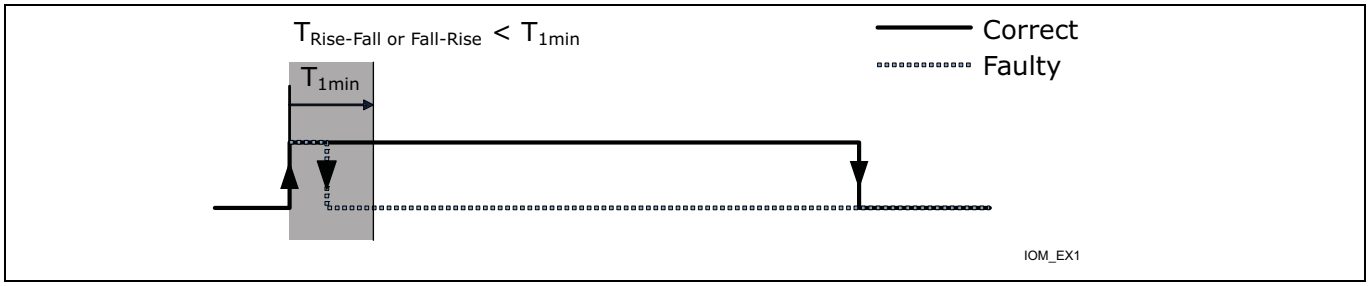


Figure 795 example 1 - pulse or duty cycle too short

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don’t care

LAMCFG.IVM: 0x0 ; don’t invert monitor signal (or invert for non-duty part of period)

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x1 ; select monitor signal for event window

LAMCFG.EDS: 0x9 ; CLR on positive edge, EVT on negative edge

LAMCFG.IVW: 0x1 ; invert window, capture events when the counter is below the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; select appropriate threshold (maximum pulse width forbidden. If pulse width is shorter or equal to this value then an event will be triggered)

48.3.8.2 Example 2 - Pulse or duty cycle too long

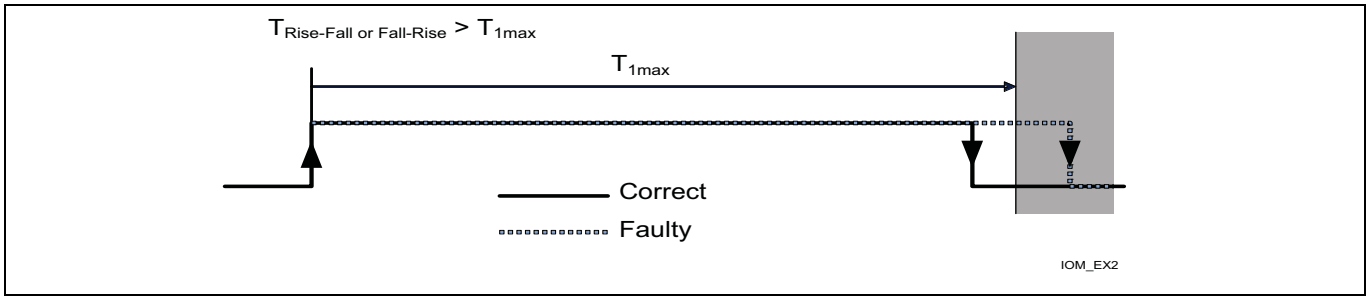


Figure 796 example 2 - pulse or duty cycle too long

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don’t care

LAMCFG.IVM: 0x0 ; don’t invert monitor signal (or invert for non-duty part of period)

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x1 ; select monitor signal for event window

LAMCFG.EDS: 0x9 ; CLR on positive edge, EVT on negative edge

LAMCFG.IVW: 0x0 ; do not invert window, capture events when the counter is equal or above the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; select appropriate threshold (maximum pulse width accepted. If pulse width is longer than this value then an event will be triggered)

48.3.8.3 Example 3 - Period too short

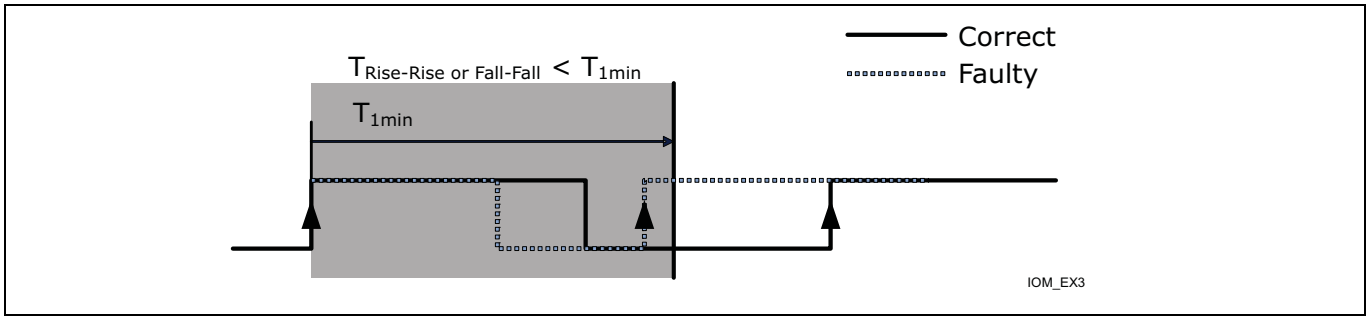


Figure 797 example 3 - period too short

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don’t care

LAMCFG.IVM: 0x0 ; don’t invert monitor signal

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x1 ; select monitor signal for event window

LAMCFG.EDS: 0x5 ; CLR on positive edge, EVT on positive edge

LAMCFG.IVW: 0x1 ; invert window, capture events when the counter is below the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; select appropriate threshold (maximum period length forbidden. If period is shorter or equal than this value then an event will be triggered)

48.3.8.4 Example 4 - Period too long

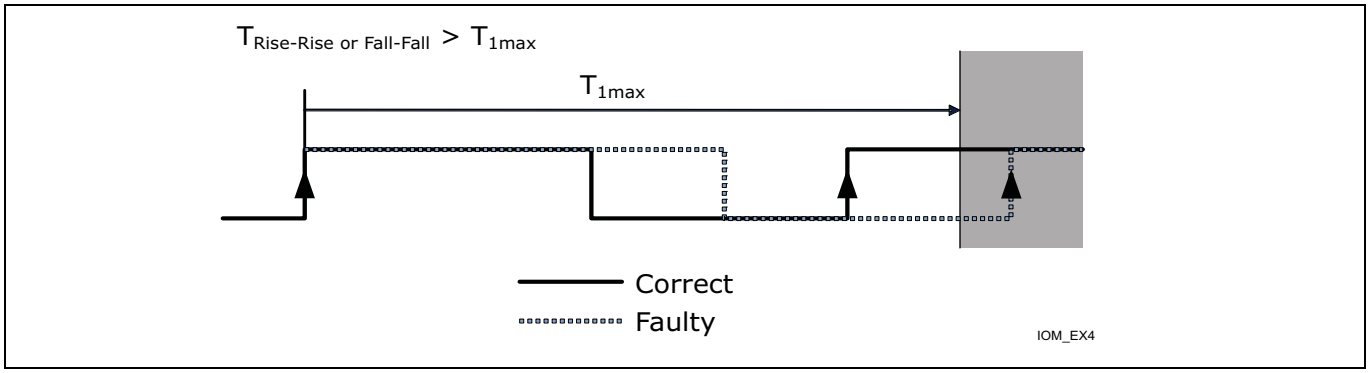


Figure 798 example 4 - period too long

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don’t care

LAMCFG.IVM: 0x0 ; don’t invert monitor signal

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x1 ; select monitor signal for event window

LAMCFG.EDS: 0x5 ; CLR on positive edge, EVT on positive edge

LAMCFG.IVW: 0x0 ; do not invert window, capture events when the counter is equal or above the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; select appropriate threshold (maximum period length required. If period is longer than this value then an event will be triggered)

48.3.8.5 Example 5 - Diagnosis of Command and Feedback - acceptable propagation window and/or signal consistency check

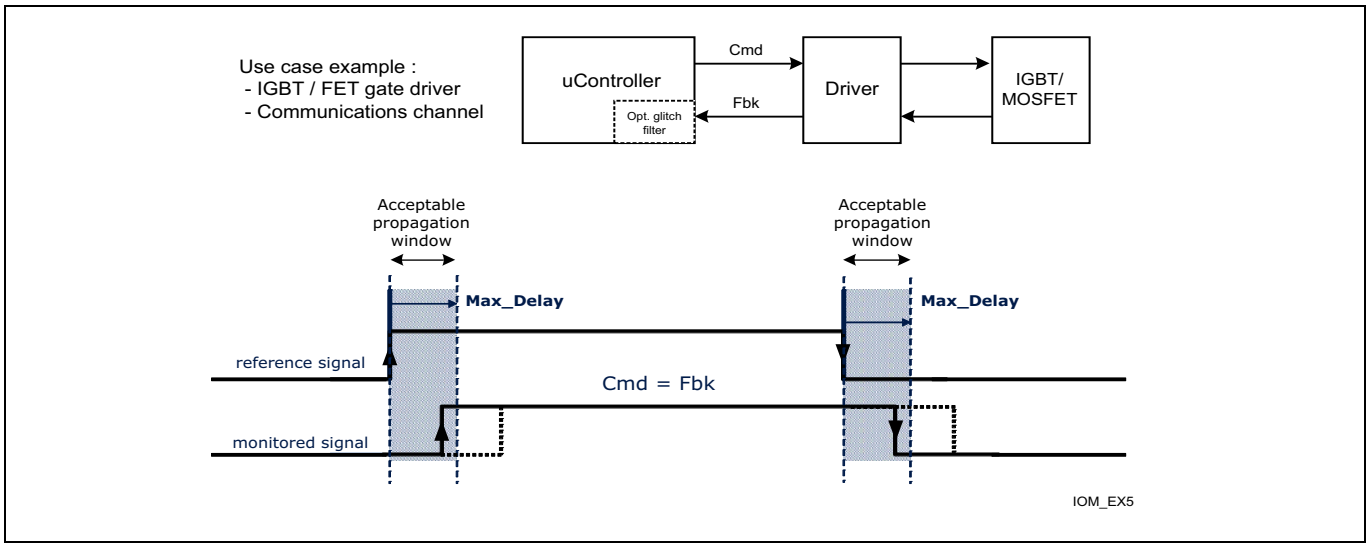


Figure 799 example 5 - Diagnosis of Command & Feedback

Using the driven signal (off-chip) as a reference, using a feedback signal (input to device) as the monitor. Checking that the monitored signal shows activity within an acceptable propagation window, and/or that the feedback signal shows the same behavior as the reference (Command = Feedback).

Example settings for LAM block registers:

LAMCFG.IVR: 0x0 ; don’t invert reference signal

LAMCFG.IVM: 0x0 ; don’t invert monitor signal

LAMCFG.MOS: 0x1 ; (MON xor REF) signal used for event generation

LAMCFG.RMS: 0x0 ; free-running

LAMCFG.EWS: 0x0 ; select reference signal for event window

LAMCFG.EDS: 0xB ; CLR on both edges of REF, EVT on falling edge (of xor’d signal)

LAMCFG.IVW: 0x0 ; do not invert window, capture events when the counter is equal or above the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; set to max delay allowed (if the delay between corresponding edges of reference and monitor signals is longer than this value, the event will be triggered)

Note: For proper diagnosis in example 5, the edges of the reference and monitor signals must arrive to the IOM in the following order: first leading edge of reference signal, second leading edge of monitor signal, third trailing edge of reference signal and fourth trailing edge of monitor signal.

48.3.8.6 Example 6 - Diagnosis of Set-up and Hold times

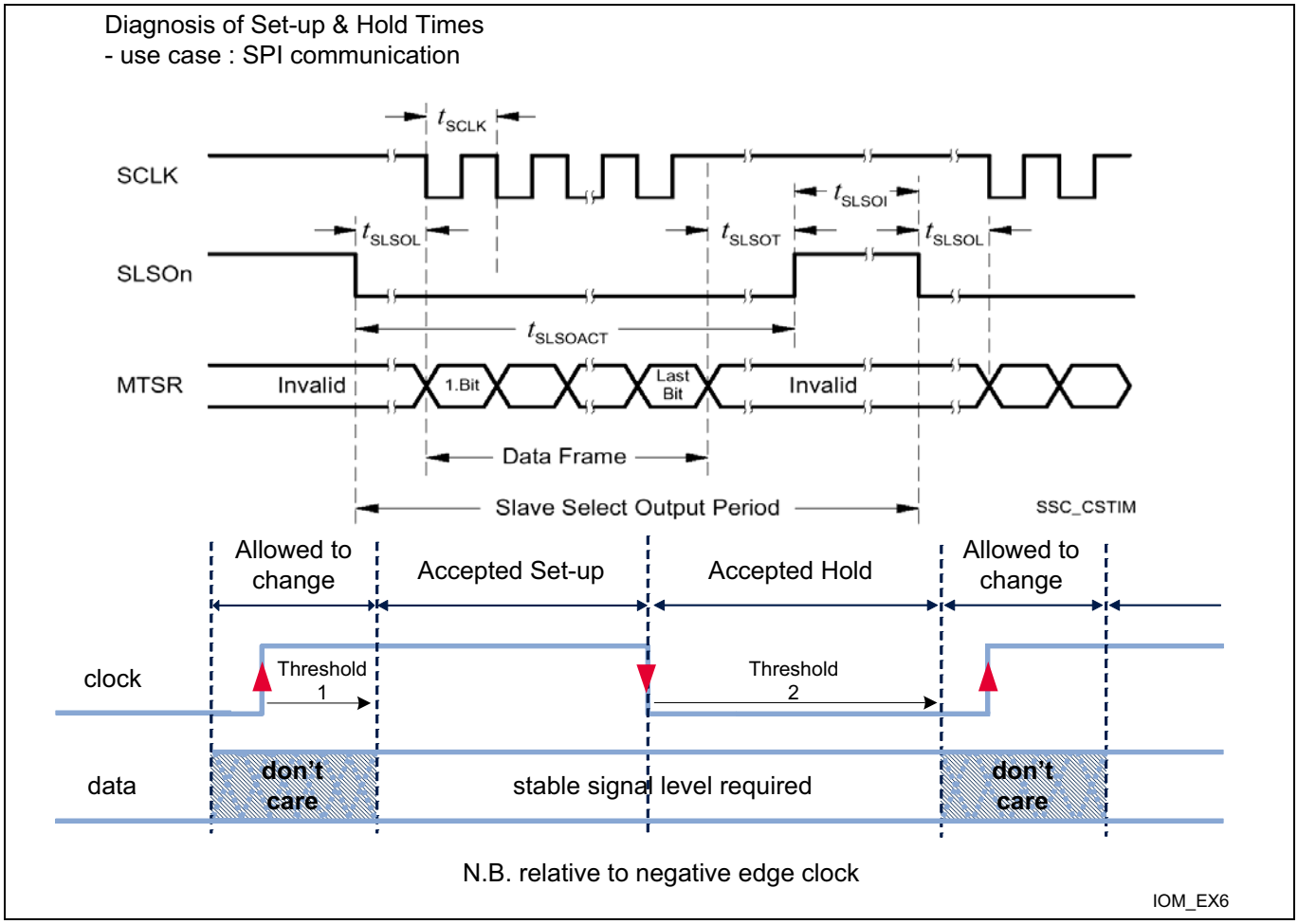


Figure 800 example 6 - Diagnosis of Set-up & Hold times

Ensuring that data remains stable within the Set-up and Hold windows, generating an event otherwise, and/or ensuring that data only changes within the specified windows.

Note: 2 LAM channels will be required for this purpose, one for set-up, one for hold.

Example settings for LAM block registers for Set-up

LAMCFG.IVR: 0x0 ; don't invert reference signal

LAMCFG.IVM: 0x0 ; don't invert monitor signal

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x1 ; run is gated

LAMCFG.EWS: 0x0 ; select reference signal for event window

LAMCFG.EDS: 0xF ; CLR on both edges, EVT on both edges

LAMCFG.IVW: 0x0 ; do not invert window, capture events when the counter is equal or above the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; Acceptable Set-up (ref Threshold 1 on waveforms shown, changes in monitor signal will be accepted for “THR” cycles after a raising edge in the reference signal. After that, a change in the monitor signal while the reference signal is “1”, will generate an alarm).

Example settings for LAM block registers for Hold

LAMCFG.IVR: 0x1 ; invert reference signal (use for gating)

LAMCFG.IVM: 0x0 ; don’t invert monitor signal

LAMCFG.MOS: 0x0 ; monitor signal used for event generation

LAMCFG.RMS: 0x1 ; event window is gated

LAMCFG.EWS: 0x0 ; select reference signal for event window

LAMCFG.EDS: 0xD ; CLR on positive edge (inverted neg. edge), EVT on both edges

LAMCFG.IVW: 0x1 ; invert window, capture events when the counter is below the threshold

LAMCFG.MCS/RCS ; select according to device configuration

LAMEWS.THR ; Acceptable Hold (ref Threshold 2 on waveforms shown, changes in monitor signal will generate an alarm if they occur inside the “THR” cycles after a falling edge in the reference signal).

48.4 Registers

This section describes the kernel registers of the IOM unit.

All read and write accesses to an undefined address location within the IOM address space will result in a bus error on the System Peripheral Bus (SPB).

Table 543 Register Overview - IOM (ascending Offset Address)

Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
CLC	IOM Clock Control Register	000_H	U,SV	SV,E,P	Application Reset	39
ID	IOM Identification Register	008_H	U,SV	BE	Application Reset	23
KRSTCLR	IOM Kernel Reset Status Clear Register	$01C_H$	U,SV	SV,E,P	Application Reset	42
KRST1	IOM Kernel Reset Register 1	020_H	U,SV	SV,E,P	Application Reset	42
KRST0	IOM Kernel Reset Register 0	024_H	U,SV	SV,E,P	Application Reset	41
ACCEN1	IOM Access Enable Register 1	028_H	U,SV	SV,SE	Application Reset	40
ACCEN0	IOM Access Enable Register 0	$02C_H$	U,SV	SV,SE	Application Reset	40
ECMCCFG	IOM Event Combiner Module Counter Configuration Register	030_H	U,SV	U,SV,P	Application Reset	33
ECMSELR	IOM Event Combiner Module Global Event Selection Register	034_H	U,SV	U,SV,P	Application Reset	35
ECMETH0	IOM Event Combiner Module Event Trigger History Register 0	038_H	U,SV	U,SV,P	Application Reset	36
ECMETH1	IOM Event Combiner Module Event Trigger History Register 1	$03C_H$	U,SV	U,SV,P	Application Reset	37
GTMEXR	IOM GTM Input EXOR Combiner Selection Register	040_H	U,SV	U,SV,P	Application Reset	27
FPCESR	IOM Filter and Prescaler Channels Rising & Falling Edge Status Register	078_H	U,SV	U,SV,P,32	Application Reset	24
FPCCTRk	IOM Filter and Prescaler Channel Control Register k	080_H+k* 4	U,SV	U,SV,P	Application Reset	24
FPCTIMk	IOM Filter and Prescaler Channel Timer Register k	$0C0_H$+k* 4	U,SV	U,SV,P	Application Reset	25

Table 543 Register Overview - IOM (ascending Offset Address) (cont'd)

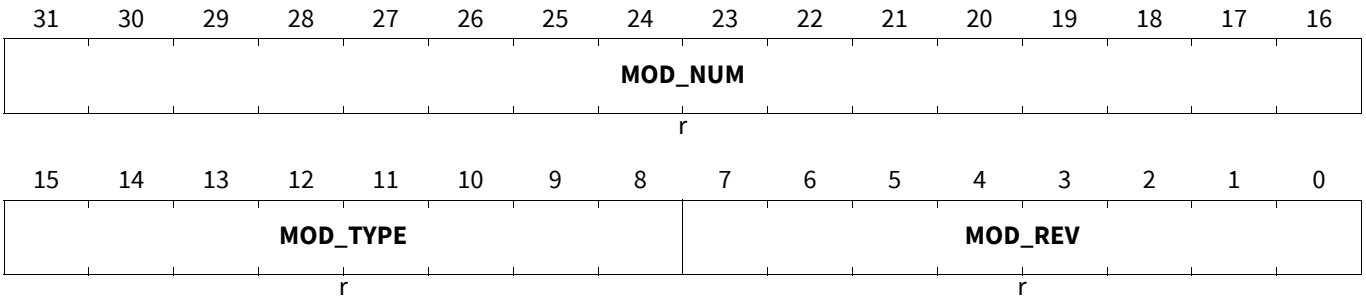
Short Name	Long Name	Offset Address	Access Mode		Reset	Page Number
			Read	Write		
LAMEWCm	IOM Logic Analyzer Module Event Window Count Status Register m	100_H+m*4	U,SV	U,SV,P	Application Reset	28
LAMCFGm	IOM Logic Analyzer Module Configuration Register m	180_H+m*4	U,SV	U,SV,P	Application Reset	28
LAMEWSm	IOM Logic Analyzer Module Event Window Configuration Register m	$1C0_H$+m*4	U,SV	U,SV,P	Application Reset	32

48.4.1 IOM Identification Register (IOM_ID)

IOM Identification Register

The IOM Identification Register ID contains read-only information about the module version.

ID
IOM Identification Register **(008_H)** **Application Reset Value: $00CC\ C002_H$**



Field	Bits	Type	Description
MOD_REV	7:0	r	**Module Revision Number** MOD_REV defines the Module revision number.
MOD_TYPE	15:8	r	**Module Number Value** This bit field defines the module as a 32 bit module: $C0_H$
MOD_NUM	31:16	r	**Module Number Value** This bit field defines the identification number for the IOM.

48.4.2 Filter & Prescaler Channel (FPC) Registers

IOM Filter and Prescaler Channels Rising & Falling Edge Status Register

The Filter and Prescaler Edge Status Register stores the state of detected rising and falling edges from each of the FPC.

FPCESR.FEGk and FPCESR.REGk are written disregarding the filter configuration FPCCTRk.MOD.

The register can be used to selectively clear individual bits by writing a 0 in the respective bitfield.

Individual bits are also cleared with a write to the control register (FPCCTRk) or timer register (FPCTIMk).

NOTE: LDMST or SWAPMSK.W should be used only with bit mask enabled for all ´rwh´ bits in the addressed register.

FPCESR

IOM Filter and Prescaler Channels Rising & Falling Edge Status Register(078_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG15	**REG14**	**REG13**	**REG12**	**REG11**	**REG10**	**REG9**	**REG8**	**REG7**	**REG6**	**REG5**	**REG4**	**REG3**	**REG2**	**REG1**	**REG0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEG15	**FEG14**	**FEG13**	**FEG12**	**FEG11**	**FEG10**	**FEG9**	**FEG8**	**FEG7**	**FEG6**	**FEG5**	**FEG4**	**FEG3**	**FEG2**	**FEG1**	**FEG0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
FEGk (k=0-15)	k	rwh	**Falling Edge Glitch Flag for FPCk** 0_B No falling edge of glitch detected 1_B Falling edge of glitch detected
REGk (k=0-15)	k+16	rwh	**Rising Edge Glitch Flag for FPCk** 0_B No rising edge of glitch detected 1_B Rising edge of glitch detected

IOM Filter and Prescaler Channel Control Register k

The IOM Filter & Prescaler Control Register is used to configure the FPC and mux selection logic. A write to FPCTIMk will return FPCTIMk.TIM, FPCESR.REGk, and PCESR.FEGk to the reset value.

FPCCTRk (k=0-15)

IOM Filter and Prescaler Channel Control Register k(080_H+k*4) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0					**ISR**			**0**	**RTG**	**0**	**ISM**		**MOD**		
r					rw			r	rw	r	rw		rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP															
rw															

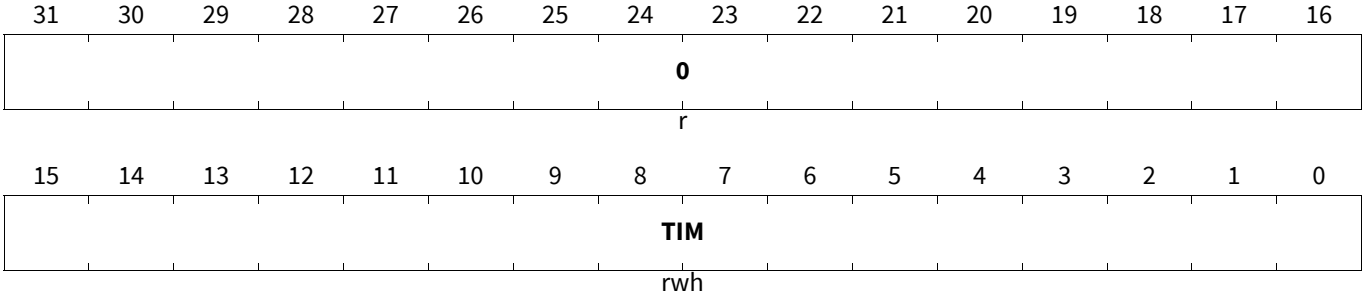
Field	Bits	Type	Description
CMP	15:0	rw	**Threshold Value of Filter & Prescaler Channel k** CMP is the 16-bit threshold value that is compared with the 16-bit timer value FPCTIMk.TIM.
MOD	18:16	rw	**Operation Mode Selection for Filter & Prescaler Channel k** 000_B Delayed Debounce Filter Mode on both edges 001_B Immediate Debounce Filter Mode on both edges 010_B Rising edge: Immediate Debounce Filter Mode, falling edge: no filtering 011_B Rising edge: no filtering, falling edge: Immediate Debounce Filter Mode 100_B Rising edge: Delayed Debounce Filter Mode, falling edge: Immediate Debounce Filter Mode 101_B Rising edge: Immediate Debounce Filter Mode, falling edge: Delayed Debounce Filter Mode 110_B Prescaler Mode (triggered on rising edge) 111_B Prescaler Mode (triggered on falling edge)
ISM	20:19	rw	**Monitor Input Signal Selection for Filter & Prescaler Channel k** ISM selects the monitor signal that goes to the monitor input (MON) of the Logic Analyzer Module. 00_B Processed input signal pin selected (SOL) 01_B Monitor Signal Input 0 (MON0) selected 10_B Monitor Signal Input 1 (MON1) selected 11_B Monitor Signal Input 2 (MON2) selected
RTG	22	rw	**Reset Timer behavior for Filter & Prescaler Channel k on Glitch** This bit is effective in Delayed Debounce Filter Mode only. 0_B Timer for FPCk is decremented on glitch 1_B Timer for FPCk is cleared on glitch
ISR	26:24	rw	**Reference Input Signal Selection for Filter & Prescaler Channel k** ISR select the reference signal that goes to the reference input (REF) of the Logic Analyzer Module. 000_B Processed input signal pin selected (SOL) 001_B Reference Signal Input 0selected 010_B Reference Signal Input 1selected 011_B Reference Signal Input 2selected 100_B GTM XOR Combiner selected … 111_B GTM XOR Combiner selected
0	21, 23, 31:27	r	**Reserved** Read as 0; shall be written with 0.

IOM Filter and Prescaler Channel Timer Register k

The IOM Filter & Prescaler Channel Timer Register is used to reset the value of the timer for the required application. A write to this register will result in its contents being set to the reset value. This is also undertaken whenever the FPCCTRk register is written with a new value. Writing FPCTIMk also clears the relevant fields of FPCESR.

FPCTIMk (k=0-15)

IOM Filter and Prescaler Channel Timer Register k ($0C0_H$+k*4) **Application Reset Value: $0000\ 0000_H$**



Field	Bits	Type	Description
TIM	15:0	rwh	**Timer Value of Filter and Prescaler Channel k** This bit bitfield shows the values of the timer of the Filter and Prescaler Channel. Writing to TIM will result in its content being set to its reset value.
0	31:16	r	**Reserved** Read as 0; shall be written with 0.

48.4.3 GTM Input Related Registers

IOM GTM Input EXOR Combiner Selection Register

Enables the incoming GTM-connected input signals for inclusion within a combined EXOR function.

GTMEXR

IOM GTM Input EXOR Combiner Selection Register(040_H) **Application Reset Value: $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
r								rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
EN0	0	rw	**GTM input 0 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN1	1	rw	**GTM input 1 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN2	2	rw	**GTM input 2 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN3	3	rw	**GTM input 3 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN4	4	rw	**GTM input 4 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN5	5	rw	**GTM input 5 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN6	6	rw	**GTM input 6 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
EN7	7	rw	**GTM input 7 selection for EXOR combiner** 0_B Input not selected for EXOR combiner. 1_B Input selected for EXOR combiner.
0	31:8	r	**Reserved** Read as 0; shall be written with 0.

48.4.4 Logic Analyzer Module (LAM) Registers

IOM Logic Analyzer Module Event Window Count Status Register m

Used to store the window count value reached in the LAM block once an event has been generated. If the edge which triggered the event also clears the counter, the value stored here is taken before clearing takes place. It also provides a bitfield for signaling the counter overflow. Any write command to one of the LAM registers (LAMEWCm, LAMCFGm and LAMEWSm) will clear this register.

LAMEWCm (m=0-15)
IOM Logic Analyzer Module Event Window Count Status Register m(100_H+m*4) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTO	0							CNT							
rwh	r							rwh							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
rwh															

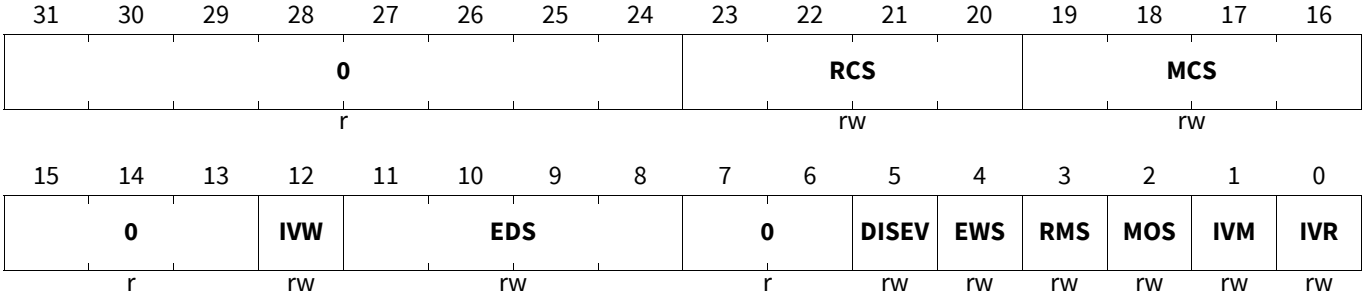
Field	Bits	Type	Description
CNT	23:0	rwh	**Event Window Count Value LAM block m** The count value of the event window attained coincident with an event occurring.
CNTO	31	rwh	**Count Overflow Flag LAM block m** This bit field provides the information wether the count has reached its maximum value. 0_B The count value has not reached its maximum value since the previous count clear. 1_B The count value has reached its maximum value since the previous count clear.
0	30:24	r	**Reserved** Read as 0; shall be written with 0.

IOM Logic Analyzer Module Configuration Register m

Used to configure the application-specific settings for each LAM block, including signal selection multiplexers at the block inputs.

LAMCFGm (m=0-15)

IOM Logic Analyzer Module Configuration Register m(180_H+m*4) Application Reset Value: $0000\ 0000_H$



Field	Bits	Type	Description
IVR	0	rw	**Invert Reference LAM block m** This bit field determines whether the reference signal from the FPC (reference channel) is inverted or not. 0_B Don't invert reference signal from FPC. 1_B Invert reference signal from FPC.
IVM	1	rw	**Invert Monitor LAM block m** This bit field determines whether the monitor signal from the FPC (monitor channel) is inverted or not. 0_B Don't invert monitor signal from FPC. 1_B Invert monitor signal from FPC.
MOS	2	rw	**Monitor Source Select LAM block m** This bit field determines whether the monitor signal from the FPC (monitor channel) is sourced directly or compared (EXOR'd) with the reference signal from the FPC(reference channel) for the event compare. 0_B Monitor signal is sourced directly from FPC(monitor channel). 1_B Monitor signal is EXOR'd with FPC(reference channel).
RMS	3	rw	**Runmode Select LAM block m** This bit field determines whether the event window generation is free-running or gated with the monitor or reference. 0_B Event window generation is free-running. 1_B Event window generation is gated with the monitor or reference signal.
EWS	4	rw	**Event Window Select LAM block m** This bit field determines whether the event window generation is from the monitor or reference signal. 0_B Event window generation is determined from the reference signal. 1_B Event window generation is determined from the monitor signal.

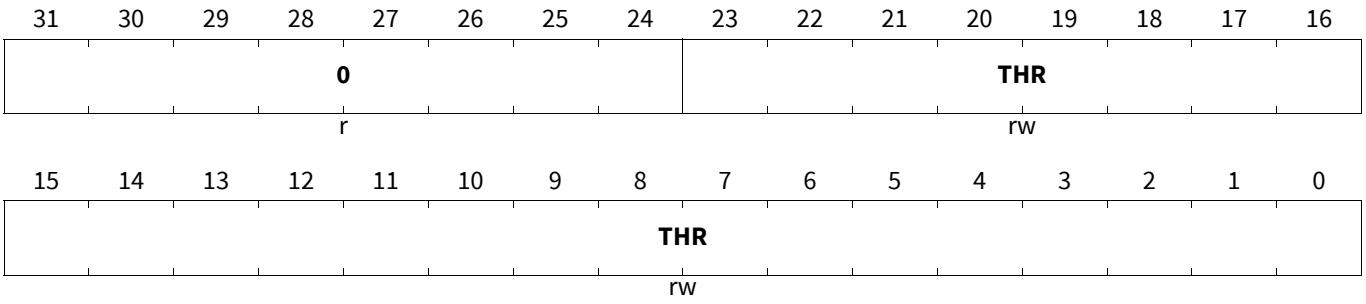
Field	Bits	Type	Description
DISEV	5	rw	**Disable Events LAM block m** This bit field allows to suppress alarm outputs from LAM block m to the ECM. Except for sending alarms to the ECM, all other effects of an alarm condition being detected (for instance, setting LAMEWCm.CNT to the value of the counter) will still take place inside LAM block m. 0_B Events will be generated. 1_B No events will be sent form LAM to ECM. Except for sending alarms to the ECM, all other effects of an alarm condition being detected (for instance, setting LAMEWCm.CNT to the value of the counter) will still take place inside LAM block m.
EDS	11:8	rw	**Event Window Active Edge Selection LAM block m** This bit field determines which active edges of the monitor and reference signals are used for the event window generation. 0_H Neither edge used to clear event window counter. Neither edge used to gate event generation. 1_H Positive edge used to clear event window counter. Neither edge used to gate event generation. 2_H Negative edge used to clear event window counter. Neither edge used to gate event generation. 3_H Either edge used to clear event window counter. Neither edge used to gate event generation. 4_H Neither edge used to clear event window counter. Positive edge used to gate event generation. 5_H Positive edge used to clear event window counter. Positive edge used to gate event generation. 6_H Negative edge used to clear event window counter. Positive edge used to gate event generation. 7_H Either edge used to clear event window counter. Positive edge used to gate event generation. 8_H Neither edge used to clear event window counter. Negative edge used to gate event generation. 9_H Positive edge used to clear event window counter. Negative edge used to gate event generation. A_H Negative edge used to clear event window counter. Negative edge used to gate event generation. B_H Either edge used to clear event window counter. Negative edge used to gate event generation. C_H Neither edge used to clear event window counter. Either edge used to gate event generation. D_H Positive edge used to clear event window counter. Either edge used to gate event generation. E_H Negative edge used to clear event window counter. Either edge used to gate event generation. F_H Either edge used to clear event window counter. Either edge used to gate event generation.

Field	Bits	Type	Description
IVW	12	rw	**Invert Event Window LAM block m** This bit field determines whether the event window polarity is inverted or not. 0_B Event window non-inverted. 1_B Event window inverted.
MCS	19:16	rw	**Monitor Input Signal Selection LAM block m** This bit field determines which FPC/mux block k monitor output signal is to be used for LAM block m. 0_H Monitor signal provided by FPC/mux block 0. 1_H Monitor signal provided by FPC/mux block 1. 2_H Monitor signal provided by FPC/mux block 2. 3_H Monitor signal provided by FPC/mux block 3. 4_H Monitor signal provided by FPC/mux block 4. 5_H Monitor signal provided by FPC/mux block 5. 6_H Monitor signal provided by FPC/mux block 6. 7_H Monitor signal provided by FPC/mux block 7. 8_H Monitor signal provided by FPC/mux block 8. 9_H Monitor signal provided by FPC/mux block 9. A_H Monitor signal provided by FPC/mux block 10. B_H Monitor signal provided by FPC/mux block 11. C_H Monitor signal provided by FPC/mux block 12. D_H Monitor signal provided by FPC/mux block 13. E_H Monitor signal provided by FPC/mux block 14. F_H Monitor signal provided by FPC/mux block 15.
RCS	23:20	rw	**Reference Input Signal Selection LAM block m** This bit field determines which FPC/mux block k reference output signal is to be used for LAM block m. 0_H Reference signal provided by FPC/mux block 0. 1_H Reference signal provided by FPC/mux block 1. 2_H Reference signal provided by FPC/mux block 2. 3_H Reference signal provided by FPC/mux block 3. 4_H Reference signal provided by FPC/mux block 4. 5_H Reference signal provided by FPC/mux block 5. 6_H Reference signal provided by FPC/mux block 6. 7_H Reference signal provided by FPC/mux block 7. 8_H Reference signal provided by FPC/mux block 8. 9_H Reference signal provided by FPC/mux block 9. A_H Reference signal provided by FPC/mux block 10. B_H Reference signal provided by FPC/mux block 11. C_H Reference signal provided by FPC/mux block 12. D_H Reference signal provided by FPC/mux block 13. E_H Reference signal provided by FPC/mux block 14. F_H Reference signal provided by FPC/mux block 15.
0	7:6, 15:13, 31:24	r	**Reserved** Read as 0; shall be written with 0.

IOM Logic Analyzer Module Event Window Configuration Register m

Used to set the threshold value for the event window counter.

LAMEWSm (m=0-15)
IOM Logic Analyzer Module Event Window Configuration Register m(1C0$_H$+m*4)Application Reset Value: 0000 0000$_H$



Field	Bits	Type	Description
THR	23:0	rw	**Event Window Count Threshold** This bit field contains the value of the counter at which the event window becomes active (before optional inversion).
0	31:24	r	**Reserved** Read as 0; shall be written with 0.

48.4.5 Event Combiner Module (ECM) Registers

IOM Event Combiner Module Counter Configuration Register

The Event Combiner Configuration register is used to configure each of the four event counter submodules, enabling multiple events from one selected LAM block (for each event counter submodule) to be counted in the generation of a system event. Writing to this register will clear the internal ECM counters.

ECMCCFG

IOM Event Combiner Module Counter Configuration Register(030_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
THRC3				SELC3				THRC2				SELC2			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THRC1				SELC1				THRC0				SELC0			
rw				rw				rw				rw			

Field	Bits	Type	Description
SELC0	3:0	rw	**Event Channel Select** This bit field determines which channel event output is to be routed to counter C0. 0_H Select channel 0 event output to be counted. 1_H Select channel 1 event output to be counted. 2_H Select channel 2 event output to be counted. 3_H Select channel 3 event output to be counted. 4_H Select channel 4 event output to be counted. 5_H Select channel 5 event output to be counted. 6_H Select channel 6 event output to be counted. 7_H Select channel 7 event output to be counted. 8_H Select channel 8 event output to be counted. 9_H Select channel 9 event output to be counted. A_H Select channel 10 event output to be counted. B_H Select channel 11 event output to be counted. C_H Select channel 12 event output to be counted. D_H Select channel 13 event output to be counted. E_H Select channel 14 event output to be counted. F_H Select channel 15 event output to be counted.
THRC0	7:4	rw	**Channel Event Counter Threshold** This bit field determines the threshold count value. Upon the counter meeting the threshold, the counter event output becomes active high for one clock cycle and the count is reset to zero. 0_H Counter disabled (counter event output set to inactive, despite any incoming channel events). 1_H Minimum threshold count value. F_H Maximum threshold count value.

Field	Bits	Type	Description
SELC1	11:8	rw	**Event Channel Select** This bit field determines which channel event output is to be routed to counter C1. 0_H Select channel 0 event output to be counted. 1_H Select channel 1 event output to be counted. 2_H Select channel 2 event output to be counted. 3_H Select channel 3 event output to be counted. 4_H Select channel 4 event output to be counted. 5_H Select channel 5 event output to be counted. 6_H Select channel 6 event output to be counted. 7_H Select channel 7 event output to be counted. 8_H Select channel 8 event output to be counted. 9_H Select channel 9 event output to be counted. A_H Select channel 10 event output to be counted. B_H Select channel 11 event output to be counted. C_H Select channel 12 event output to be counted. D_H Select channel 13 event output to be counted. E_H Select channel 14 event output to be counted. F_H Select channel 15 event output to be counted.
THRC1	15:12	rw	**Channel Event Counter Threshold** This bit field determines the threshold count value. Upon the counter meeting the threshold, the counter event output becomes active high for one clock cycle and the count is reset to zero. 0_H Counter disabled (counter event output set to inactive, despite any incoming channel events). 1_H Minimum threshold count value. F_H Maximum threshold count value.
SELC2	19:16	rw	**Event Channel Select** This bit field determines which channel event output is to be routed to counter C2. 0_H Select channel 0 event output to be counted. 1_H Select channel 1 event output to be counted. 2_H Select channel 2 event output to be counted. 3_H Select channel 3 event output to be counted. 4_H Select channel 4 event output to be counted. 5_H Select channel 5 event output to be counted. 6_H Select channel 6 event output to be counted. 7_H Select channel 7 event output to be counted. 8_H Select channel 8 event output to be counted. 9_H Select channel 9 event output to be counted. A_H Select channel 10 event output to be counted. B_H Select channel 11 event output to be counted. C_H Select channel 12 event output to be counted. D_H Select channel 13 event output to be counted. E_H Select channel 14 event output to be counted. F_H Select channel 15 event output to be counted.

Field	Bits	Type	Description
THRC2	23:20	rw	**Channel Event Counter Threshold** This bit field determines the threshold count value. Upon the counter meeting the threshold, the counter event output becomes active high for one clock cycle and the count is reset to zero. 0_H Counter disabled (counter event output set to inactive, despite any incoming channel events). 1_H Minimum threshold count value. F_H Maximum threshold count value.
SELC3	27:24	rw	**Event Channel Select** This bit field determines which channel event output is to be routed to counter C3. 0_H Select channel 0 event output to be counted. 1_H Select channel 1 event output to be counted. 2_H Select channel 2 event output to be counted. 3_H Select channel 3 event output to be counted. 4_H Select channel 4 event output to be counted. 5_H Select channel 5 event output to be counted. 6_H Select channel 6 event output to be counted. 7_H Select channel 7 event output to be counted. 8_H Select channel 8 event output to be counted. 9_H Select channel 9 event output to be counted. A_H Select channel 10 event output to be counted. B_H Select channel 11 event output to be counted. C_H Select channel 12 event output to be counted. D_H Select channel 13 event output to be counted. E_H Select channel 14 event output to be counted. F_H Select channel 15 event output to be counted.
THRC3	31:28	rw	**Channel Event Counter Threshold** This bit field determines the threshold count value. Upon the counter meeting the threshold, the counter event output becomes active high for one clock cycle and the count is reset to zero. 0_H Counter disabled (counter event output set to inactive, despite any incoming channel events). 1_H Minimum threshold count value. F_H Maximum threshold count value.

IOM Event Combiner Module Global Event Selection Register

Configured to combine individual and multiple (counted) events from the LAM modules in order to generate a global (system) event. Writing to this register will clear the internal ECM counters.

ECMSELR

IOM Event Combiner Module Global Event Selection Register(034_H) Application Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												CTS3	CTS2	CTS1	CTS0
r												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CES15	CES14	CES13	CES12	CES11	CES10	CES9	CES8	CES7	CES6	CES5	CES4	CES3	CES2	CES1	CES0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CESn (n=0-15)	n	rw	**Event Combiner Selection** The setting of individual bitfields determines the inclusion of the respective channel event in the generation of the global event. 0_B Don't include channel event/event counter output within the global event generation. 1_B Include channel event/event counter output within the global event generation.
CTSn (n=0-3)	n+16	rw	**Accumulated (Counted) Event Combiner Selection** The setting of individual bitfields determines the inclusion of the respective channel event counter output (1 of 4) in the generation of the global event. 0_B Don't include channel event/event counter output within the global event generation. 1_B Include channel event/event counter output within the global event generation.
0	31:20	r	**Reserved** Read as 0; shall be written with 0.

IOM Event Combiner Module Event Trigger History Register 0

One of two registers (IOM_ECMETH0 and IOM_ECMETH1) that record the status of LAM's 0 to 15 event outputs for the generated system events.

The generation of a system event will cause the status of the event outputs from LAM blocks 0 to 15 that generated the system event to be recorded in ETA0-15, respectively. Just prior to this happening, the previous status held in ETA0-15 is moved to ETB0-15. Similarly, the previous content of ETB0-15 is moved to ETC0-15. In addition, the previous content of the ETC0-15 is OR'ed with the current value of ETD15-0 and placed in ETD15-0. The contents of ETD15-0 will be maintained until the application explicitly clears the ETD15-0 bits.

At reset (or a register write to either IOM_ECMETH0 or IOM_ECMETH1), all Event Trigger Active flags (ETA, ETB, ETC, ETD) will be cleared.

ECMETH0

IOM Event Combiner Module Event Trigger History Register 0(038_H)Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETB15	**ETB14**	**ETB13**	**ETB12**	**ETB11**	**ETB10**	**ETB9**	**ETB8**	**ETB7**	**ETB6**	**ETB5**	**ETB4**	**ETB3**	**ETB2**	**ETB1**	**ETB0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETA15	**ETA14**	**ETA13**	**ETA12**	**ETA11**	**ETA10**	**ETA9**	**ETA8**	**ETA7**	**ETA6**	**ETA5**	**ETA4**	**ETA3**	**ETA2**	**ETA1**	**ETA0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
ETAn (n=0-15)	n	rwh	**LAM 0-15 Event Trigger Activity (last)** Contains the status of the event trigger outputs for each of the LAM blocks for the last generated event(s). 0_B LAM channel n did not trigger a system event. 1_B LAM channel n triggered a system event.
ETBn (n=0-15)	n+16	rwh	**LAM 0-15 Event Trigger Activity (previous ETA0-15)** Contains the previous contents of ETA0-15 prior to being updated. 0_B LAM channel n did not trigger a system event. 1_B LAM channel n triggered a system event.

IOM Event Combiner Module Event Trigger History Register 1

One of two registers (IOM_ECMETH0 and IOM_ECMETH1) that record the status of LAM's 0 to 15 event outputs for the generated system events.

The generation of a system event will cause the status of the event outputs from LAM blocks 0 to 15 that generated the system event to be recorded in ETA0-15, respectively. Just prior to this happening, the previous status held in ETA0-15 is moved to ETB0-15. Similarly, the previous content of ETB0-15 is moved to ETC0-15. In addition, the previous content of the ETC0-15 is OR'ed with the current value of ETD15-0 and placed in ETD15-0. The contents of ETD15-0 will be maintained until the application explicitly clears the ETD15-0 bits.

At reset (or a register write to either IOM_ECMETH0 or IOM_ECMETH1), all Event Trigger Active flags (ETA, ETB, ETC, ETD) will be cleared.

ECMETH1

IOM Event Combiner Module Event Trigger History Register 1($03C_H$)Application Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETD15	**ETD14**	**ETD13**	**ETD12**	**ETD11**	**ETD10**	**ETD9**	**ETD8**	**ETD7**	**ETD6**	**ETD5**	**ETD4**	**ETD3**	**ETD2**	**ETD1**	**ETD0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETC15	**ETC14**	**ETC13**	**ETC12**	**ETC11**	**ETC10**	**ETC9**	**ETC8**	**ETC7**	**ETC6**	**ETC5**	**ETC4**	**ETC3**	**ETC2**	**ETC1**	**ETC0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
ETCn (n=0-15)	n	rwh	**LAM 0-15 Event Trigger Activity (previous ETB0-15)** Contains the previous contents of ETB0-15 prior to being updated. 0_B LAM channel n did not trigger a system event. 1_B LAM channel n triggered a system event.
ETDn (n=0-15)	n+16	rwh	**LAM 0-15 Event Trigger Activity (previous contents of ETC0-15 OR'ed with the previous value of ETD0-15)** Contains the previous content of ETC0-15 OR'ed with the previous value of ETD0-15. 0_B LAM channel n did not trigger a system event. 1_B LAM channel n triggered a system event.

48.4.6 System Registers

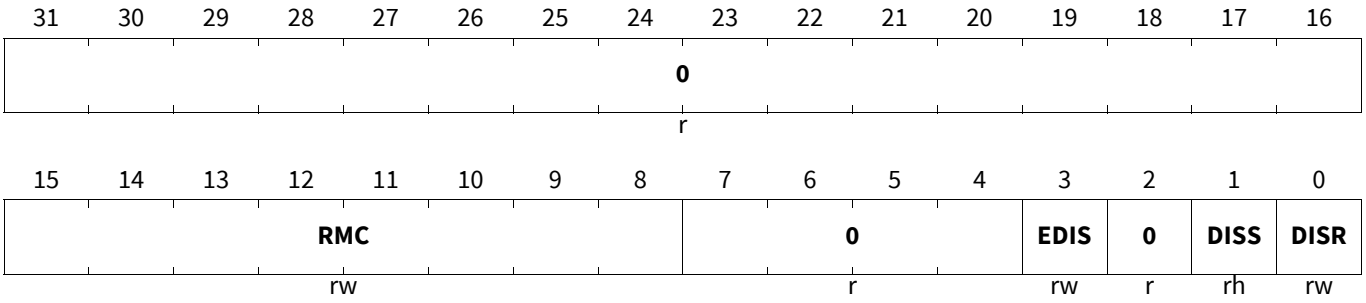
IOM Clock Control Register

The Clock Control Register CLC allows the programmer to adapt the functionality and power consumption of the module to the requirements of the application. The description below shows the clock control register functionality which is implemented in the BPI_FPI for the module. Where a module kernel is connected to the CLC clock control interface, CLC controls the f_{IOM} module clock signal, sleep mode and disable mode for the module.

Note: The number of module clock cycles (wait states) which are required by the kernel to execute a read or write access depends on the selected CLC clock frequency, which is selected via bit field RMC in the CLC register. Therefore, increasing CLC.RMC may result in a longer FPI Bus read cycle access time for kernel registers and can also slow down the write throughput to the kernel registers.

CLC

IOM Clock Control Register (000_H) Application Reset Value: 0000 0003$_H$



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested 1_B Module disable is requested
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module. 0_B Module is enabled 1_B Module is disabled
EDIS	3	rw	**Sleep Mode Enable Control** Used to control module's sleep mode. 0_B Sleep mode request is regarded. Module is enabled to go to into Sleep Mode. 1_B Sleep mode request is disregarded. Sleep Mode cannot be entered upon a request.
RMC	15:8	rw	**8-bit Clock Divider Value in RUN Mode** 00_H No clock signal fIOM generated (default after reset) 01_H Clock fIOM equal to the input clock frequency selected 02_H fIOM = (input clock frequency)/2 selected … FF_H fIOM = (input clock frequency)/255 selected
0	2, 7:4, 31:16	r	**Reserved** Read as 0; shall be written with 0.

IOM Access Enable Register 0

The Access Enable Register 0 controls write access[1] for transactions with the on chip bus master TAG ID 000000B to 011111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. The registers ACCEN0 / ACCEN1 are providing one enable bit for each possible 6-bit TAG ID encoding.

Mapping of TAG IDs to ACCEN0.ENx: EN0 -> TAG ID 000000B, EN1 -> TAG ID 000001B, … , EN31 -> TAG ID 011111B.

ACCEN0

IOM Access Enable Register 0 ($02C_H$) Application Reset Value: FFFF FFFF$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	**EN30**	**EN29**	**EN28**	**EN27**	**EN26**	**EN25**	**EN24**	**EN23**	**EN22**	**EN21**	**EN20**	**EN19**	**EN18**	**EN17**	**EN16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	**EN14**	**EN13**	**EN12**	**EN11**	**EN10**	**EN9**	**EN8**	**EN7**	**EN6**	**EN5**	**EN4**	**EN3**	**EN2**	**EN1**	**EN0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n=0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the module kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

IOM Access Enable Register 1

The Access Enable Register 1 controls write access[1] for transactions with the on chip bus master TAG ID 100000B to 111111B (see On Chip Bus chapter for the products TAG ID <-> master peripheral mapping). The BPI_FPI is prepared for a 6-bit TAG ID. ACCEN1 is not implemented with register bits as the related TAG IDs are not used in this family of devices.

Mapping of TAG IDs to ACCEN1.ENx: EN0 -> TAG ID 100000B, EN1 -> TAG ID 100001B, … , EN31 -> TAG ID 111111B.

ACCEN1

IOM Access Enable Register 1 (028_H) Application Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

1) The BPI_FPI Access Enable functionality controls only write transactions to the CLC, OCS, KRSTx and the kernel registers. Read transactions are not influenced. SW has to take care for destructive/modifying read functionality in kernel registers

Field	Bits	Type	Description
0	31:0	r	**Reserved** Read as 0; shall be written with 0.

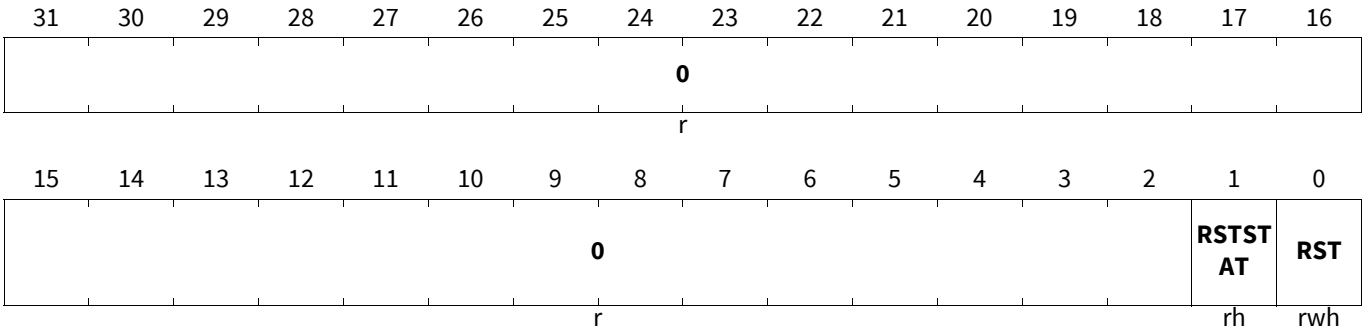
IOM Kernel Reset Register 0

The Kernel Reset Register 0 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset Registers.The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

Kernel Reset Register 0 includes a kernel reset status bit that is set to ´1´ by the BPI_FPI in the same clock cycle the RST bit is re-set by the BPI_FPI. This bit can be used to detect that a kernel reset was processed. The bit can be re-set to ´0´ by writing ´1´ to the KRSTCLE.CLR register bit.

During the execution of the kernel reset until RSTSTAT is set, access to the kernel registers will result in an error acknowledge.

KRST0
IOM Kernel Reset Register 0 **(024_H)** **Application Reset Value: $0000\ 0000_H$**

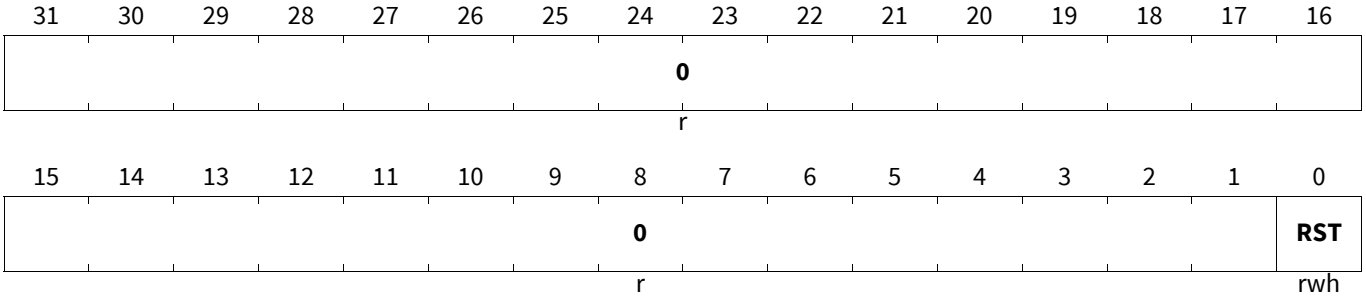


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel registers are set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
RSTSTAT	1	rh	**Kernel Reset Status** This bit indicates wether a kernel reset was executed or not. This bit is set by the BPI_FPI after the execution of a kernel reset in the same clock cycle both reset bits. This bit can be cleared by writing with ´1´ to the CLR bit in the related KRSTCLR register. 0_B No kernel reset was executed 1_B Kernel reset was executed
0	31:2	r	**Reserved** Read as 0; should be written with 0.

IOM Kernel Reset Register 1

The Kernel Reset Register 1 is used to reset the related module kernel. Kernel registers related to the Debug Reset (Class 1) are not influenced. To reset a module kernel it is necessary to set the RST bits by writing with ´1´ in both Kernel Reset registers. The RST bit will be re-set by the BPI with the end of the BPI kernel reset sequence.

KRST1
IOM Kernel Reset Register 1 **(020_H)** **Application Reset Value: 0000 0000_H**

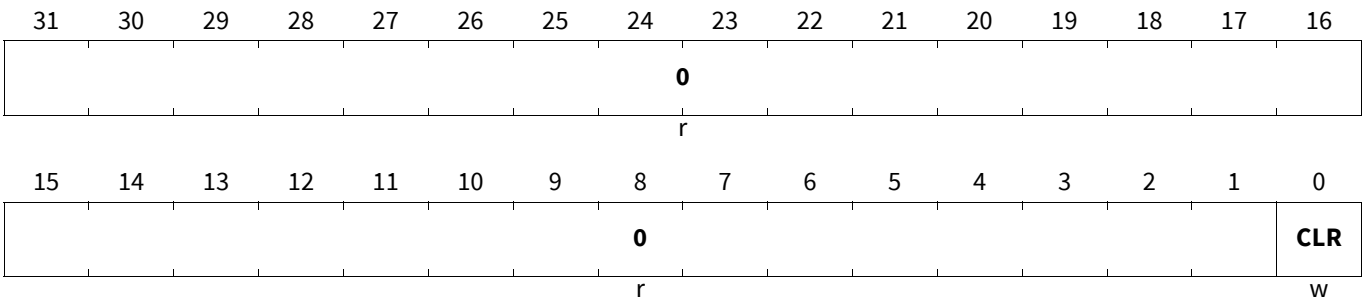


Field	Bits	Type	Description
RST	0	rwh	**Kernel Reset** This reset bit can be used to request for a kernel reset. The kernel reset will be executed if the reset bits of both kernel reset registers is set. The RST bit will be cleared (re-set to ´0´) by the BPI_FPI after the kernel reset was executed. 0_B No kernel reset was requested 1_B A kernel reset was requested
0	31:1	r	**Reserved** Read as 0; shall be written with 0.

IOM Kernel Reset Status Clear Register

The Kernel Reset Status Clear register is used to clear the Kernel Reset Status bit (KRST0.RSTSTAT).

KRSTCLR
IOM Kernel Reset Status Clear Register **($01C_H$)** **Application Reset Value: 0000 0000_H**



Field	Bits	Type	Description
CLR	0	w	**Kernel Reset Status Clear** Read always as 0. 0_B No action 1_B Clear Kernel Reset Status KRST0.RSTSTAT

Field	Bits	Type	Description
0	31:1	r	**Reserved** Read as 0; shall be written with 0.

48.5 IO Interfaces

The table below lists all the interfaces of the IOM to other modules.

Table 544 List of IOM Interface Signals

Interface Signals	I/O	Description
MON0(15:0)	in	**Monitor input 0** One bit per channel.
MON1(15:0)	in	**Monitor input 1** One bit per channel.
MON2(15:0)	in	**Monitor input 2** One bit per channel.
REF0(15:0)	in	**Reference input 0** One bit per channel.
REF1(15:0)	in	**Reference input 1** One bit per channel.
REF2(15:0)	in	**Reference input 2** One bit per channel.
PIN(15:0)	in	**GPIO pad input to FPC** One bit per channel.
GTM(7:0)	in	**GTM-provided inputs to EXOR combiner**
alarm	out	**Event for SMU** This signal is set to 1 if IOM mismatch is detected.

48.6 Revision History

Table 545 Revision History

Reference	Change to Previous Version	Comment
V2.1.15		
Page 24	In register FPCCTRk (k=0-15) bitfield descriptions for ISM, RTG and ISR changed.	–
Page 33	In register ECMCCFG bitfield descriptions for SELC0 and THRC1 changed.	–
–	Previous versions removed from revision history.	

49 8-Bit Standby Controller (SCR)

The SCR is an 8-bit microcontroller that can continue to run during the standby mode. It is based on the XC800 Core that is compatible with the industry standard 8051 processor. There is an embedded 8 KB XRAM for program code and data.

Key features include a 16-bit general purpose timer with a capture/compare unit (T2CCU) for digital signal generation like pulse generation, pulse width modulation and pulse width measuring. It also includes a Real Time Clock (RTC) to support periodic wake-up in standby mode and an On-Chip Debug Support (OCDS) unit for software development and debugging of XC800-based systems. Local Interconnect Network (LIN) applications are also supported through extended UART features and the provision of LIN low level drivers for the AURIX™ TC3xx Platform. For low power applications, various power saving techniques are available for selection by the user. Control of the numerous on-chip peripheral functionalities is achieved by extending the Special Function Register (SFR) address range with an intelligent paging mechanism optimized for interrupt handling.

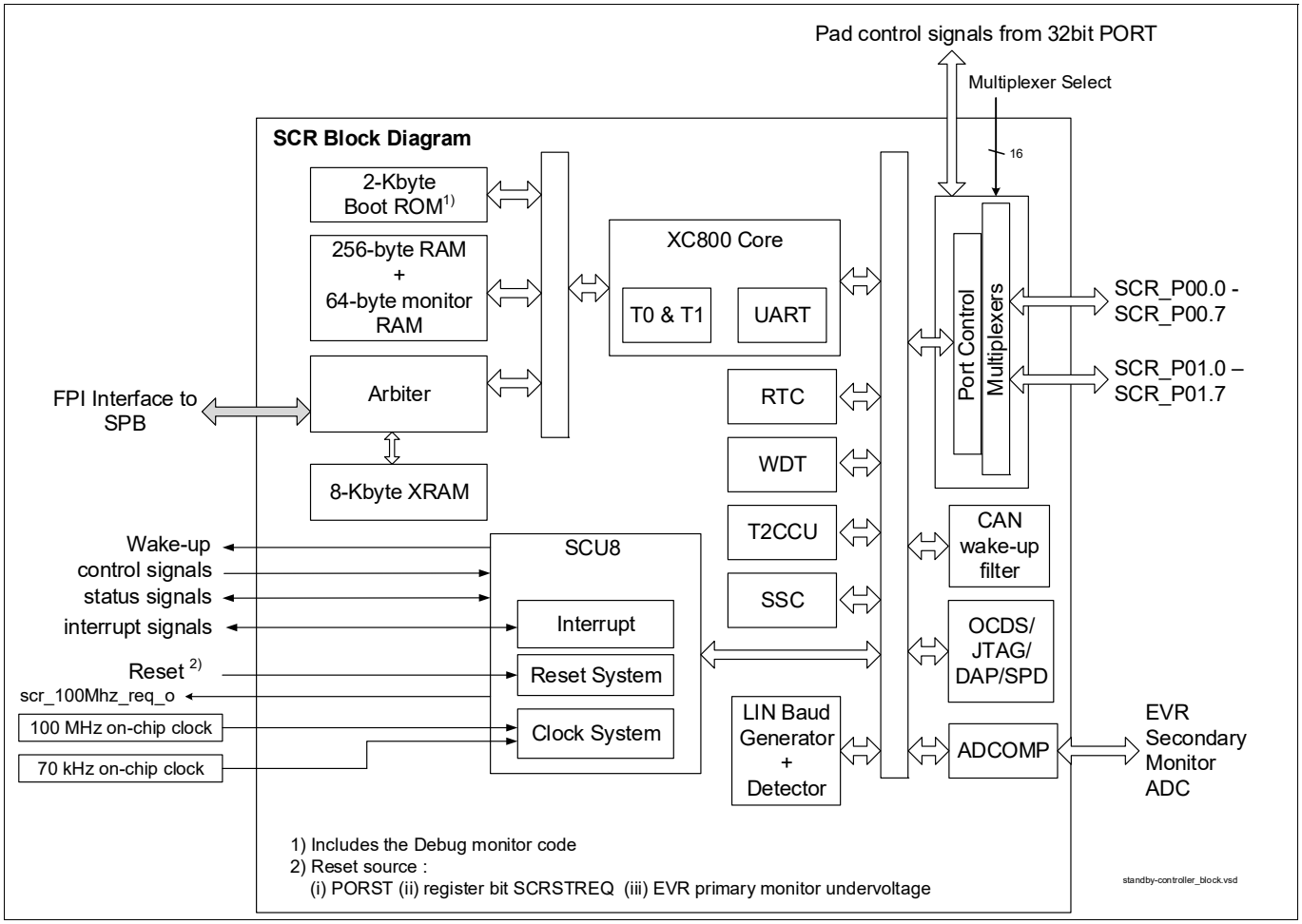


Figure 801 SCR Block Diagram

Note: Regarding the version numbers in the bottom line of the pages in this SCR chapter, please note that the 1st number represents the AURIX-SCR-sub-chapter version, and the 2nd number represents the SCR chapter version.
Note that if there are changes in the SCR sub-chapters, one or both version numbers will change.

49.1 Features of the SCR

The following list summarizes the main features of the SCR:

- High performance XC800 Core
  - compatible to standard 8051 Core
  - two clocks per CPU cycle architecture (for memory access without wait state)
  - two data pointers
  - maximum 20 MHz of core frequency
- On-chip Memory
  - 2-Kbyte Boot ROM for startup firmware
  - 256-byte RAM; plus 64-byte Monitor RAM
  - 8-Kbyte XRAM for program code and data; accessible by CPUx via the main SPB bus.
- Power saving modes
  - idle mode
  - clock gating control to each peripheral
- Watchdog Timer (WDT) with programmable window feature for refresh operation and warning prior to overflow
- Two general purpose I/O ports
  - able to control the shared pins in SCR Ports P00 and P01 (P33 and P34 of TriCore)
- Three 16-bit timers - Timer 0, Timer 1, Timer 2 of T2CCU
- Capture/compare unit of T2CCU for digital signal generation
- Real-time clock with on-chip oscillator to support periodic wake-up in standby mode
- Full duplex serial interface (UART)
- Synchronous serial channel (SSC)
- ADCOMP unit which provides ADC compare functionality.
- Interrupt supported from SCR to TriCore and vice versa.
- XRAM Programmable via the System Peripheral Bus.
- On-chip Debug Support via single pin DAP interface (SPD)
  - 1.5-Kbyte monitor ROM (part of the Boot ROM)
  - 64-byte monitor RAM

49.2 Revision History

Table 546 Revision History

Reference	Change to Previous Version	
V2.3		
	First Official Release of completely reworked SCR chapter	
	No change	
V2.4		
Page 1	Added note in first chapter regarding version numbers	
	Corrected chapter number.	

49.3 XC800 CPU

This chapter describes the XC800 CPU.

49.3.1 SFRs of the CPU

The XC800 CPU registers occupy direct Internal Data Memory space locations in the range 80_H to FF_H.

49.3.1.1 Stack Pointer (SP, $D4_H$)

The SP register contains the Stack Pointer. The Stack Pointer is used to load the program counter into internal data memory during LCALL and ACALL instructions and to retrieve the program counter from memory during RET and RETI instructions. Data may also be saved on or retrieved from the stack using PUSH and POP instructions. Instructions that use the stack automatically pre-increment or post-decrement the stack pointer so that the stack pointer always points to the last byte written to the stack, i.e. the top of the stack. On reset, the Stack Pointer is reset to 07_H. This causes the stack to begin at a location = 08_H above register bank zero. The SP can be read or written under software control. The programmer must ensure that the location and size of the stack in internal data memory do not overlap with other application data.

49.3.1.2 Data Pointer (DPTR, D5-6_H)

The Data Pointer (DPTR) is stored in registers DPL (Data Pointer Low byte) and DPH (Data Pointer High byte) to form 16-bit addresses for External Data Memory accesses (MOVX A,@DPTR and MOVX @DPTR,A), for program byte moves (MOVC A,@A+DPTR) and for indirect program jumps (JMP @A+DPTR).

Two true 16-bit operations are allowed on the Data Pointer: load immediate (MOV DPTR, #data) and increment (INC DPTR).

49.3.1.3 Accumulator (ACC, $E0_H$)

This register provides one of the operands for most ALU operations. ACC is the symbol for the accumulator register. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as “A”.

49.3.1.4 B Register (DA_H)

The B register is used during multiply and divide operations to provide the second operand. For other instructions, it can be treated as another scratch pad register.

49.3.1.5 Program Status Word (PSW, $D0_H$)

The PSW contains several status bits that reflect the current state of the core.

Program Status Word Register

PSW
Program Status Word Register **($0D0_H$)** **Reset Value: Table 547**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
CY	**AC**	**F0**	**RS**		**OV**	**F1**	**P**
rwh	rwh	rw	rw		rwh	rw	rh

Field	Bits	Type	Description
P	0	rh	**Parity Flag** Set/cleared by hardware after each instruction to indicate an odd/even number of "one" bits in the accumulator, i.e. even parity.
F1	1	rw	**General Purpose Flag**
OV	2	rwh	**Overflow Flag** Used by arithmetic instructions.
RS	4:3	rw	**Register Bank Select** These bits are used to select one of the four register banks. 00_B **BANK0**, Bank 0 selected, data address 0x00 - 0x07 01_B **BANK1**, Bank 1 selected, data address 0x08 - 0x0F 10_B **BANK2**, Bank 2 selected, data address 0x10 - 0x17 11_B **BANK3**, Bank 3 selected, data address 0x18 - 0x1F
F0	5	rw	**General Purpose Flag**
AC	6	rwh	**Auxiliary Carry Flag** Used by instructions which execute BCD operations.
CY	7	rwh	**Carry Flag** Used by arithmetic instructions.

Table 547 Reset Values of PSW

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

49.3.1.6 Extended Operation Register (EO, $D7_H$)

The instruction set includes an additional instruction MOVC @(DPTR++),A which writes to program memory implemented as RAM. This instruction may be used both to download code into the program memory when the CPU is initialized and subsequently, to provide software updates. The instruction copies the contents of the accumulator to the code memory at the location pointed to by the current data pointer, then increments the data pointer.

The instruction uses the opcode $A5_H$, which is the same as the software break instruction TRAP (see **Table 550**). Bit TRAP_EN in the Extended Operation (EO) register is used to select the instruction executed by the opcode $A5_H$. When bit TRAP_EN is 0 (default), the $A5_H$ opcode executes the MOVC instruction. When bit TRAP_EN is 1, the $A5_H$ opcode executes the software break instruction TRAP, which switches the CPU to debug mode for breakpoint processing.

Register EO is also used to select the current data pointer.

Extended Operation Register

EO
Extended Operation Register **($0D7_H$)** **Reset Value: Table 548**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
0			**TRAP_EN**	**0**	**DPSEL**		
r			rw	r	rw		

Field	Bits	Type	Description
DPSEL	2:0	rw	**Data Pointer Select** These bits are used to select the current data pointer. 000_B **DTPR0**, DPTR0 selected 001_B **DTPR1**, DPTR1 selected **others**, Reserved
TRAP_EN	4	rw	**TRAP Enable** 0_B **SEL_MOVC**, Select MOVC @(DPTR++),A 1_B **SEL_TRAP**, Select software TRAP instruction
0	3, 7:5	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 548 Reset Values of EO

Reset Type	Reset Value	Note
Generated Reset	$XXX0\ X000_B$	
LVD Reset	$---0\ -000_B$	

49.3.1.7 Power Control Register (PCON, D9$_H$)

The PCON register provides control for entering idle mode, baud rate control for UART in mode 2, as well as two general purpose flags.

The XC800 CPU has two power-saving modes: idle mode and power-down mode. The idle mode can be entered via the PCON register. In idle mode, the clock to the core is disabled while the timers, serial port and interrupt controller continue to run. In power-down mode 1, the clock to the entire core is stopped.

Note: Power down mode is not available in SCR.

Power Control Register

PCON
Power Control Register (0D9$_H$) Reset Value: Table 549
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
SMOD		**0**		**GF1**	**GF0**	**0**	**IDLE**
rw		r		rw	rw	r	rw

Field	Bits	Type	Description
IDLE	0	rw	**Idle Mode Enable** 0_B Do not enter Idle Mode 1_B Enter Idle Mode
GF0	2	rw	**General Purpose Flag Bit 0**
GF1	3	rw	**General Purpose Flag Bit 1**
SMOD	7	rw	**Double Baud Rate Enable** 0_B Do not double the baud rate of serial interface in mode 2 1_B Double the baud rate of serial interface in mode 2
0	1, 6:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 549 Reset Values of PCON

Reset Type	Reset Value	Note
Generated Reset	0XXX 00X0$_B$	
LVD Reset	0––– 00–0$_B$	

49.3.1.8 Interrupt Registers

One non-maskable and fourteen maskable interrupt nodes are available.

Refer to Interrupt chapter for details of the interrupt registers.

49.3.2 SFRs of The Core Peripherals

49.3.2.1 Timer Registers

Two 16-bit timers are provided - Timer 0 (T0) and Timer 1 (T1).

Refer to Timer 0 and Timer 1 chapter for details of the timer registers.

49.3.2.2 UART Registers

The UART uses three SFRs - PCON, SCON and SBUF.

Refer to **Section 49.3.1.7** and UART chapter for details of the UART registers.

49.3.3 Instruction Timing

A CPU cycle comprises two input clock periods, referred to as Phase 1 (P1) and Phase 2 (P2), that correspond to two different CPU states. A CPU state within an instruction is referenced by the CPU cycle and state number, e.g., C2P1 means the first clock period within CPU cycle 2. Memory access takes place during one or both phases of the CPU cycle. SFR writes occur only at the end of P2. Instructions are 1, 2, or 3 bytes long and can take 1, 2 or 4 CPU cycles to execute. Registers are generally updated and the next opcode pre-fetched at the end of P2 of the last CPU cycle for the current instruction.

The XC800 CPU supports access to slow memory by using wait cycle(s). Each wait cycle lasts one CPU cycle, i.e. two clock periods. For example, in case of a memory requiring one/two wait state(s), the access time is increased by one CPU cycle for every byte of opcode/operand fetched.

Figure 802 shows the fetch/execute timing related to the internal states and phases. Execution of an instruction occurs at C1P1. For a 2-byte instruction, the second reading starts at C1P1.

Figure 802 (a) shows two timing diagrams for a 1-byte, 1-cycle (1 x CPU cycle) instruction. The first diagram shows the instruction being executed within one CPU cycle since the opcode (C1P2) is fetched from a memory without wait state. The second diagram shows the corresponding states of the same instruction being executed over two CPU cycles (instruction time extended), with one wait cycle inserted for opcode fetching from the flash memory.

Figure 802 (b) shows two timing diagrams for a 2-byte, 1-cycle (1 x CPU cycle) instruction. The first diagram shows the instruction being executed within one CPU cycle since the second byte (C1P1) and the opcode (C1P2) are fetched from a memory without wait state. The second diagram shows the corresponding states of the same instruction being executed over three CPU cycles (instruction time extended), with one wait cycle inserted for each access to the flash memory. In this case, two wait cycles are inserted in total.

Figure 802 (c) shows two timing diagrams of a 1-byte, 2-cycle (2 x CPU cycle) instruction. The first diagram shows the instruction being executed over two CPU cycles with the opcode (C2P2) fetched from a memory without wait state. The second diagram shows the corresponding states of the same instruction being executed over three CPU cycles (instruction time extended), with one wait cycle inserted for opcode fetching from the slow memory requiring one/two wait state(s).

Note: For instructions that are executed over two or more CPU cycles, execution cycle may or may not be extended in case of access to slow memory with one/two wait states. The execution cycle is,

nonetheless, guaranteed consistent for each instruction when accessed from slow memory with defined wait state(s). Reference: ***Table 550****.*

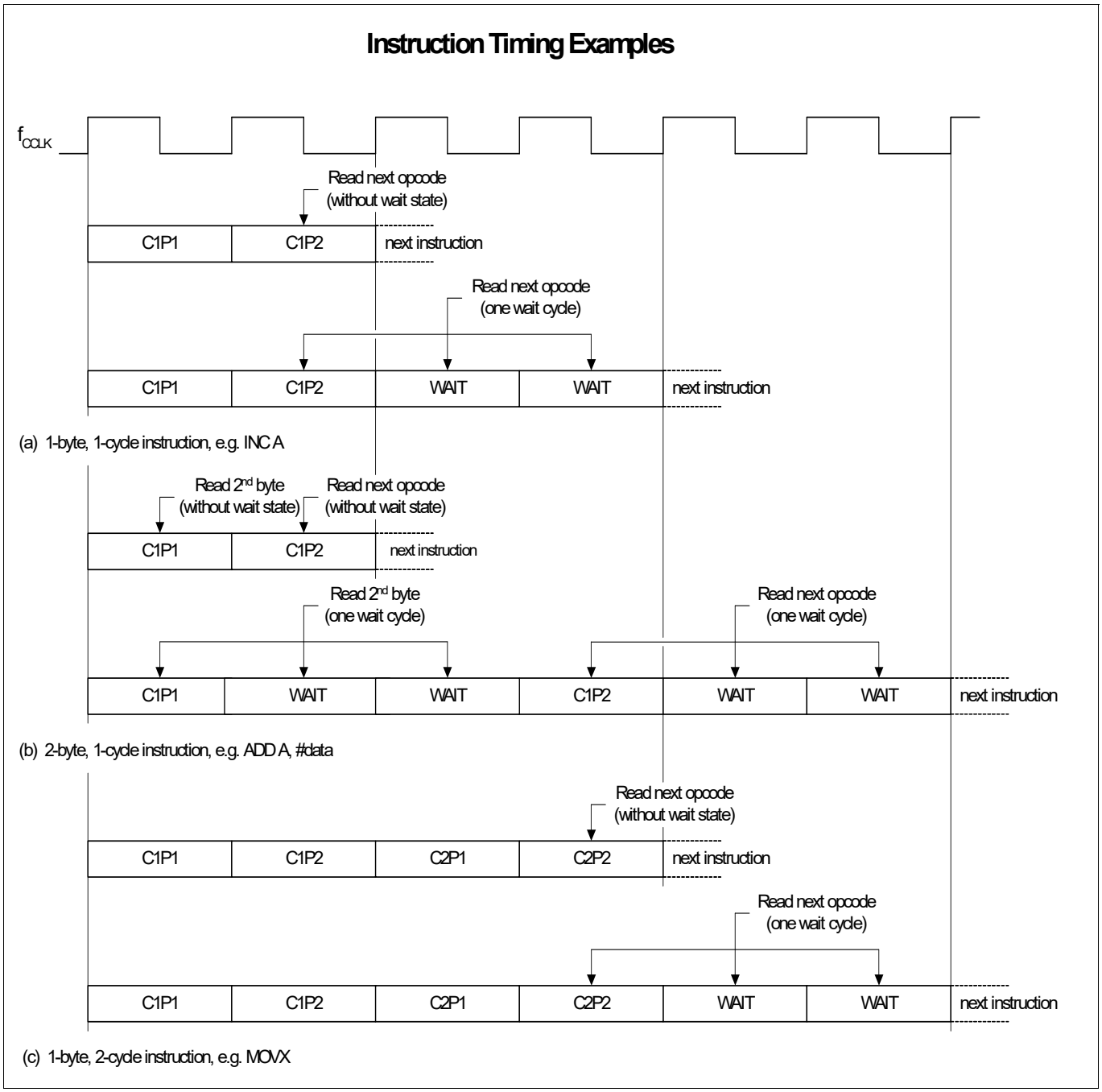


Figure 802 CPU Instruction Timing

The time taken for each instruction includes:

- decoding/executing the fetched opcode
- fetching the operand/s (for instructions > 1 byte)
- fetching the first byte (opcode) of the next instruction (due to CPU pipeline)

Note: The XC800 CPU fetches the opcode of the next instruction while executing the current instruction.

Table 550 lists all the instructions supported by the XC800 CPU. Instructions are 1, 2 or 3 bytes long as indicted in the 'Bytes' column. Each instruction takes 1, 2 or 4 CPU cycles to execute (with no wait cycle). The table gives two values for the number of CPU cycles required by each instruction. The first value applies to fetching operand/s and opcode from fast memory (e.g. Boot ROM and XRAM) without wait state. The second value applies to fetching operand/s and opcode (and in some cases accessing data) from slow memory (e.g. Flash) with wait cycles inserted due to memory requiring one/two wait state(s). One CPU cycle comprises two CCLK clock cycles.

Table 550 Instruction Table

Mnemonic	Hex Code	Bytes	CPU Cycles (no wait state)	Cpu Cycles (one wait state[1)])
ARITHMETIC				
ADD A,Rn	28-2F	1	1	2
ADD A,dir	25	2	1	3
ADD A,@Ri	26-27	1	1	2
ADD A,#data	24	2	1	3
ADDC A,Rn	38-3F	1	1	2
ADDC A,dir	35	2	1	3
ADDC A,@Ri	36-37	1	1	2
ADDC A,#data	34	2	1	3
SUBB A,Rn	98-9F	1	1	2
SUBB A,dir	95	2	1	3
SUBB A,@Ri	96-97	1	1	2
SUBB A,#data	94	2	1	3
INC A	04	1	1	2
INC Rn	08-0F	1	1	2
INC dir	05	2	1	3
INC @Ri	06-07	1	1	2
DEC A	14	1	1	2
DEC Rn	18-1F	1	1	2
DEC dir	15	2	1	3
DEC @Ri	16-17	1	1	2
INC DPTR	A3	1	2	2
MUL AB	A4	1	4	4
DIV AB	84	1	4	4
DA A	D4	1	1	2
LOGICAL				
ANL A,Rn	58-5F	1	1	2
ANL A,dir	55	2	1	3
ANL A,@Ri	56-57	1	1	2
ANL A,#data	54	2	1	3
ANL dir,A	52	2	1	3

Table 550 Instruction Table (cont'd)

Mnemonic	Hex Code	Bytes	CPU Cycles (no wait state)	Cpu Cycles (one wait state[1)])
ANL dir,#data	53	3	2	5
ORL A,Rn	48-4F	1	1	2
ORL A,dir	45	2	1	3
ORL A,@Ri	46-47	1	1	2
ORL A,#data	44	2	1	3
ORL dir,A	42	2	1	3
ORL dir,#data	43	3	2	5
XRL A,Rn	68-6F	1	1	2
XRL A,dir	65	2	1	3
XRL A,@Ri	66-67	1	1	2
XRL A,#data	64	2	1	3
XRL dir,A	62	2	1	3
XRL dir,#data	63	3	2	5
CLR A	E4	1	1	2
CPL A	F4	1	1	2
SWAP A	C4	1	1	2
RL A	23	1	1	2
RLC A	33	1	1	2
RR A	03	1	1	2
RRC A	13	1	1	2
DATA TRANSFER				
MOV A,Rn	E8-EF	1	1	2
MOV A,dir	E5	2	1	3
MOV A,@Ri	E6-E7	1	1	2
MOV A,#data	74	2	1	3
MOV Rn,A	F8-FF	1	1	2
MOV Rn,dir	A8-AF	2	2	4
MOV Rn,#data	78-7F	2	1	3
MOV dir,A	F5	2	1	3
MOV dir,Rn	88-8F	2	2	4
MOV dir,dir	85	3	2	5
MOV dir,@Ri	86-87	2	2	4
MOV dir,#data	75	3	2	5
MOV @Ri,A	F6-F7	1	1	2
MOV @Ri,dir	A6-A7	2	2	4
MOV @Ri,#data	76-77	2	1	3
MOV DPTR,#data	90	3	2	5

Table 550 Instruction Table (cont'd)

Mnemonic	Hex Code	Bytes	CPU Cycles (no wait state)	Cpu Cycles (one wait state[1])
MOVC A,@A+DPTR	93	1	2	3 or 4[2]
MOVC A,@A+PC	83	1	2	3 or 4[2]
MOVX A,@Ri	E2-E3	1	2	3
MOVX A,@DPTR	E0	1	2	3
MOVX @Ri,A	F2-F3	1	2	3
MOVX @DPTR,A	F0	1	2	3
PUSH dir	C0	2	2	4
POP dir	D0	2	2	4
XCH A,Rn	C8-CF	1	1	2
XCH A,dir	C5	2	1	3
XCH A,@Ri	C6-C7	1	1	2
XCHD A,@Ri	D6-D7	1	1	2
BOOLEAN				
CLR C	C3	1	1	2
CLR bit	C2	2	1	3
SETB C	D3	1	1	2
SETB bit	D2	2	1	3
CPL C	B3	1	1	2
CPL bit	B2	2	1	3
ANL C,bit	82	2	2	4
ANL C,/bit	B0	2	2	4
ORL C,bit	72	2	2	4
ORL C,/bit	A0	2	2	4
MOV C,bit	A2	2	1	3
MOV bit,C	92	2	2	4
BRANCHING[3]				
ACALL addr11	11->F1	2	2	4
LCALL addr16	12	3	2	5
RET	22	1	2	2 or 3[2]
RETI	32	1	2	2 or 3[2]
AJMP addr 11	01->E1	2	2	4
LJMP addr 16	02	3	2	5
SJMP rel	80	2	2	4
JC rel	40	2	2	4
JNC rel	50	2	2	4
JB bit,rel	20	3	2	5
JNB bit,rel	30	3	2	5

Table 550 Instruction Table (cont'd)

Mnemonic	Hex Code	Bytes	CPU Cycles (no wait state)	Cpu Cycles (one wait state[1])
JBC bit,rel	10	3	2	5
JMP @A+DPTR	73	1	2	2 or 3[2]
JZ rel	60	2	2	4
JNZ rel	70	2	2	4
CJNE A,dir,rel	B5	3	2	5
CJNE A,#d,rel	B4	3	2	5
CJNE Rn,#d,rel	B8-BF	3	2	5
CJNE @Ri,#d,rel	B6-B7	3	2	5
DJNZ Rn,rel	D8-DF	2	2	4
DJNZ dir,rel	D5	3	2	5
MISCELLANEOUS				
NOP	00	1	1	2
ADDITIONAL INSTRUCTIONS				
MOVC @(DPTR++),A	A5	1	2	2 or 3[2]
TRAP	A5	1	1	–

1) In case of fetch from slow memory requiring only 1 wait state, no wait cycle may be required per the instruction fetched (normally for opcodes that do not require fetch in the next cycle).
2) Depending on whether the operation is accessing memory with zero or one/two wait state.
3) For branch instructions, the instruction time may vary depending on jump destination.

49.3.4 XRAM Addressing Modes

External Data Memory (XRAM) is accessed using MOVX instructions only. These use either 8-bit or 16-bit indirect address. The DPTR register is used for 16-bit addressing and either register R0 or R1 is used to form the 8-bit address.

49.3.4.1 Access to XRAM Using the DPTR (16-bit addressing Mode)

The external memory can be accessed by two read/write instructions, which use the 16-bit DPTR for indirect addressing. These instructions are:

- MOVX A, @DPTR (Read)
- MOVX @DPTR, A (Write)

49.3.4.2 Access to XRAM Using the Register R0/R1 (8-bit addressing Mode)

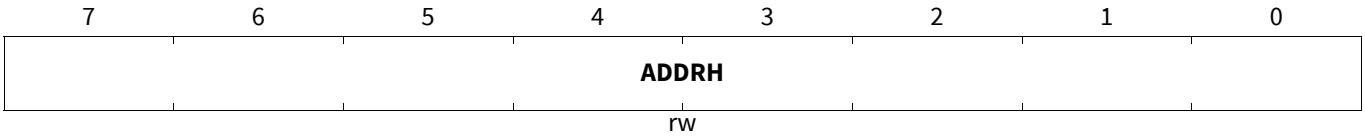
The SCR provides also instructions for accesses to XRAM using the 8-bit address (indirect addressing with registers R0 or R1). These instructions are:

- MOVX A, @Ri (Read)
- MOVX @Ri, A (Write)

To access the XRAM properly, higher order address must be put in SFR XADDRH . And this write instruction must be preceding the MOVX instructions.

On-Chip XRAM Address Higher Order

XADDRH
On-Chip XRAM Address Higher Order (087_H) Reset Value: Table 551
RMAP: X, PAGE: X



Field	Bits	Type	Description
ADDRH	7:0	rw	**Higher Order of On-chip XRAM Address** The range for SCR is from 00_H.

Table 551 Reset Values of XADDRH

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.3.5 Revision History

Table 552 Revision History

Reference	Change to Previous Version	Change Request
V2.4		
	First Official Release of completely reworked SCR chapter	
	No change	

49.4 Memory Organization

The SCR CPU operates in the following four address spaces:

- 2 Kbyte of Boot ROM program memory, ECC protected.
- 256 byte of internal RAM data memory, ECC protected.
- 8 Kbyte of XRAM memory (32-bit memory, accessible to TriCore via the System Peripheral Bus):
  - ECC protected;
  - XRAM can be read/written as program memory or external data memory;
- a 128-byte Special Function Register area.

Figure 803 illustrates the memory address spaces of the SCR.

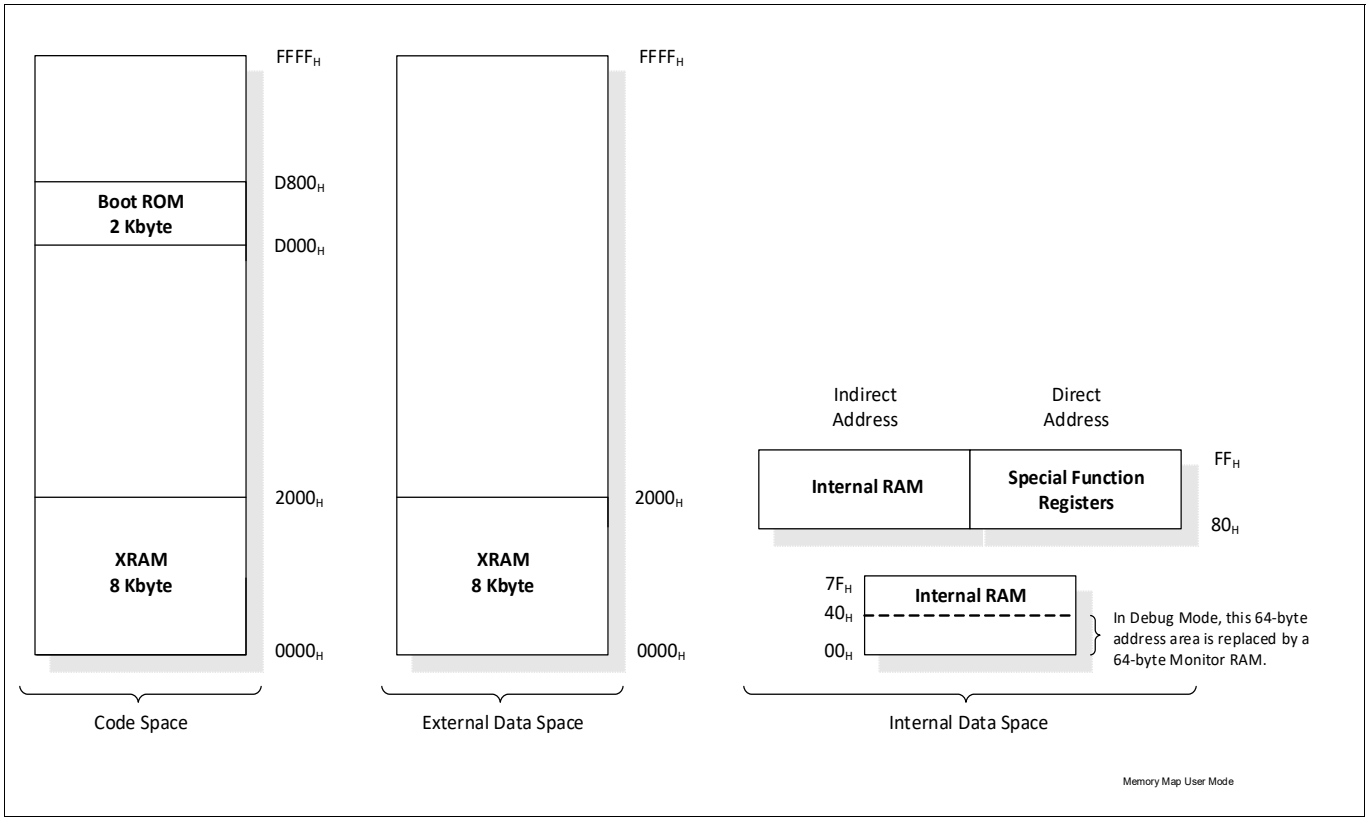


Figure 803 Memory Map of the SCR

49.4.1 Program Memory

The code space is theoretically 64 Kbytes. However, only access to defined program memory (as shown in memory map figure) is supported. For the MemOrg, defined code space is occupied by on-chip memories.

49.4.2 Data Memory

The data space consists of an internal and external data space. Access to internal and external data space are distinguished by different sets of instruction opcodes. In the SCR, the on-chip XRAM is located in external data space and accessed by MOVX instructions. SCR does not support access to external (off-chip) memory. Internal data space is occupied by Internal RAM (IRAM) and Special Function Registers (SFRs), distinguished by direct or indirect addressing.

49.4.2.1 Internal Data Memory

The internal data memory is divided into two physically separate and distinct blocks: the 256-byte RAM and the 128-byte Special Function Register (SFR) area. While the upper 128 bytes of RAM and the SFR area share the same address locations, they are accessed through different addressing modes. The lower 128 bytes of RAM can be accessed through either direct or register indirect addressing, while the upper 128 bytes of RAM can be accessed through register indirect addressing only. The SFRs are accessible through direct addressing.

The 16 bytes of RAM that occupy addresses from 20_H to $2F_H$ are bit-addressable. RAM occupying direct addresses from 30_H to $7F_H$ can be used as scratch pad registers or used for the stack.

49.4.2.2 External Data Memory

The 8-Kbyte XRAM is mapped to both the external data memory area and the program memory area. The SCR XRAM can be accessed by both the SCR's XC800 core and the main Tricore CPU. The arbiter module determines which CPU has control of the XRAM at any one time. The Tricore CPU accesses the XRAM via the peripheral bus. An FPI slave interface is provided to the SCR to interface to the SPB bus. The arbiter module is responsible for arbitrating between the SCR core and the Tricore accesses to the XRAM via the SPB.

SCR core accesses the XRAM using either the 'MOVX' or 'MOVC' instructions. The 'MOVX' instructions for XRAM access use either 8-bit or 16-bit indirect addresses, as shown in **Table 553**.

Table 553 Instructions for External Data Memory Access

Type of Access	16-bit Addressing Mode	8-bit Addressing Mode
Read	MOVX A, @DPTR	MOVX A, @Ri
Write	MOVX @DPTR, A	MOVX @Ri, A

While the DPTR register is used for 16-bit addressing, either register R0 or R1 is used to form the 8-bit address. The upper byte of the XRAM address during execution of the 8-bit accesses is defined by the value stored in register XADDRH. Hence, the write instruction for setting the higher order XRAM address in register XADDRH must precede the 'MOVX' instruction.

A write access to a undefined external data memory address has no effect while a read access returns undefined data.

XADDRH is specified in the core part of the manual.

49.4.3 Special Function Registers

The Special Function Registers (SFRs) occupy direct internal data memory space in the range 80_H to FF_H. All registers, except the program counter, reside in the SFR area. The SFRs include pointers and registers that provide an interface between the CPU and the on-chip peripherals. As the 128-SFR range is less than the total number of registers required, address extension mechanisms are required to increase the number of addressable SFRs. The address extension mechanisms include:

- Mapping
- Paging

49.4.3.1 Address Extension by Mapping

Address extension is performed at the system level by mapping. The SFR area is extended into two portions: the standard (non-mapped) SFR area and the mapped SFR area. Each portion supports the same address range 80_H to FF_H, bringing the number of addressable SFRs to 256. The extended address range is not directly controlled by the CPU instruction itself, but is derived from bit RMAP in the system control register SYSCON0 at address $8F_H$. To access SFRs in the mapped area, bit RMAP in SFR SYSCON0 must be set. However, the SFRs in the standard area can be accessed by clearing bit RMAP. **Figure 804** shows how the SFR area can be selected.

As long as bit RMAP is set, the mapped SFR area can be accessed. This bit is not cleared automatically by hardware. Thus, before standard/mapped registers are accessed, bit RMAP must be cleared/set, respectively, by software.

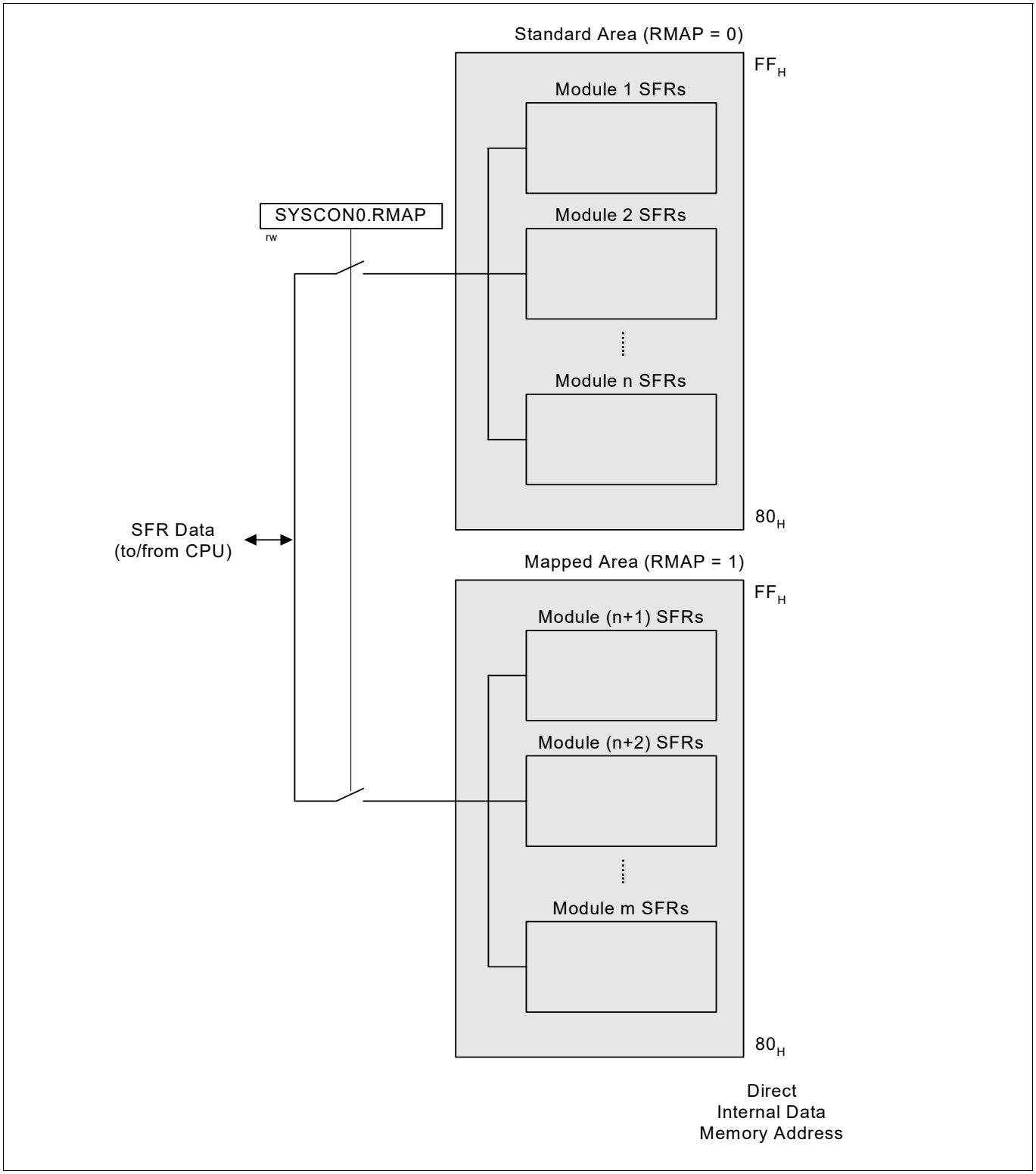


Figure 804 Address Extension by Mapping

49.4.3.2 System Control Register 0

The SYSCON0 register contains the bit to select the SFR mapping.

System Control Register 0

SYSCON0
System Control Register 0 **(088_H)** **Reset Value: Table 554**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
		0			**AMSEL**	**0**	**RMAP**
		r			rw	r	rw

Field	Bits	Type	Description
RMAP	0	rw	**Special Function Register Map Control** 0_B Access to non-mapped (standard) special function register area 1_B Access to mapped special function register area (this setting is not used for applications; therefore, RMAP should be programmed with 0)
AMSEL	2	rw	**Active Memory Map Select** 0_B Active memory map 0: Boot ROM is only mapped from 0xD000 1_B Active memory map 1: Last 512 bytes of Boot ROM are mapped to both, 0x0000 and 0xD600
0	1, 7:3	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 554 Reset Values of SYSCON0

Reset Type	Reset Value	Note
LVD Reset	XXXX X1X0_B	
Generated Reset	---- -1-0_B	

49.4.3.3 Address Extension by Paging

Address extension is further performed at the module level by paging. Certain peripherals have a built-in local address extension mechanism for increasing the number of addressable SFRs. The extended address range is not directly controlled by the CPU instruction itself, but is derived from bit field PAGE in the module page register MOD_PAGE. Hence, the bit field PAGE must be programmed before accessing the SFRs of the target module. Each module may contain a different number of pages and a different number of SFRs per page, depending on the specific requirement. Besides setting the correct RMAP bit value to select the SFR area, the user must also ensure that a valid PAGE is selected to target the desired SFRs.

Figure 805 shows how a page inside the extended address range can be selected.

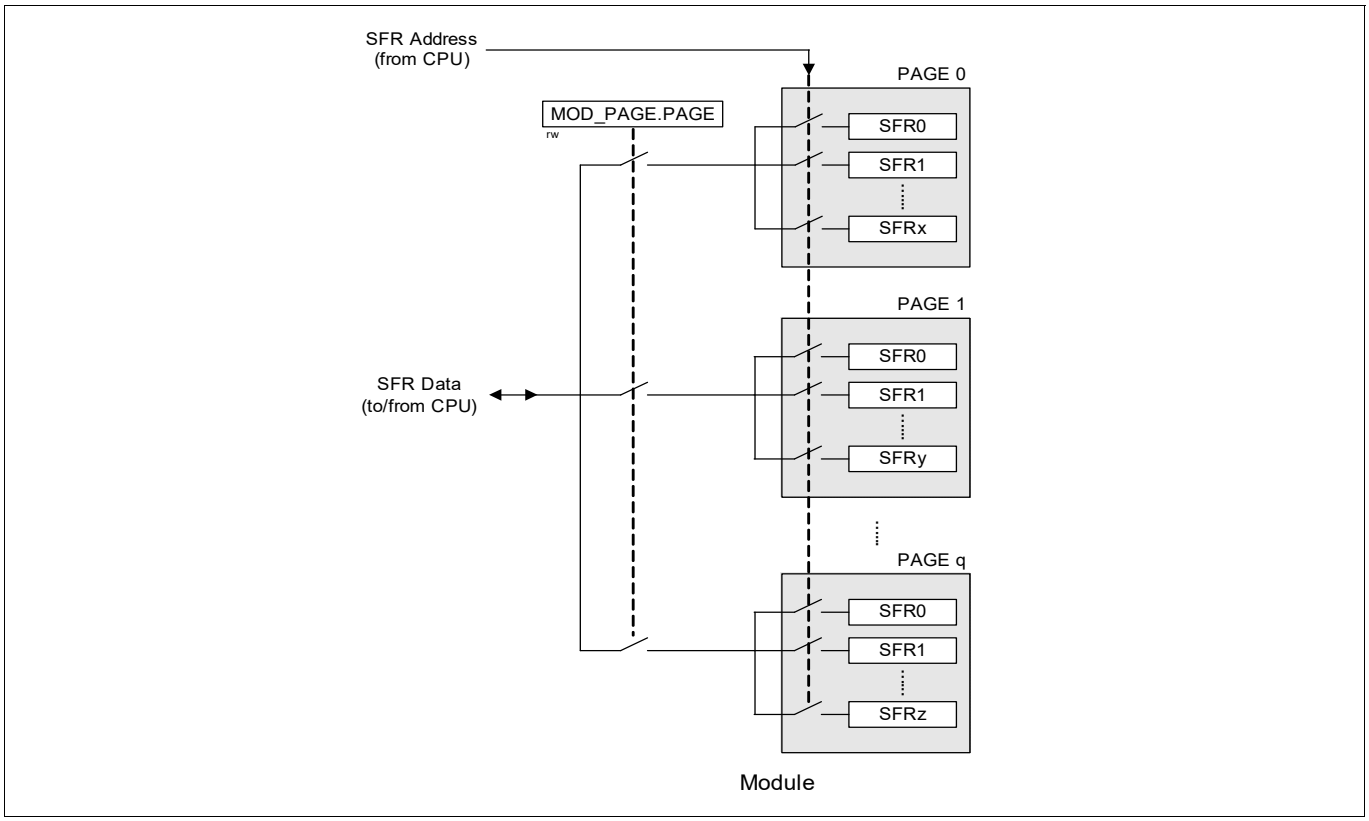


Figure 805 Address Extension by Paging

In order to access a register located in a page other than the current one, the current page must be exited. This is done by reprogramming the bit field PAGE in the page register. Only then can the desired access be performed.

If an interrupt routine is initiated between the page register access and the module register access, and the interrupt needs to access a register located in another page, the current page setting can be saved, the new one programmed, and the old page setting restored. This is possible with the storage fields MOD_STx (x = 0 - 3) for the save and restore action of the current page setting. By indicating which storage bit field should be used in parallel with the new page value, a single write operation can:

- Save the contents of PAGE in MOD_STx before overwriting with the new value
  (this is done at the beginning of the interrupt routine to save the current page setting and program the new page number); or
- Overwrite the contents of PAGE with the contents of MOD_STx, ignoring the value written to the bit positions of PAGE
  (this is done at the end of the interrupt routine to restore the previous page setting before the interrupt occurred)

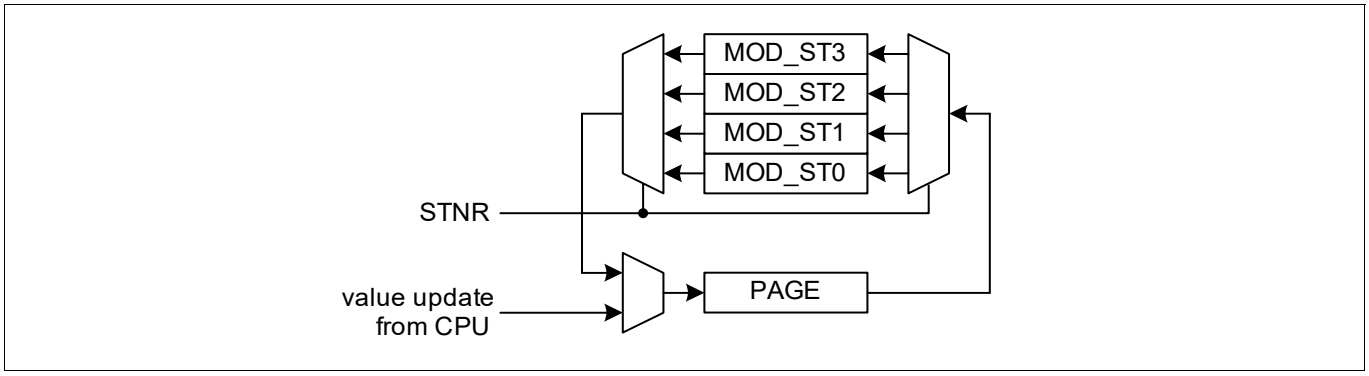


Figure 806 Storage Elements for Paging

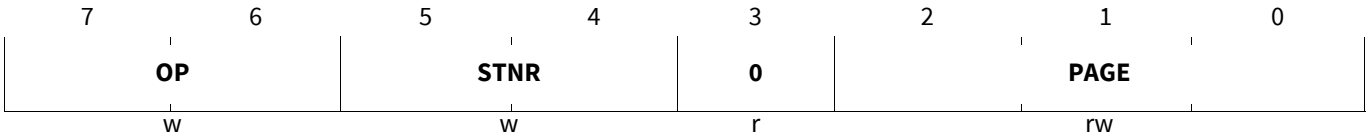
With this mechanism, a certain number of interrupt routines (or other routines) can perform page changes without reading and storing the previously used page information. The use of only write operations makes the system simpler and faster. Consequently, this mechanism significantly improves the performance of short interrupt routines.

49.4.3.4 Page Register

The page register has the following definition:

Page Register for module MOD

MOD_PAGE
Page Register for module MOD **Reset Value: 00_H**



Field	Bits	Type	Description
PAGE	[2:0]	rw	**Page Bits** When written, the value indicates the new page. When read, the value indicates the currently active page.
STNR	[5:4]	w	**Storage Number** This number indicates which storage bit field is the target of the operation defined by bit field OP. If OP = 10_B, the contents of PAGE are saved in MOD_STx before being overwritten with the new value. If OP = 11_B, the contents of PAGE are overwritten by the contents of MOD_STx. The value written to the bit positions of PAGE is ignored. 00_B MOD_ST0 is selected. 01_B MOD_ST1 is selected. 10_B MOD_ST2 is selected. 11_B MOD_ST3 is selected.
OP	[7:6]	w	**Operation** $0X_B$ Manual page mode. The value of STNR is ignored and PAGE is directly written. 10_B New page programming with automatic page saving. The value written to the bit positions of PAGE is stored. In parallel, the previous contents of PAGE are saved in the storage bit field STx indicated by STNR. 11_B Automatic restore page action. The value written to the bit positions PAGE is ignored and instead, PAGE is overwritten by the contents of the storage bit field STx indicated by STNR.
0	3	r	**Reserved** Returns 0 if read; shall be written with 0.

49.4.3.5 Bit-Addressing

SFRs that have addresses in the form of $1XXXX000_B$ (e.g., 80_H, 88_H, 90_H,..., $F0_H$, $F8_H$) are bit addressable.

49.4.3.6 Bit Protection Scheme

The bit protection scheme prevents direct software writing of selected bits (i.e., protected bits) using the PASSWD register. When the bit field MODE is 11_B, writing 10011_B to the bit field PASS opens access to writing of all protected bits, and writing 10101_B to the bit field PASS closes access to writing of all protected bits. In both cases, the value of the bit field MODE is not changed even if PASSWD register is written with 98H or A8H. It can only be changed when bit field PASS is written with 11000_B, for example, writing C0H to PASSWD register disables the bit protection scheme.

Note that access is opened for maximum 34 CCLKs if the “close access” password is not written. If “open access” password is written again before the end of 34 CCLK cycles, there will be a recount of 34 CCLK cycles.

Password Register

PASSWD
Password Register **(086_H)** **Reset Value: Table 555**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
PASS					PROTECT_S	MODE	
w					rh	rw	

Field	Bits	Type	Description
MODE	1:0	rw	**Bit-Protection Scheme Control Bit** These two bits cannot be written directly. To change the value between 11_B and 00_B, the bitfield PASS must be written with 11000_B, only then the MODE[1:0] will be registered. 00_B **SCHEME_DIS**, Scheme Disabled 11_B **SCHEME_EN3**, Scheme Enabled **others**, Reserved; do not use
PROTECT_S	2	rh	**Bit-Protection Signal Status Bit** This bit shows the status of the protection. 0_B Software is able to write to all protected bits 1_B Software is unable to write to any protected bits (default)
PASS	7:3	w	**Password Bits** The Bit-Protection Scheme only recognizes three patterns. 13_H **PROT_OPEN**, Opens access to writing of all protected bits 15_H **PROT_CLOSE**, Closes access to writing of all protected bits 18_H **MODE_EN**, Enables writing of the bit field MODE

Table 555 Reset Values of PASSWD

Reset Type	Reset Value	Note
LVD Reset	07_H	
Generated Reset	07_H	

49.4.4 Memory Control Unit

The Memory Control Unit (MCU Module) is the interface between the CPU and the memories, providing three functions to the MemOrg:

- Active memory map selection
- Allows one physical memory (i.e. XRAM) to be accessed as both program and data memory at the same time
- Memory protection

49.4.4.1 Memory Protection Unit

The SCR consists of the following memory blocks:

- Boot ROM
- XRAM

The target of the memory protection scheme is to prevent unauthorized read out of critical data from the Boot ROM.

The memory protection scheme is hardware-based and is used to protect the Boot ROM;MOVC read instructions executed out of the “unsafe” program memory address range (e.g. XRAM) that target the Boot ROM are blocked when protection is enabled.

49.4.5 Arbiter

The arbiter serves the following purposes:

- Provides access to the XRAM via the SoC System Peripheral Bus (SPB), using an FPI slave interface.
- Arbitrates the XRAM access between SCR core and Tricore CPU (access via the peripheral bus). SCR is given higher priority to access the XRAM.

49.4.5.1 Arbiter Logic

The Arbiter is clocked by SCR f_{sys} clock (i.e., either undivided 100MHz clock, or the 70kHz clock).

The arbiter always grants the SCR higher priority when accessing the XRAM. During a read from the XC800 core, the output from the XRAM is latched until the value is used by the XC800 core. During a write, the data is simply written to the XRAM at the next f_{sys} clock edge.

49.4.6 Revision History

Table 556 Revision History

Reference	Change to Previous Version	
V3.2		
	First Official Release of completely reworked SCR chapter	
	No change	

49.5 SCR Firmware

This chapter describes the BootROM firmware for the SCR. It contains the following sections:

- BootROM Overview
- Startup Procedure
- On-Chip Debug Support (OCDS)

49.5.1 Overview

The SCR BootROM firmware provides the following features:

- Startup procedure for stable operation of the SCR
- Basic debugging capability via OCDS mode

Startup procedure initializes the SCR to a known state to ensure stable operation. The startup procedure consists of all the settings and initialization to be done after SCR reset, before branching to various boot modes. On-Chip Debug Support provides basic debugging capability to the user.

Please refer to the relevant sections for more details.

49.5.1.1 BootROM location

The size of the BootROM is 2 Kbyte, and it operates in two different memory maps:

- In Active Memory Map 1 (AMSEL=1), BootROM is mapped to two address spaces, 0000_H - $01FF_H$ and $D600_H$ - $D7FF_H$
- In Active Memory Map 0 (AMSEL=0), BootROM is mapped to address space $D000_H$ -$D7FF_H$

Upon reset, the active memory map is 1 and the BootROM code is executed from address 0000_H. At the beginning of the BootROM code, a jump instruction to address $D60x_H$ is executed, followed by an instruction to clear the AMSEL bit. This is to switch the active memory map to 0. All other BootROM instructions will be executed in active memory map 0.

49.5.1.2 Boot Mode Option

Entry to various boot modes such as User Mode 0, User Mode 1 and debug mode (OCDS Mode) are done in the startup code of the BootROM. The startup code will depend on the SCR configuration value in the SFR SCR_HWCFG, to enter each mode. **Table 557** shows the Boot Modes configuration value.

Table 557 SCR Boot Mode

Boot Mode[1)]	SCRCFG Value
User Mode 0, XRAM is not programmed	00_H
User Mode 1, XRAM is programmed and will be executed	01_H
OCDS Mode, SCR DAP interface	02_H - 03_H
OCDS Mode, SCR SPD interface	04_H- 07_H
OCDS Mode, SOC DAP interface	$0A_H$- $0B_H$
OCDS Mode, SOC SPD interface	$0C_H$- $0F_H$

1) SPD : Single Pin DAP; DPD: Dual Pin DAP

49.5.2 OCDS Mode

This section describes the On Chip Debug System (OCDS) module and the interaction with the Monitor ROM and external debugger.The OCDS provides basic functionality to support development without the need to purchase expensive In Circuit Emulator (ICE) device which is the traditional way of debugging the embedded systems. The interface that can used to support this function is DAP.

The TRAP_EN bit in Extended Operation register (EO.4) need to enable for OCDS functionality to work properly.

When entering OCDS mode, the firmware configures the debug ports.

49.5.2.1 Features

List of features supported:

- Set breakpoints on instruction address and within a specified range
- Set breakpoints on internal RAM address for read and write
- Support unlimited software breakpoint in RAM code region
- Processing External breaks
- Stepping through the program code

In summary, up to four hardware breakpoints can be set for the current implementation. It can be used for detection specific Instruction Pointer (IP), a range of IP and specific range of read/write of internal RAM, unlimited software breakpoint (only possible in writable region) and external break. When any of the above is encountered, that corresponds to a debug event which will then trigger the execution of Monitor ROM.

49.5.2.2 OCDS Basic Understanding

For a complete understanding of the OCDS concept, its description can be found in the OCDS chapter.

49.5.2.2.1 Introduction of the XC800 Core Debug mode

The Program Counter will freeze when the core entered the debug mode and it will enter this mode when the TRAP instruction is executed and with the TRAP_EN = '1'. It will also enter upon DebugReq activation for IRAM and External Breaks. For hardware breakpoint, the TRAP instruction is actually supplied by the Monitor Mode Control (MMC) hardware block.

49.5.2.2.2 Minimum hardware and overhead are added

OCDS hardware consists of the MMC, additional internal RAM and JTAG and additional ROM memory.

The OCDS is built on top of debug mode together with minimum hardware overhead, Monitor ROM and additional 64 bytes of RAM to perform the breakpoint debugging functionality.

Monitor is resident in the BootROM region and it will make use of the additional 64 bytes of internal RAM termed Monitor RAM and together with the hardware to realize the OCDS functionality. The Monitor RAM which occupied the same memory location as the standard 8051 internal RAM, and it is selected through the MRAMS bit (MMCR).

49.5.2.2.3 Implementation of OCDS with the debug mode of XC800 Core

Some combination logic together with the support of the Monitor ROM to coordinate the processes. In this document, the tasks of the Monitor ROM will be explained in details in the next section.

49.5.2.3 OCDS Monitor Firmware

This section will describe the Monitor firmware interaction between the Monitor ROM with the OCDS hardware and the debugger software in host PC.

49.5.2.3.1 OCDS mode entry

OCDS mode is entered if:

- OCDS Boot option for SCR DAP interface, SFR SCR_HWCFG = 02_H - 03_H;
- OCDS Boot option for SCR SPD interface, SFR SCR_HWCFG = 04_H - 07_H;
- OCDS Boot option for SOC DAP interface, SFR SCR_HWCFG = $0A_H$ - $0B_H$;
- OCDS Boot option for SOC SPD interface, SFR SCR_HWCFG = $0C_H$ - $0F_H$;

IP hardware breakpoint at XRAM address 0x0000 is set before it exits OCDS mode.

Note: Due to a sideeffect, the first instruction in Wait for Debug Event needs to be HMONITOR, otherwise debugging will not work.

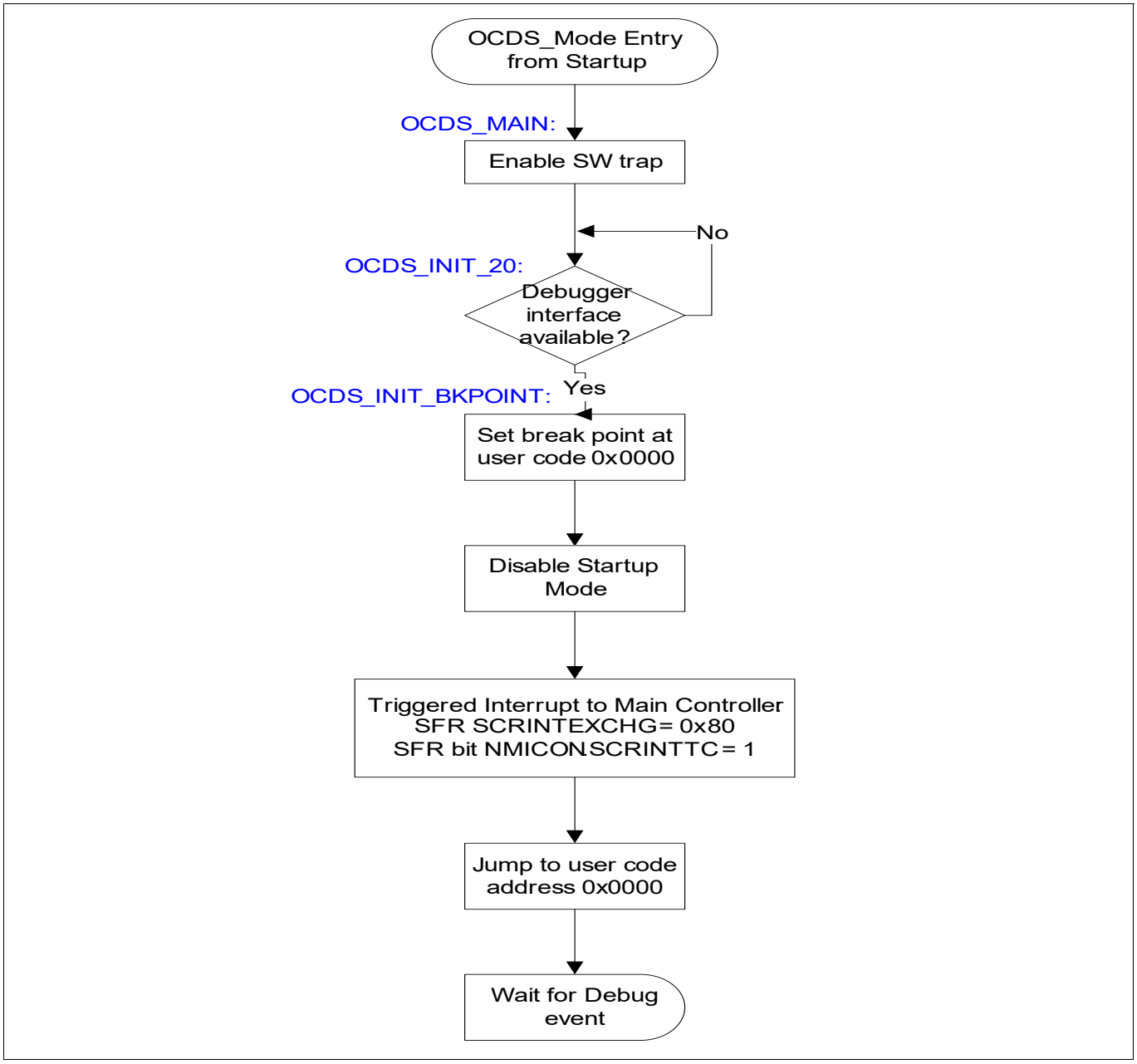


Figure 807 OCDS Initialisation

49.5.2.3.2 Communication between the Debugger and Monitor

The debugger will be the master and it will control all the read and write transactions. A virtual memory map is defined as followed.

Virtual Address Translation

To facilitate the debugger to get an insight of the memory map and all the registers in the system, a set of virtual address is defined to relate them to the physical address of the program memory, external memory and all of the other memories and registers within the system. In this way, the debugger will read or write to the virtual address and the actual reading and writing to the correct memory segment and register are controlled by the Monitor program.

Table 558 Translation of Virtual Memory to Physical Memory

Description	Physical address	Valid Virtual Address for SCR	Virtual Address (2.5 bytes)
Program memory	0000_H - $1FFF_H$	00000_H - $01FFF_H$ (8K)	00000_H - $0FFFF_H$ (64K)
External memory	0000_H - $1FFF_H$	10000_H - $11FFF_H$ (8K)	10000_H - $1FFF_H$ (64K)
Internal memory	00_H - FF_H	20000_H - $200FF_H$ (256)	20000_H - $2FFF_H$ (64K)
Indirectly accessible OCDS register	HWBPxH/L 08_H - $0F_H$	22208_H - $2220F_H$ (8)	22200_H - $222FF_H$ (256)
Monitor Virtual Registers	22300_H - $2233F_H$	22300_H - $2233F_H$ (up to 64 bytes but not all contain the valid data)	22300_H - $223FF_H$
Reserved			30000_H -$3FFFF_H$
SFR registers and page number. (Including standard, mapped and paged)	80_H-FF_H	40000_H - $7FFFF_H$ (Always BIT18=1) BIT17:15 Page number BIT14:8 SFR page address BIT7: SFR standard (RMAP=0), mapped (RMAP=1) BIT6:0 SFR address	40000_H - $7FFF_H$

For the XRAM program memory, the virtual translated address is the same as their address. For external memory and internal IRAM, the upper nibble of the virtual address is 1_H and 2_H respectively.

There are 8 indirect OCDS registers. They are the OCDS hardware breakpoint (HWBPxH/L). These registers are only accessible by setting HWBPSR first to select the relevant registers. To simplify the debugger task of setting breakpoint, the registers are mapped to the virtual address. The monitor program will base on the virtual address and write to these registers appropriately.

Table 559 Indirectly Accessible OCDS Registers

Register name	Virtual translated Address	Description
HWBP0L	22208_H	Set the address for breakpoint 0
HWBP0H	22209_H	
HWBP1L	$2220A_H$	Set the address for breakpoint 1
HWBP1H	$2220B_H$	

Table 559 Indirectly Accessible OCDS Registers (cont'd)

Register name	Virtual translated Address	Description
HWBP2L	$2220C_H$	Set the address for breakpoint 2
HWBP2H	$2220D_H$	
HWBP3L	$2220E_H$	Set the address for breakpoint 3
HWBP3H	$2220F_H$	

In order to optimize the amount of virtual address allocated to the SFR. A scheme has been devised to cope with all the SFR type i.e. standard, mapped, paged type.In order to access all the SFR, the following fields are required and defined below.

Table 560 Distribution of the Bit Definition

Field	Bit definition	Description
RMAP	BIT7	To indicate which region of SFR to access. '0' - standard registers and '1' - (R)mapped
SFR_ADDRESS	BIT 6:0	SFR to be accessed. The value of the SFR is always 80_H or above. Thus it is represented by 7bit omitting the most significant bit.
SFR_PAGE_ ADDRESS	BIT14:8	For accessing paged SFR, there is a need to write the page number to the SFR PAGE address first. Once again, it is represented by 7 bit. For non-paged SFR, this field must be written with 00_H.
PAGE_NUMBER	BIT17:15	For non-paged SFR, the PAGE_NUMBER must be set to $0F_H$. For paged SFR, this field can be used to specify page number from 0 to 07_H.
BIT18	BIT18	According to **Chapter 558** always 1.

The standard SFR refers to those SFR that required SYSCON0.RMAP=0 before they can be accessed and the mapped SFR refer to those SFR that can be accessed only when SYSCON0.RMAP=1. As for paged SFR, in additional to the standard or mapped properties, they also required the page number to be set, and it is written to the corresponding SFR_PAGE_ADDRESS. If the SFR does not paging, then SFR_PAGE_ADDRESS field must set to zero.

Bit17-Bit15 will represent the page number from 0x0-0x7, if the value equal to 0x07 and it is a non-paged SFR, then it is not necessary to set the SFR_PAGE_ADDRESS and BIT14-8 can be clear to zero.Otherwise, Bit14-8 will indicate the address (7bit only) for that SFR page. Bit7 = 0 will represent standard and 1 if mapped SFR. Bit6-0 will indicate the address (7bit only) for that desired SFR to be read out. The SFR is represented by 7 bit as its address is always above 80H and the MSB is not required and automatically handled by the firmware.

The list of registers that are used in the monitor program need to be preserved, thus to access these few SFR, the corresponding virtual address should be supplied instead of following the format given in **Table 560**

Table 561 Monitor Virtual Registers

Monitor Virtual Register Name	Virtual Translated Address	Description
mons_SCU_PAGE	$2231A_H$	Current SCU_PAGE register
mons_IEN0	$2231B_H$	Current Interrupt Enabled register
mons_DPL	$2231C_H$	Current data pointer low byte
mons_DPH	$2231D_H$	Current data pointer high byte
mons_B	$2231E_H$	Current B register
mons_A	$2231F_H$	Current A register
mons_SYSCON0	22330_H	Current SYSCON0 register

Table 561 Monitor Virtual Registers (cont'd)

Monitor Virtual Register Name	Virtual Translated Address	Description
mons_PSW	22332_H	Program Status Word
mons_RETH	22333_H	Return address high byte
mons_RETL	22334_H	Return address low byte
mons_SP	22335_H	Current Stack Pointer

Protocol Definition between the Debugger and Monitor ROM

There are 5 Monitor processed commands that are defined and described as follows.

Table 562 Monitor Processed Commands

Command	Value	Description
READ_BYTE	$1X_H$	Read a byte.
READ_BLOCK	$2X_H$	Read a block of data and return byte by byte to the host.
WRITE_BYTE	$3X_H$	Write a byte to memory location.
WRITE_BLOCK	$4X_H$	Write a block of data starting from memory location.
WRITE_PROTECTED_SFR	$5X_H$	Write a byte to the password protected bit SFR

Note: In the command field, only the upper nibble will indicate the command type, the lower nibble is used to store the 4 MSB bit of the virtual address and is not part of the command.

Some assumptions are made:

- Instead of 12 bit, it is round up to 16 bit for easy handling. Thus some are dummy bits that are not used but need to be there for data padding purpose to ensure the proper operation.
- Debugger send commands and data to Monitor, the Monitor ROM will acknowledge all bytes that are the command, parameters and data field that have been received. This is to ensure all the bytes from the debugger is received correctly.
- All commands (read or write) will be issued with a unique command_ID, since there are only five commands defined.
- A new command 50_H is introduced to support the password protected bit SFR write operation.

Monitor Processed Commands

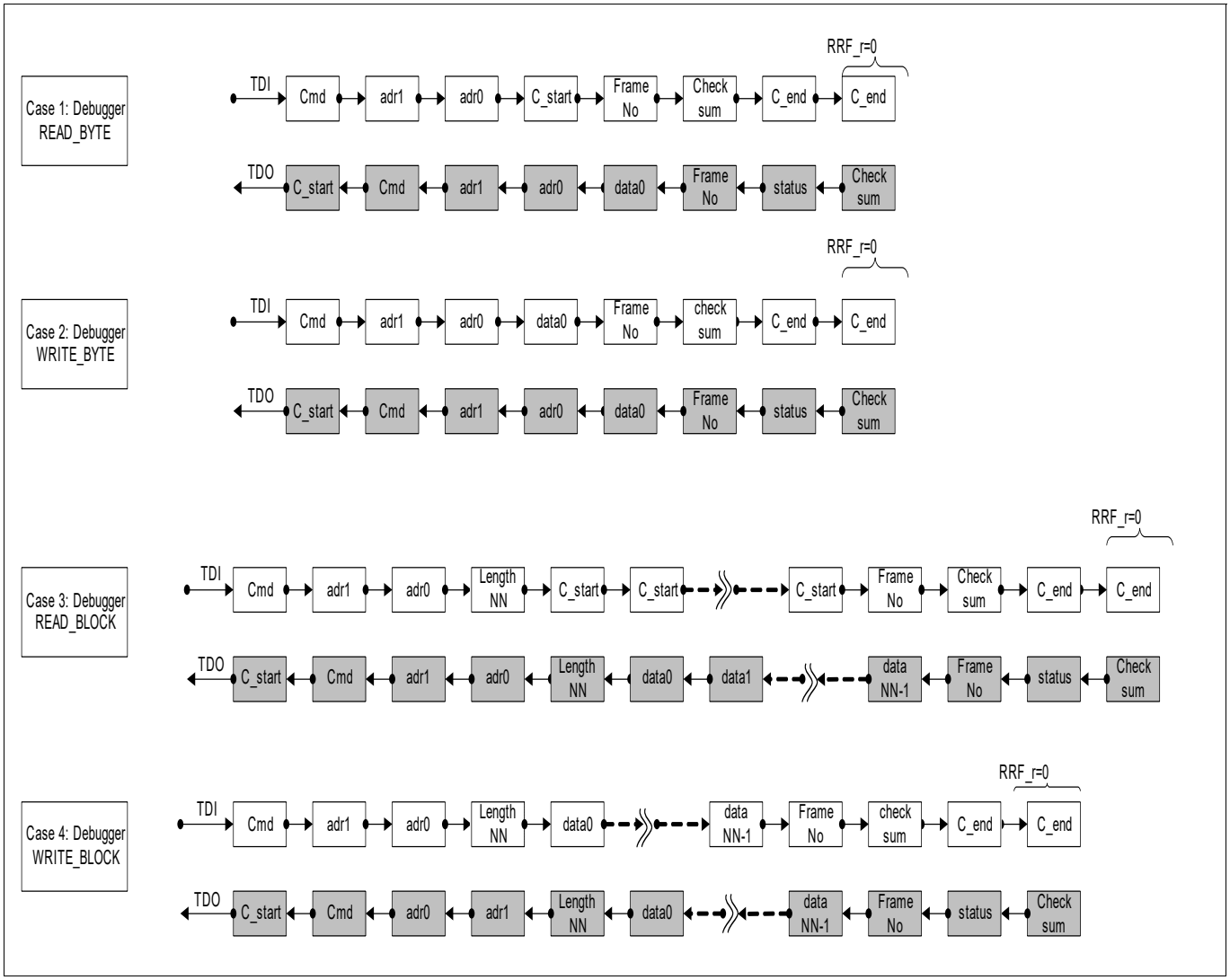


Figure 808 Monitor Processed Commands

The following is a list of the Monitor processed commands that are supported by the Monitor firmware.

Monitor processed command: READ_BYTE ($1X_H$)

The debugger will send 1 byte of command and 2 bytes of address.Next C_START, Frame No and Checksum are sent. Lastly followed by the C_end and C_end with RRF_r =0. The corresponding fields and the status byte are then checked by the debugger.

Monitor processed command: READ_BLOCK ($2X_H$)

The debugger will send 1 byte of command, 2 bytes of address and 1 byte of length. Next the C_START is send one after other until all the data are being transferred.Next the Frame No and Checksum are send followed by the C_end and C_end with RRF_r =0.

Monitor processed command: WRITE_BYTE ($3X_H$)

This command flow is very similar to the READ_BYTE command except that the C_START is not send out by debugger. Instead the data byte is written there.

Monitor processed command: WRITE_BLOCK ($4X_H$)

This command flow is similar to the READ_BLOCK except the C_START is not send out by the debugger. Instead all the data fields are send.Next the Frame No and checksum are also being sent out followed by the C_end and C_end with RRF_r=0.

Refer to the flow chart **Figure 808** that describe the commands flow.

Monitor processed command: WRITE_PROTECTED_SFR ($5X_H$)

This command is added to support writing to the password protected SFR and its functionality is similar to the WRITE_BYTE.

Note: For all the 5 commands being described, the upper 4 bit MSB of the address field is embedded inside the lower nibble of command byte. The Frame No is a 1 byte field wide, it just a sequence number that increment itself and warp around when it is overflows.

Description of the various fields use in the protocol

Various fields are define to establish the communication link between the debugger and Monitor ROM. They are outlined in **Table 563**.

Table 563 Different fields that are defined

Description (byte)	Value	Description
C_START (1)	01_H	To indicate start or ongoing event
C_END (1)	02_H	To indicate ending event
FRAMENO (1)		The debugger will send a 00_H first and sequentially increase it by 1 and wraps around if the value exceeds the single byte length. This value must reset and start from 00_H when Monitor Mode is (re-)entered or when COMRST has been activated.
CHECKSUM (1)		The checksum for the debugger and Monitor side are calculated by XORing all the fields except its checksum.
STATUS(1)		The following bit will set to indicate an error condition. BIT0 - Debugger CHECKSUM and the calculated value does not match. Read/Write operation to the memory is executed. BIT1 - Frame number out of sequence. Read/Write operation to the memory is executed. BIT2 - Control bytes, C_START not received correctly. Read/Write operation to the memory is executed. BIT3 - Invalid Range of the memory location. Read/Write operation to the memory is not executed.
ADR0 (1)		This represent the least significant 8 bit of the virtual address.
ADR1 (1)		This represent the next higher 8 bit of the virtual address.

Table 563 Different fields that are defined (cont'd)

Description (byte)	Value	Description
CMD (1)		The upper nibble contains the command as specified in **Figure 808**. The other nibble represent the most significant 4 bit of the virtual address. Thus together, they made up the 20 bits address.
Length (1)		This field is used in the read block command and write block command. It indicates the number of data bytes to be read or written into the memory. Valid range of this field is from 1 to 255.

JTAG communication and controlling the OCDS hardware control bits

Table 564 list the monitor hardware related commands.

Table 564 Monitor hardware related command

Mode/ Command	Data	Jena_r	RRF_r	MMODE_r	MBCON_r	COMRST_r	Description
HSTATE	XX_H	0	0	0	0	0	Get the status of the OCDS status bits.
HCONFIG	$A5_H$	1	0	0	0	0	Debug interface enabled by MMCR2.JENA = 1
HDEBUG	DATA	0	1	0	0	0	Data read and write
HMONITOR	$A5_H$	0	0	1	0	0	Enter the Monitor mode
HMBCOUT	XX_H	1	0	0	X	0	Jena_r must be set together with the MBCON_r state to control the MBC output. To activate it,MBCON_r=1 and to disable it MBCON_r=0.
HCOMRST	$A5_H$	0	0	0	0	1	Reset the transaction and start with a new command.
HRESET	$5A_H$	1	1	1	1	1	Reset the complete the SCR (main reset) including OCDS, but not DAP.

The prefix with H indicate that it is a OCDS hardware related commands.

The HDEBUG command/mode is use for any data read and write transaction and it is actually being used to support the five Monitor processed commands.

49.5.2.3.3 Activate the Monitor ROM

When a debug event occurred, the debug mode will be entered and instruction will be fetched from MM_START that represent the upper 8 bit of the program code address bus.MMCR.MMODE is set by the hardware to signal that the Monitor mode is entered.

The following need to be taken care. (Extract from OCDS specification)

- Save user Interrupt Enable (IEN0) and disable the global interrupt.
- Save current (user) Stack Pointer (SP) into Monitor RAM.
- Initiate the stack to dedicate area in Monitor RAM
- Save the SYSCON0 into the Monitor RAM, to preserve the user value of RMAP control bit.

- Clear the RMAP bit in SYSCON0.
- Save the current (user) PSW,A,B,DPH and DPL to Monitor IRAM.
- Set RMAP bit in the SYSCON0, to access MMICR.
- Clear the MMICR.DRETR
- Perform a LCALL to MON_START_ST (or MON_NORMAL_ST), so the break address is saved and debug mode is exited and continue to run in Monitor ROM mode.
- NOP instruction is required here for proper functioning.
- Communication between the debugger and Monitor through the selected interface.

Figure 809 shows the flow diagram when the debug event occurred. Debugging a BootROM startup code will cause the device to enter endless loop. A reset of the device is needed to exit the endless loop.

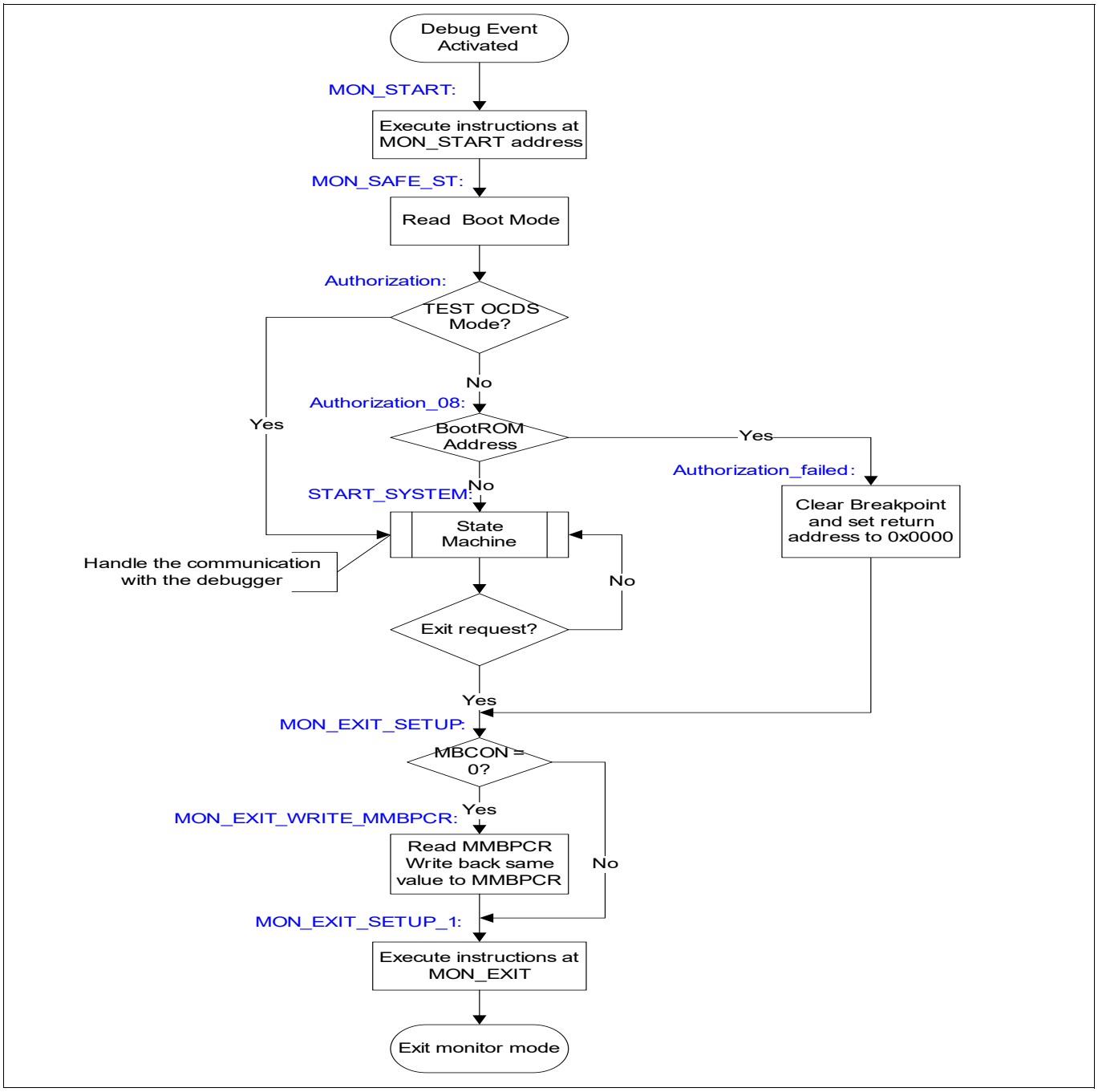
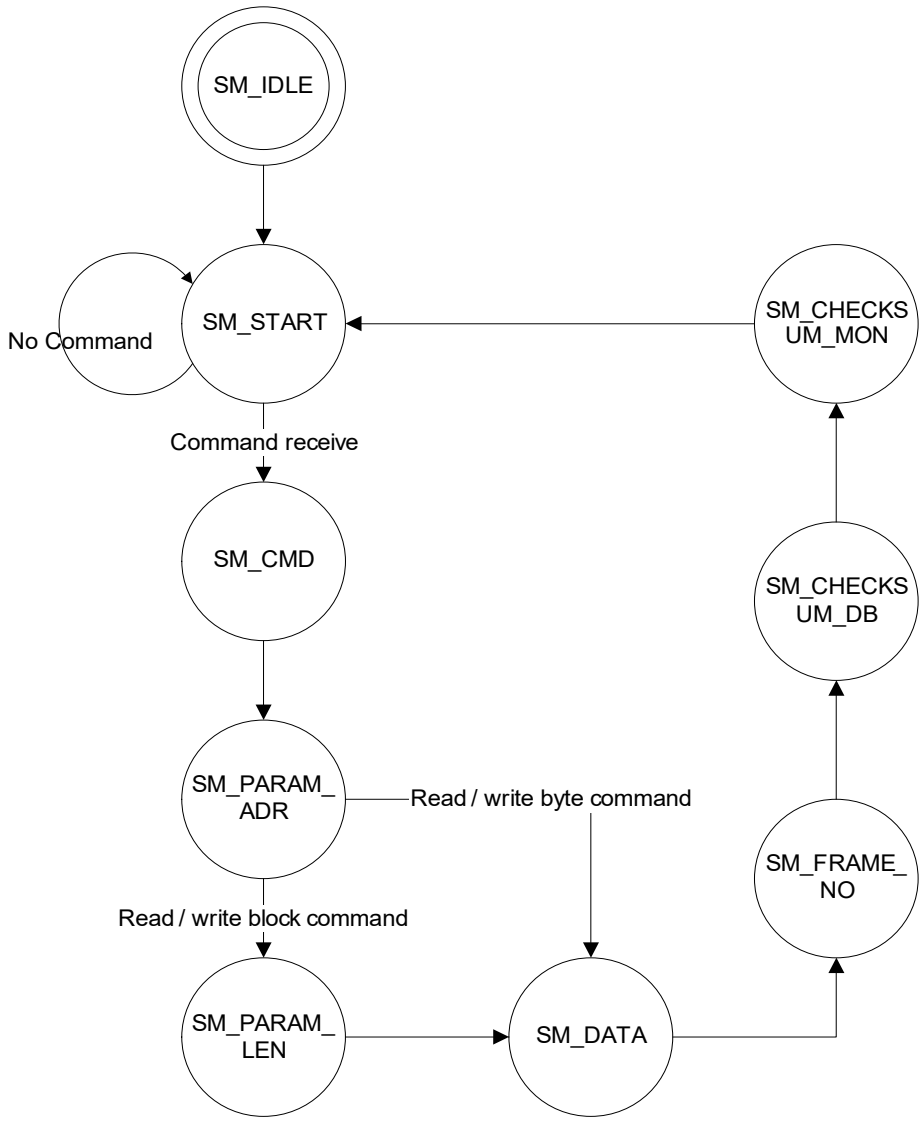


Figure 809 Debug mode Flow Diagram

Figure 810 shows the state-machine flow diagram in the monitor program. The detailed descriptions and flow diagrams of each state can be found in **Chapter 49.5.2.4.8**.



State	Description
SM_IDLE	Idle state
SM_START	Clear variables used in the state machine, waiting for new command
SM_CMD	Receive Command
SM_PARAM_ADR	Receive 20bits virtual address
SM_PARAM_LEN	Receive data length parameter for read/write block commands
SM_DATA	Receive data for write / write block commands, Execute command. For read command, output data to host. For write command, update the data into memory (SFR, XRAM)
SM_FRAME_NO	Keep track on number of frame send by the debugger
SM_CHECKSUM_DB	Calculate the debugger checksum
SM_CHECKSUM_MON	Calculate the monitor checksum

Figure 810 State Machine - To handle monitor processed commands from host

49.5.2.3.4 Exit the Monitor ROM to return to user mode

To exit monitor mode and return to the user code, the following steps are needed to restore the user status and return to the proper user address.

The steps needed to return to user mode.

- After writing to the MMCR.EXIT=1 followed by NOP, this will trigger the debug mode to be entered.
- Mons_SYSCON0 and mons_RETH are temporary stored into the 2 OCDS scratch registers.
- SYSCON.RMAP is cleared and set the SCU_PAGE=0 to restore back the saved mons_SCU_PAGE.
- IEN0,DPH,DPL,B,A,PSW and SP are restored back respectively from the Monitor IRAM.
- The return addresses which are contained in the mons_RETL and MMWR2 are pushed to the stack.
- The SYSCON0 is updated with the value stored in the MMWR1
- RET is performed, i.e jump to address that are stored in the stack which transfer the control back to the user program.
- NOP is followed to ensure proper operation.

49.5.2.3.5 Single step execution

Single step execution:

- Current instruction in the user mode is executed and a new debug event will automatically be triggered and the Monitor mode is entered again.
- The return address is calculated by minus 1 of the break address.
- In the event when other break event happened due to single step or others, then step will always take the higher priority and consider first for the calculation of the return address.

49.5.2.3.6 Software breakpoint implementation

It is only possible if the original opcode can be replaced by the TRAP opcode. And this implies that the code memory must be writable.Thus for the XRAM, the XRAM content must be saved by the debugger and modified with the TRAP opcode $A5_H$. After the break has occurred, the debugger should program back the original user code, so that the normal flow of the program is not affected.

49.5.2.4 Important information for Debugger

This section will describe the additional tasks that are required by the debugger to produce seamless integration and synchronization flow between the monitor firmware. The debugger is the master for all transaction and it must keep track of the flow and always in sync with the current MCU resources changed. The OCDS capabilities will be disabled for BootROM region.

49.5.2.4.1 Specific Infineon DAP instruction to configure DUT

The debugger on the host side act as the master and is responsible for initiating and controlling all JTAG transactions.

From the BootROM perspective, it is only concern with the JTAG communication between the JTAG module and the OCDS module. The communication can begin once the JTAG Instruction Register (IR) is set with **JTAG_IO_INSTRUCTION1 (IOINSTR1)** ($C1_H$). This will effectively connect the external JTAG pins to the OCDS JTAG Interface module thus enabling communication between the debugger and the OCDS module.

49.5.2.4.2 Initial entry to the monitor mode

There is only one possible initial entry into the monitor mode.

Upon a reset, with boot mode configured for OCDS by the SOC, it will enter OCDS mode. The IP break point is configured at address 0000_H and execution begin at that specific address and a break event is generated and monitor mode will be entered.

49.5.2.4.3 Implementation of the debugger

Upon the entry of the monitor, the PC value are saved into the mons_RETH and mons_RETL.

The 16 bytes from $1FE8_H$ to $1FF7_H$ of the XRAM is been used by the Monitor ROM to perform reading and writing the SFR. Thus the contents at these location must be saved by the debugger when the Monitor mode is entered and restored them before exiting the Monitor mode.

IP and IRAM breakpoints are supported and modified by writing to the OCDS registers. Likewise, the debugger can view and modified any SFR according based on the virtual address translation.

For every transaction, a status byte will be returned. A non-zero value will indicate that there is error, refer to the table for the STATUS byte field definition **Table 563**.

49.5.2.4.4 Calculation of the return address upon a break event

The break address is stored in mons_RETH (33_H) and mons_RETL (34_H) upon a break and enter the monitor mode. Thus the debugger must check what type of break has occurred and calculate the return address accordingly.

For IP and IRAM breakpoint, the return address = **break address - 2**. For any other break, the return address = **break address -1**.

The debugger will need to update the return address into the mons_RETH and mons_RETL before exiting the monitor mode. To implement the “Go address” function, the debugger should modify the mons_RETH and mons_RETL directly before exiting.

49.5.2.4.5 Limitation of the monitor mode

Interrupt handling, if an external interrupt has occurred. The monitor will not be able to service it. For IRAM read and write break, only the return address can be calculated but the break address might not be known.

49.5.2.4.6 Exit sequence of the monitor mode

To exit the Monitor mode, the debugger will need to write to the MMCR.MEXIT = 1 together with the MMCR.MEXIT_P = 1. Upon writing to these bits, the Monitor mode is automatically exited. Prior to this, the starting 16 bytes from ($1FE8_H$ - $1FF7_H$) of the XRAM must be restored.

49.5.2.4.7 An example of a Debugging session

Here is an example of a debugging session:

- Initial entry of the monitor mode through OCDS mode.
- The PC is saved to the stack by the monitor firmware. The debugger will buffer the 16bytes of XRAM at address $1FE8_H$ - $1FF7_H$, into its memories.
- Load a user hex file into the XRAM memory.
- Set the IP and IRAM breakpoints.
- Modify the mons_RETH and RETL to the starting address for execution.
- Clear all the breakpoint flag.
- Follow the exit sequence.
- The break will occur and once again the monitor mode is entered and PC is saved.
- Debugger will once again buffer the 16 bytes of data into the its memories.

- All the memories and SFR can be read and modified.
- The return address is calculated based on the break address and the breakpoint flags.
- The return address is updated.
- Restore back the 16 bytes of XRAM content back to its original location.
- Follow the exit sequence.
- This process will repeat as long as there is a break event.

49.5.2.4.8 Detailed description of Monitor Program State Machine

This session described the flow of the major states:

- Start state - start of the state machine
- Command state - Received OCDS command and the upper 4bits of the virtual address.
- Parameter Address state - Received the lower 16-bits virtual address.
- Parameter Data Length state - Received the frame Length
- Data state - For read command, this state will perform the read commands and send out the data. For write command, this state will receive the data and perform the write operation to the respective memory.

Start and Command States

Figure 811 shows the flow of start state and command state. The lower nibble of the command byte contains the uppermost 4 bits of the virtual address.

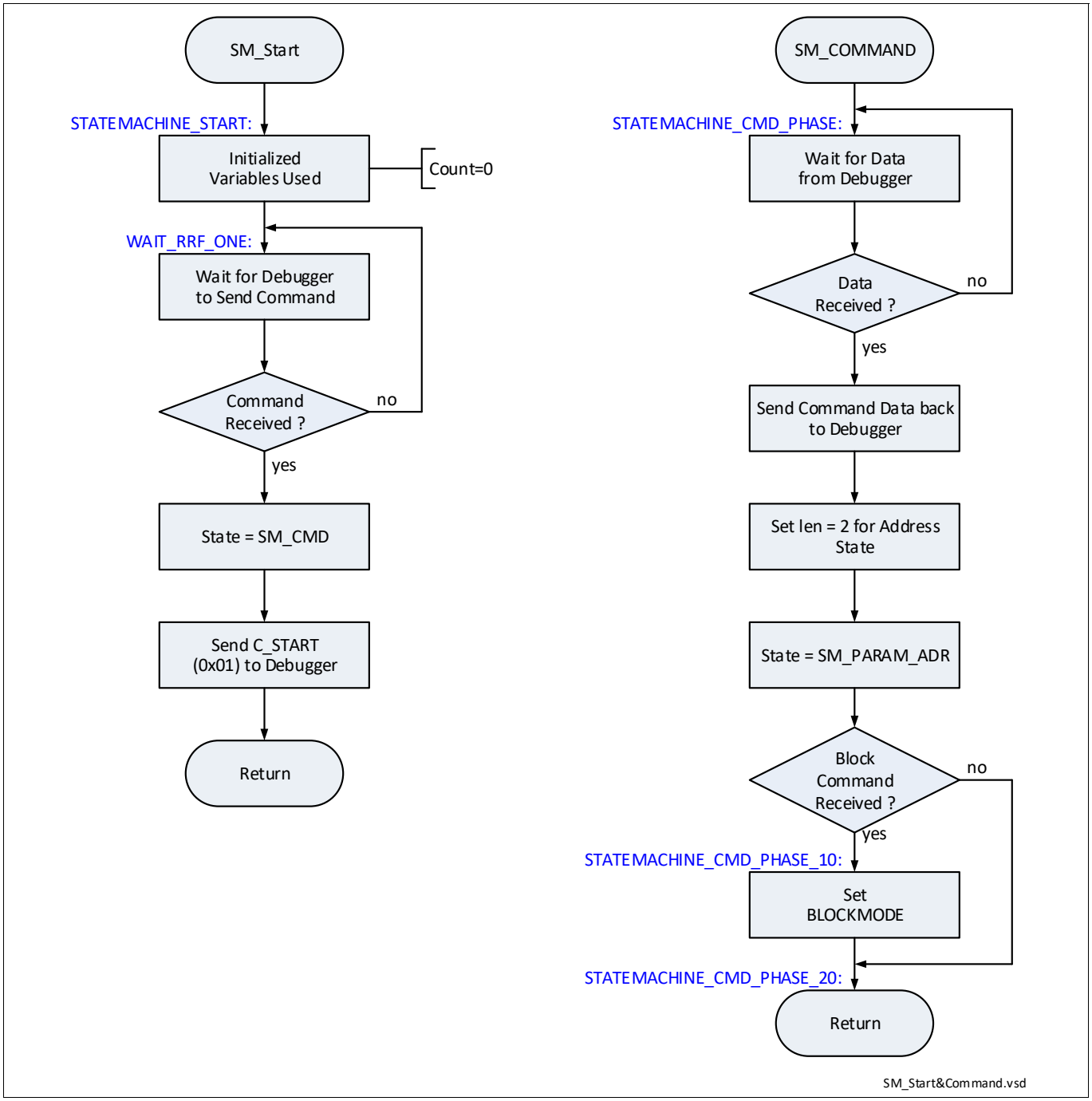


Figure 811 Start State and Command State

Parameter Address State

In this state, the next 16 bits of the virtual address are received. The 20-bits virtual address indicates which memory or SFR is to be accessed.

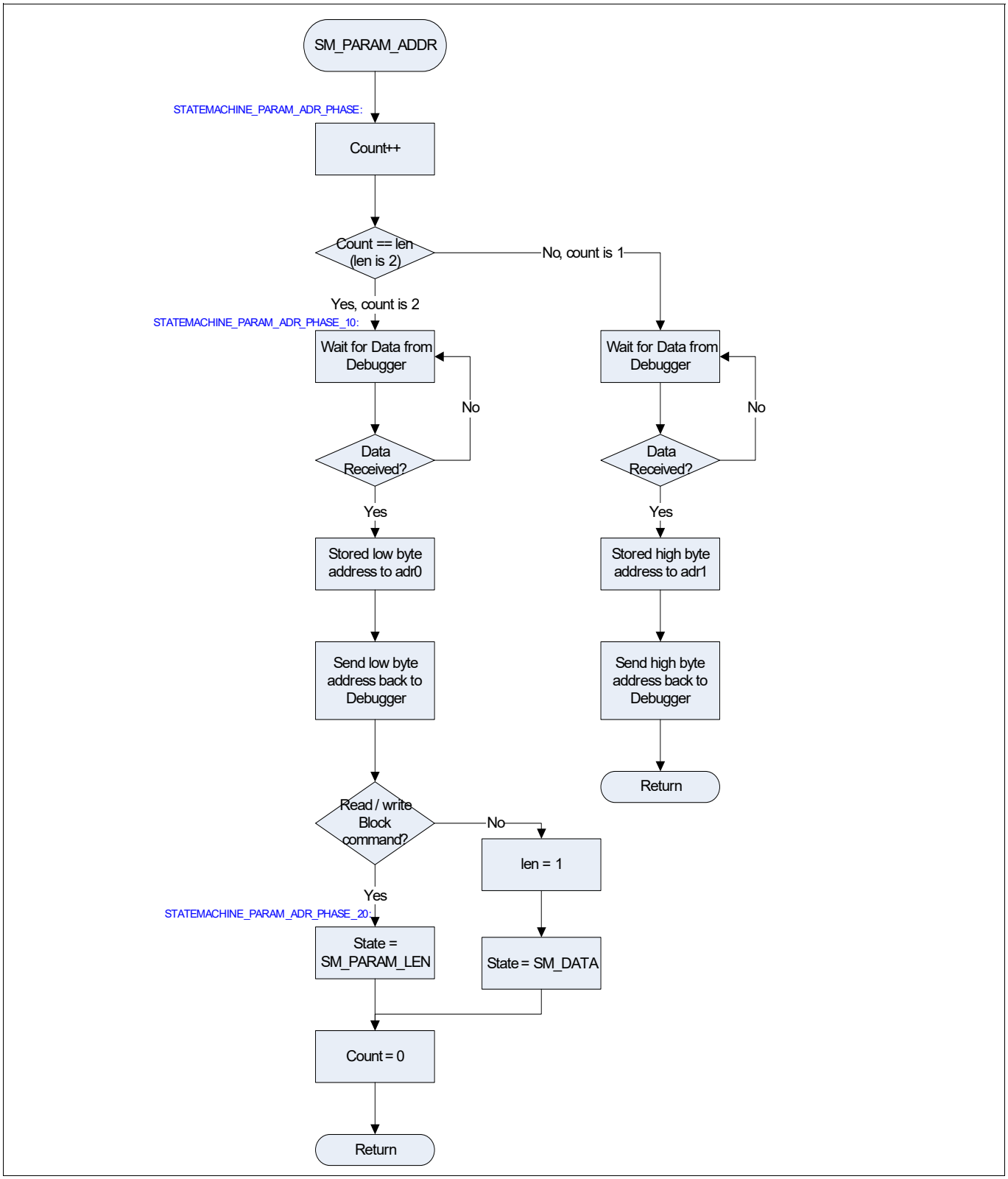


Figure 812 Parameter Address State

Parameter Length State

This state is entered if the command is read block command or write block command. The data received indicates the number of data bytes to be received or sent.

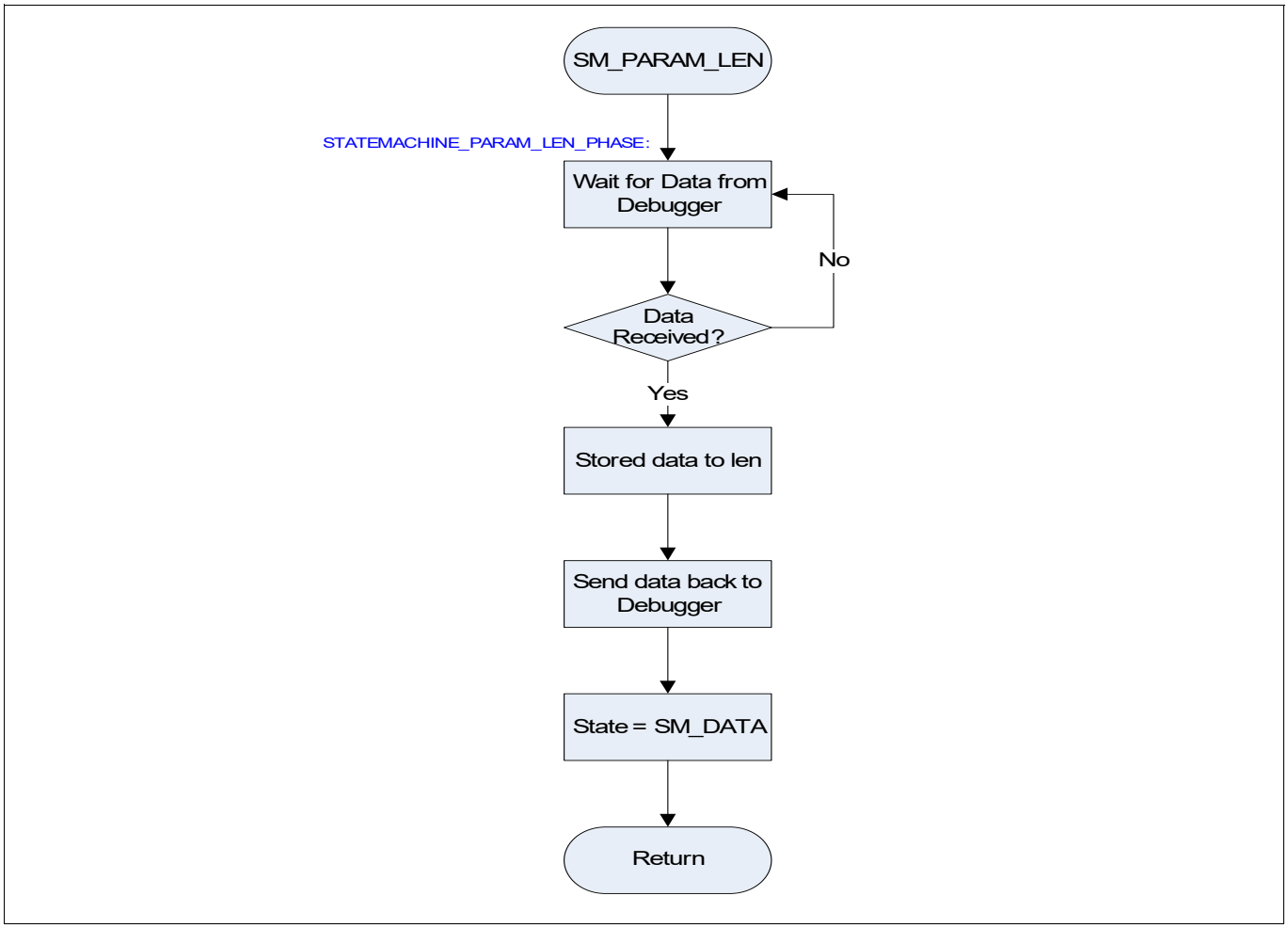


Figure 813 Parameter Length State

Data State

For Read_BYTE and READ_BLOCK commands, data is read from the respective memory or SFR and send to the debugger. For the other 3 commands, data is received from the debugger and written to the memory or SFR.

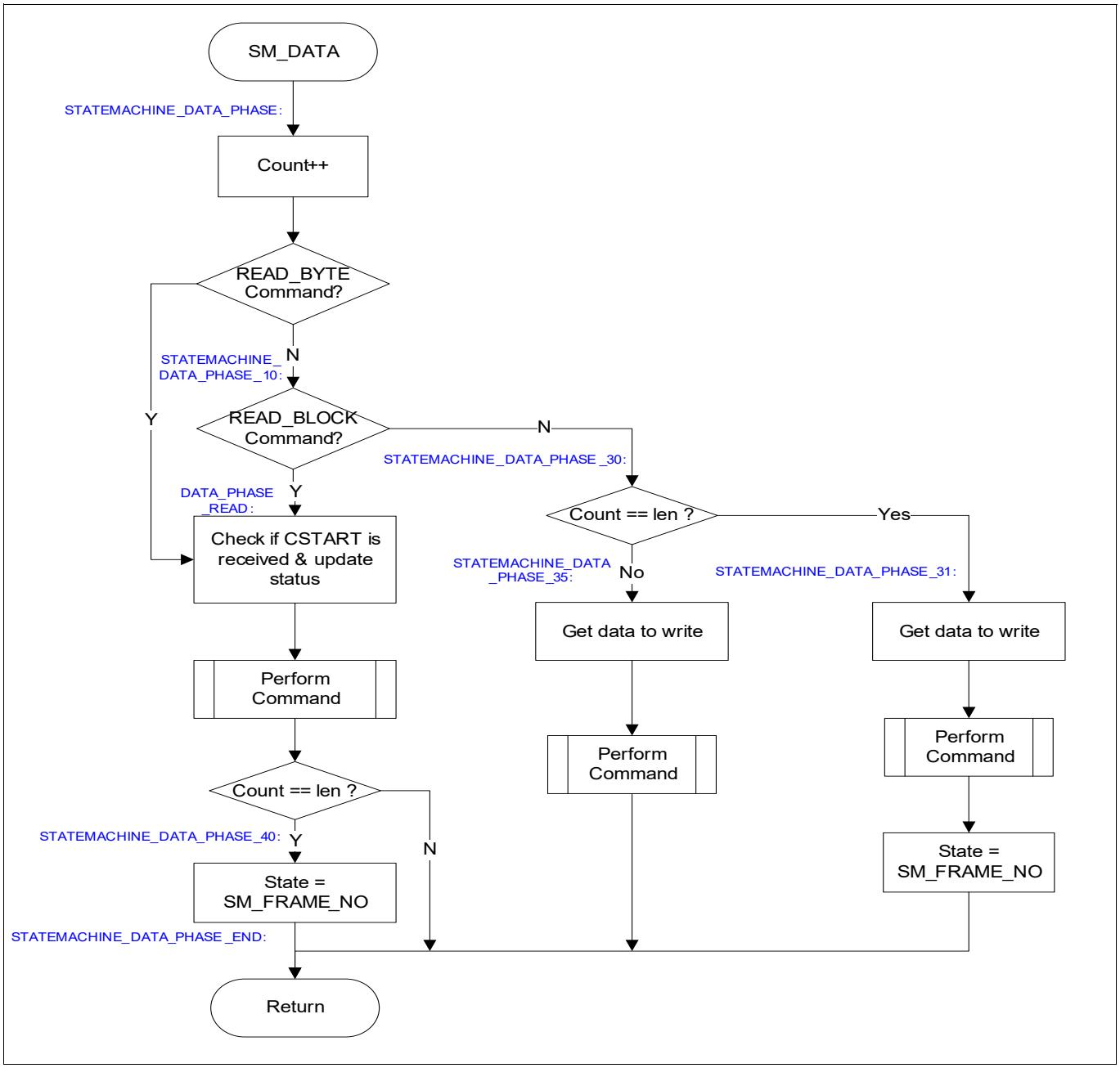


Figure 814 Data State

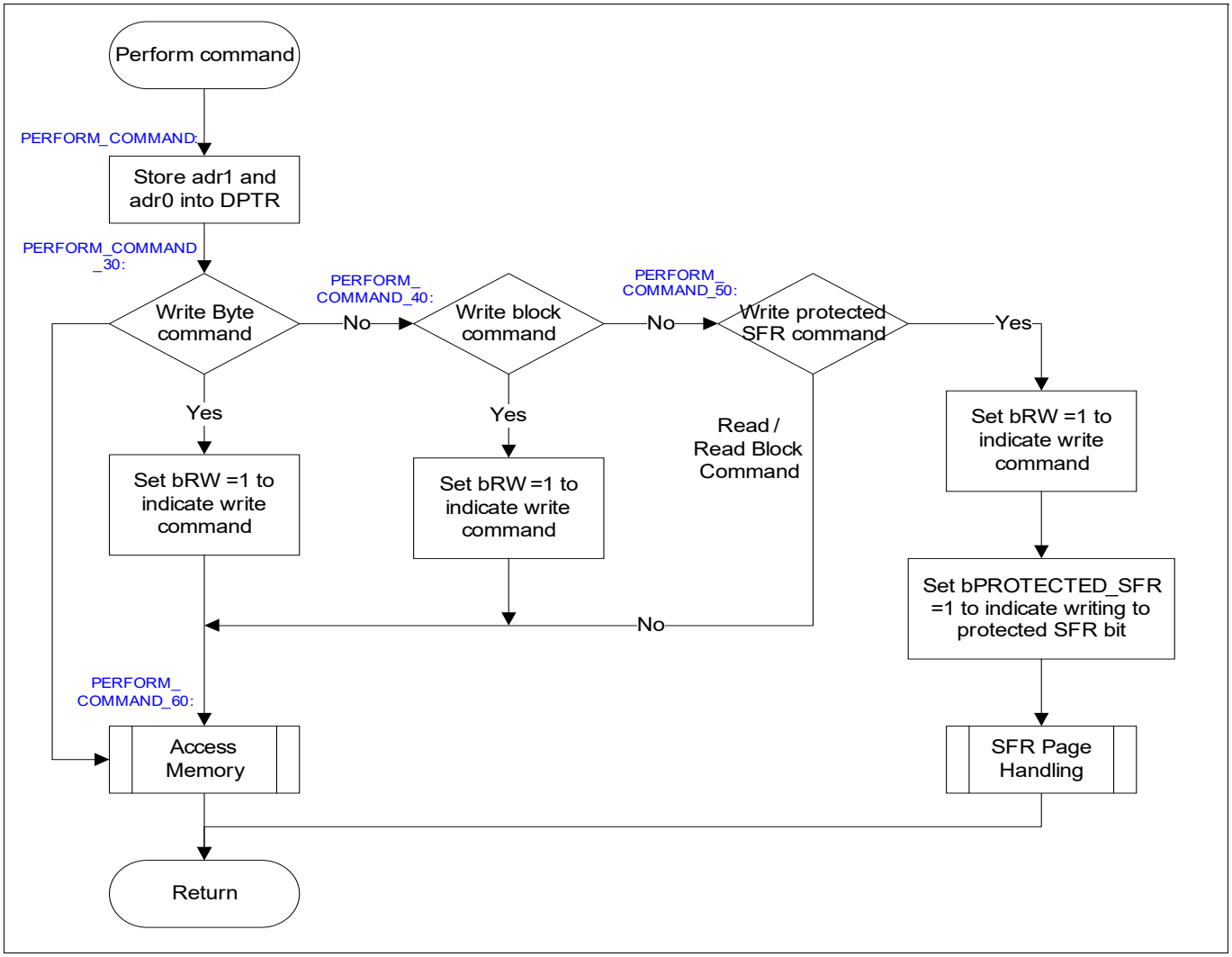


Figure 815 Perform Command

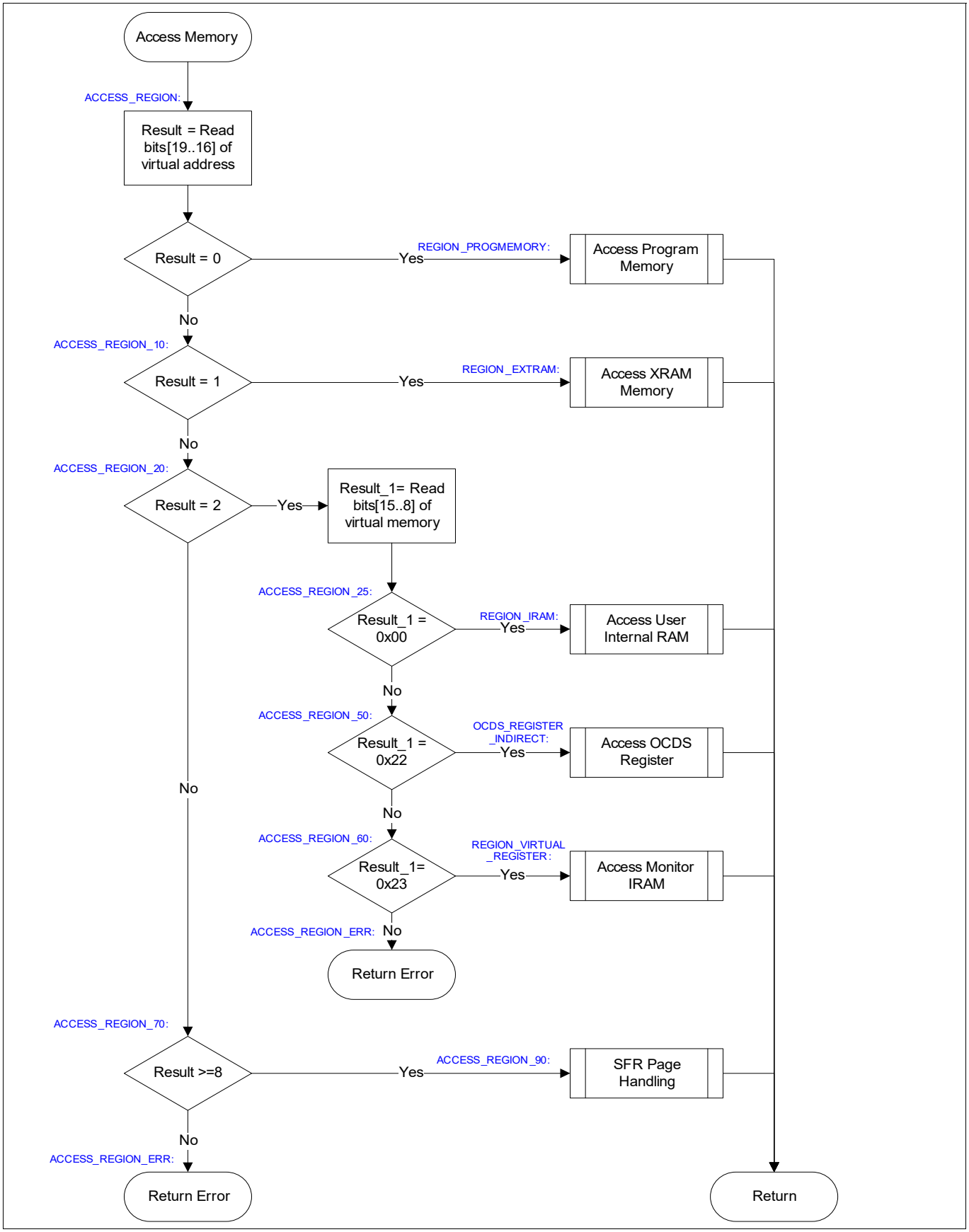


Figure 816 Access Virtual Memory

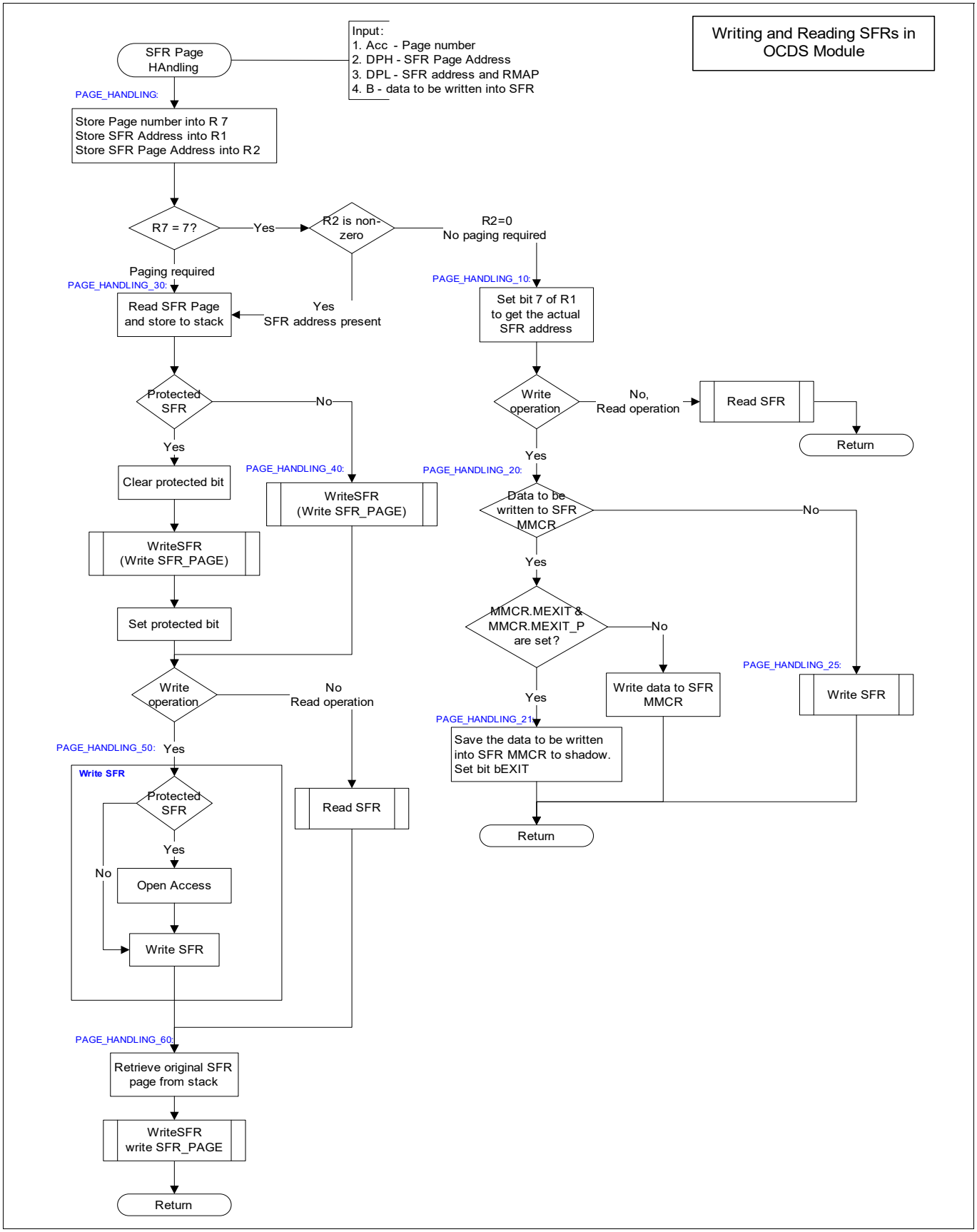


Figure 817 Access SFR

49.5.3 Revision History

Table 565 Revision History

Reference	Change to Previous Version	
V2.4		
	First Official Release of completely reworked SCR chapter	
	No change	

49.6 Special Function Register (SFR) Block

The SFR Block implements all of the special function registers.

The registers in the SFR Block are described and referred to in the relevant chapters (e.g. Interrupt chapter/SCU chapter/Ports chapter etc).

The SFR block supports the local address extension mechanism, and the SFRs in the SFR block are organized into 6 pages.

The page register, SFR SCU_PAGE, at address $F1_H$, contains the page value and the page control information.

49.6.1 Registers in the SFR Block

Page Register for SFRs

SCU_PAGE
Page Register for SFRs **($0F1_H$)** **Reset Value: Table 566**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
OP		**STNR**		**0**	**PAGE**		
w		w		r	rw		

Field	Bits	Type	Description
PAGE	2:0	rw	**Page Bits** When written, the value indicates the new page address. When read, the value indicates the currently active page = addr[y:x+1]. 000_B **PAGE0**, Selection is PAGE0 001_B **PAGE1**, Selection is PAGE1 010_B **PAGE2**, Selection is PAGE2 011_B **PAGE3**, Selection is PAGE3 100_B **PAGE4**, Selection is PAGE4 101_B **PAGE5**, Selection is PAGE5 110_B **PAGE6**, Selection is PAGE6 111_B **PAGE7**, Selection is PAGE7
STNR	5:4	w	**Storage Number** This number indicates which storage bitfield is the target of the operation defined by bit OP. If OP = 10_B, the contents of PAGE are saved in STx before being overwritten with the new value. If OP = 11_B, the contents of PAGE are overwritten by the contents of STx. The value written to bitfield PAGE is ignored. 00_B **MOD_ST0**, ST0 is selected 01_B **MOD_ST1**, ST1 is selected 10_B **MOD_ST2**, ST2 is selected 11_B **MOD_ST3**, ST3 is selected
OP	7:6	w	**Operation** 00_B **PAGE_MANUAL0**, Manual page mode. The value of STNR is ignored and PAGE is directly written. 01_B **PAGE_MANUAL1**, Manual page mode. The value of STNR is ignored and PAGE is directly written 10_B **PAGE_SAVE**, New page programming with automatic page saving. The value written to the bit positions of PAGE is stored. In parallel, the former contents of PAGE are saved in the storage bit field STx indicated by STNR. 11_B **PAGE_RESTORE**, Automatic restore page action. The value written to the bit positions PAGE is ignored and instead, PAGE is overwritten by the contents of the storage bit field STx indicated by STNR.

Field	Bits	Type	Description
0	3	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 566 Reset Values of SCU_PAGE

Reset Type	Reset Value	Note
LVD Reset	0000 X000$_B$	
Generated Reset	0000 –000$_B$	

System Registers

The addresses of the kernel SFRs are listed in the following tables.

Table 567 Register Overview - SYSTEM (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address
PASSWD	Password Register	X	X	086_H
SYSCON0	System Control Register 0	X	X	088_H
XADDRH	On-Chip XRAM Address Higher Order	X	X	087_H

Table 568 Register Overview - SCR (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address
ADCOMP_CON	ADCOMP Control Register	0	SCU_PAGE=0	$0FB_H$
ADCOMP_RES	ADCOMP Result Register	0	SCU_PAGE=0	$0FA_H$
DBG_MODSUSP	Module Suspend Control Register	0	SCU_PAGE=2	$0F9_H$
EXICON0	External Interrupt Control Register 0	0	SCU_PAGE=1	$0F4_H$
EXICON1	External Interrupt Control Register 1	0	SCU_PAGE=1	$0F5_H$
EXICON2	External Interrupt Control Register 2	0	SCU_PAGE=1	$0F6_H$
EXICON3	External Interrupt Control Register 3	0	SCU_PAGE=1	$0F7_H$
IRCON0	Interrupt Request Register 0	0	SCU_PAGE=0	$0F2_H$
IRCON1	Interrupt Request Register 1	0	SCU_PAGE=0	$0F3_H$
IRCON2	Interrupt Request Register 2	0	SCU_PAGE=0	$0F4_H$
MODPISEL0	Peripheral Input Select Register 0	0	SCU_PAGE=2	$0F2_H$
MODPISEL1	Peripheral Input Select Register 1	0	SCU_PAGE=2	$0F3_H$
MODPISEL2	Peripheral Input Select Register 2	0	SCU_PAGE=2	$0F4_H$
MODPISEL3	Peripheral Input Select Register 3	0	SCU_PAGE=2	$0F5_H$
MODPISEL4	Peripheral Input Select Register 4	0	SCU_PAGE=2	$0F6_H$
MODPISEL5	Peripheral Input Select Register 5	0	SCU_PAGE=2	$0F7_H$
NMICON	NMI Control Register	0	SCU_PAGE=1	$0F3_H$
NMISR	NMI Status Register	0	SCU_PAGE=1	$0F2_H$
SCRINTEXCHG	SCR Interrupt Data Exchange Register	0	SCU_PAGE=0	$0F5_H$
SCU_CMCON	Clock Control Register	0	SCU_PAGE=1	$0FA_H$
SCU_MODIEN	Peripheral Interrupt Enable Register	0	SCU_PAGE=1	$0F8_H$
SCU_MRSTST	Main Reset Status Register	0	SCU_PAGE=0	$0F8_H$
SCU_PAGE	Page Register for SFRs	0	X	$0F1_H$
SCU_PMCON1	Peripheral Management Control Register 1	0	SCU_PAGE=1	$0FB_H$
SCU_RSTCON	Reset Control Register	0	SCU_PAGE=1	$0F9_H$
SCU_RSTST	SCR Reset Status Register	0	SCU_PAGE=0	$0F7_H$
SCU_SR	SCU Status Register	0	SCU_PAGE=0	$0F9_H$

Table 568 Register Overview - SCR (sorted by Name) (cont'd)

Short Name	Description	RMAP	PAGE	Offset Address
SCU_STDBYWKP	Standby Mode Wake-Up Register	0	SCU_PAGE=1	$0FC_H$
TCINTEXCHG	TriCore Interrupt Data Exchange Register	0	SCU_PAGE=0	$0F6_H$

49.6.2 Revision History

Table 569 Revision History

Reference	Change to Previous Version	
V2.0		
	First Official Release of completely reworked SCR chapter	
	No change	

49.7 System Control Unit (SCU)

The System Control Unit (SCU) supports all central control tasks in the SCR. The SCU includes the following functional modules:

- Clock System and Control
- Reset Control
- Boot and Startup
- Power Management
- General Port Control
- Kernel SFR

49.7.1 Clock System and Control

Figure 818 shows the block diagram of the clock system in the SCR. It consists of two internal oscillators and a clock control unit (CCU). The two oscillators, a 100 MHz oscillator and a 70 kHz oscillator, are supplied from inside the PMS. These two clock sources to SCR are gated with bit PMSWCR4.SCREN in both normal and standby mode. If this bit is set to “0”, SCR is disabled and the clock inputs would be gated off. The system clock f_{SYS} is generated either by the 100 MHz or 70 kHz internal oscillator.

The selection between 70 kHz and 100 MHz clock is as follows: the 70 kHz clock is selected for SCR operation (f_{sys}) only during standby. During normal run mode, always the 100 MHz is selected.

In standby mode, the 100 MHz oscillator may be powered down as configured in PMSWCR4.SCRCLKSEL. When necessary, the 100 MHz oscillator can be powered up using bit CMCON.OSCPD. Whenever 100 MHz is powered up, the SCR f_{sys} runs on 100 MHz.

The CCU generates all necessary clock signals within the microcontroller from the system clock. It consists of:

- Clock source selection in active mode
- Centralized enable/disable circuit for clock control

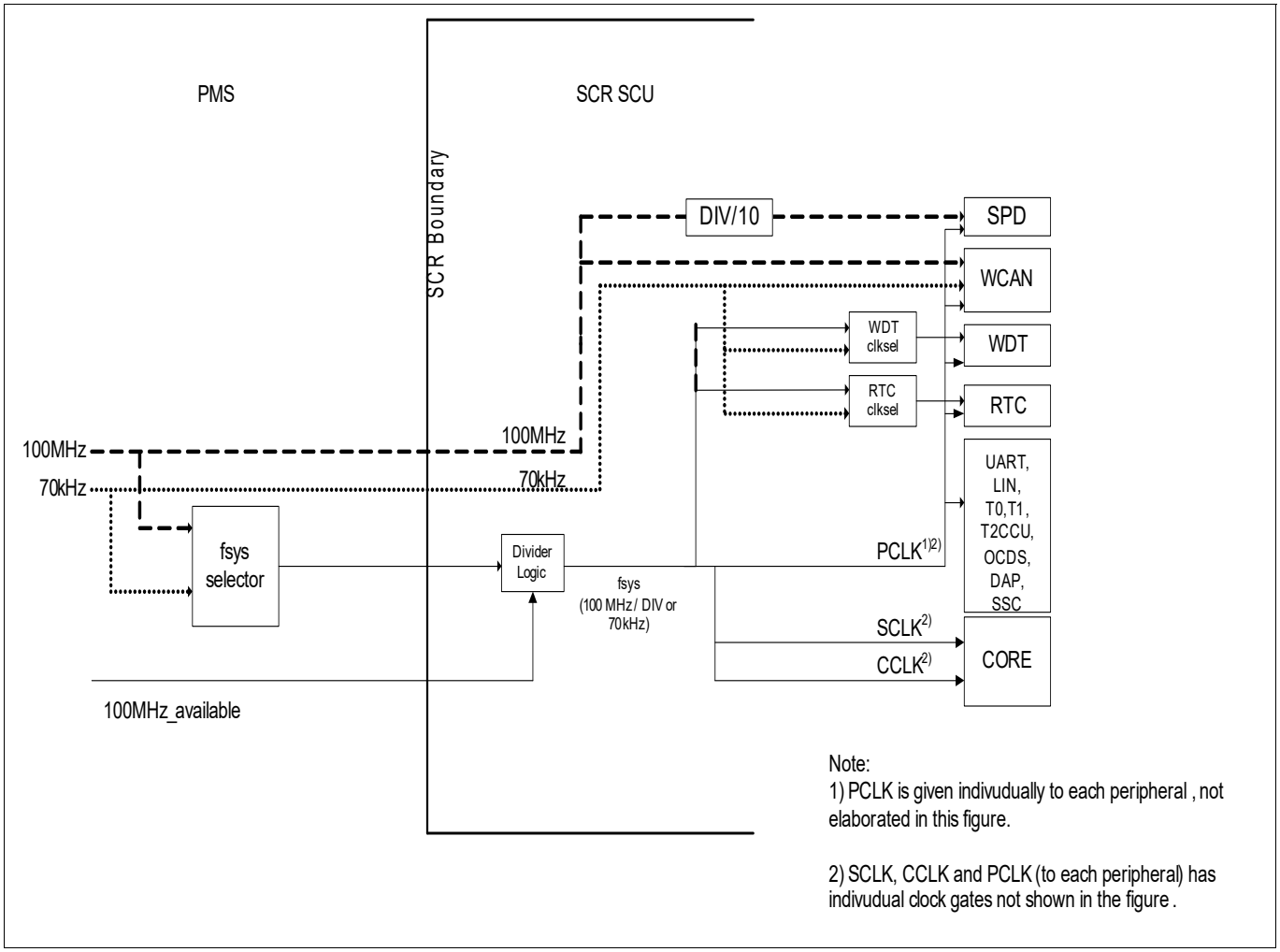


Figure 818 Clock System Block Diagram

In active mode, the main module frequencies (synchronous unless otherwise stated) are as follows:

- CPU clock (CCLK, SCLK) = 70 kHz or 100 MHz / DIV (default)
- Peripheral clock (PCLK) = CPU clock

When 100 MHz clock source is selected, the clock to the CPU and peripherals will be divided by the programmable factor DIV that is selectable by the bit field CMCON.DIV.

Peripherals that are running in PCLK frequency are e.g. UART/LIN, T0, T1, T2CCU, etc.

Note: SPD Debug interface can only be supported when the 100 MHz oscillator is available.

Note: WDT and RTC clock running faster than PCLK frequency is not supported.

49.7.1.1 CCU Register

Clock Control Register

SCU_CMCON
Clock Control Register **($0FA_H$)** **Reset Value: Table 571**
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
0		**OSCPD**	**OSCWAKE**	**DIV**			
r		rwh	rw	rw			

Field	Bits	Type	Description
DIV	3:0	rw	**Clock Divider** 0_H **DIV5**, DIV = 5; fSYS = 20 MHz 1_H **DIV8**, DIV = 8; fSYS = 12.5 MHz 2_H **DIV10**, DIV = 10; fSYS = 10 MHz 3_H **DIV16**, DIV = 16; fSYS = 6.25 MHz 4_H **DIV20**, DIV = 20; fSYS = 5 MHz 5_H **DIV32**, DIV = 32; fSYS = 3.125 MHz 6_H **DIV40**, DIV = 40; fSYS = 2.5 MHz 7_H **DIV50**, DIV = 50; fSYS = 2 MHz 8_H **DIV80**, DIV = 80; fSYS = 1.25 MHz 9_H **DIV100**, DIV = 100; fSYS = 1 MHz A_H **DIV125**, DIV = 125; fSYS = 0.8 MHz B_H **DIV160**, DIV = 160; fSYS = 0.625 MHz C_H **DIV200**, DIV = 200; fSYS = 0.5 MHz D_H **DIV250**, DIV = 250; fSYS = 0.4 MHz
OSCWAKE	4	rw	**Oscillator Wake-Up** Automatic oscillator wake-up. 0_B Modules cannot clear OSCPD 1_B Modules can clear OSCPD and enable 100 MHz oscillator
OSCPD	5	rwh	**100 MHz Oscillator Power Down Control in Standby Mode** This bit is used to power down the 100 MHz oscillator in standby mode. The OSCPD bit is a protected bit. When the Protection Scheme is activated, this bit cannot be written directly. For more information on Protection Scheme, see Section "Miscellaneous Control". 0_B 100 MHz oscillator is not powered down in standby mode 1_B 100 MHz oscillator is powered down in standby mode
0	7:6	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 570 Access Mode Restrictions of SCU_CMCON sorted by descending priority

Mode Name	Access Mode		Description
otherwise	rh	OSCPD	
PASSWD.PROTECT_S = 0 (default)	rwh	OSCPD	

Table 571 Reset Values of SCU_CMCON

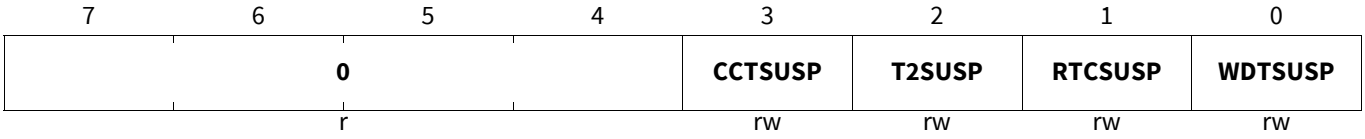
Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.7.1.2 Module Suspend Control

When the On-Chip Debug Support (OCDS) is in Monitor Mode (MMCR2.MMODE = 1) and the Debug-Suspend signal is active (MMCR2.DSUSP = 1), timers in certain modules in the SCR can be suspended based on the settings of their corresponding module suspend bits in register MODSUSP. When suspended, only the timer stops counting as the counter input clock is gated off. The module is still clocked so that module registers are accessible.

Module Suspend Control Register

DBG_MODSUSP
Module Suspend Control Register **($0F9_H$)** **Reset Value: Table 572**
RMAP: 0, PAGE: SCU_PAGE=2



Field	Bits	Type	Description
WDTSUSP	0	rw	**SCU Watchdog Timer Debug Suspend Bit** 0_B WDT will not be suspended 1_B WDT will be suspended
RTCSUSP	1	rw	**Real-Time Clock Debug Suspend Bit** 0_B Real-Time clock will not be suspended 1_B Real-Time clock will be suspended
T2SUSP	2	rw	**Timer 2 Debug Suspend Bit** 0_B Timer 2 will not be suspended 1_B Timer 2 will be suspended
CCTSUSP	3	rw	**Compare/Capture Timer (CCT) Debug Suspend Bit** 0_B CCT will not be suspended 1_B CCT will be suspended
0	7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 572 Reset Values of DBG_MODSUSP

Reset Type	Reset Value	Note
LVD Reset	$X1_H$	
Generated Reset	-1_H	

49.7.2 Reset Control

The SCR has three types of resets:

- Power reset caused by an LVD reset,
- Reset from Main Controller via PMSWCR4.SCRSTREQ and PORST,
- and generated resets.

The SCR Watchdog Timer Reset, the RAM double bit ECC Reset and the Soft Reset which have the source within the SCR are called Generated Resets.

Combination of reset via PMSWCR4.SCRSTREQ and an SCRPORST (trigger on SCR PORST pin) is known as Main Reset.

When the SCR is first powered up, the SCR enters User Mode 0 and start operation with the release of the system reset via the LVD reset. The CPU then starts to execute from the Boot ROM firmware. User Mode 0 is used by CPUx to program the user code to the SCR internal XRAM. Once the XRAM is programmed, a new boot mode must be selected via PMSWCR4.SCRCFG. A reset must be issued by CPUx via bit PMSWCR4.SCRSTREQ to start running SCR in the new boot mode.

In User Mode 0, the 8-bit CPU starts to execute an infinite loop from the Boot ROM firmware and user code will not be executed in User Mode 0 once the system reset is released. Please refer to “Boot mode” section for detailed information about the other boot modes.

For the generated resets the following applies:

- The Watchdog Timer (WDT) module is capable of resetting the device if it detects a malfunction in the system.
- A double bit ECC event in the SRAM can also trigger a reset.
- As for Soft reset, it can be triggered by application software where applicable.

49.7.2.1 Types of Reset

The following reset types affect the SCR.

External resets (reset sources not within the SCR):

- LVD reset, caused by an undervoltage detection
  - Triggered asynchronously during power failure (undervoltage),
  - OCDS/Debug-system is reset only when the LVD reset occurs.
- Main reset
  - SCRPORST triggered by the PORST pin , if enabled using PMSWCR4.PORSTREQ triggering a main reset.
  - This is a combined reset, consisting of all the Power reset sources and synchronous reset by the main controller CPUx via bit PMSWCR4.SCRSTREQ.
  - Main reset also activates the Generated reset
  - Resets all registers including the reset indication bits.

Generated Resets (i.e., reset with sources within the SCR):

- SCU Watchdog Timer (WDT) reset
  - Requested by WDT overflow or out of boundary refresh event if it is enabled via RSTCON.WDTRSTEN.
  - Resets all registers except the reset indication bits.
  - If this reset is enabled, it can also trigger an interrupt to the Tricore.
- RAM Double Bit ECC Reset
  - Requested by ECC event if it is enabled via RSTCON.ECCRSTEN.
  - Resets all registers except the reset indication bits.
  - If this reset is enabled, it can also trigger an interrupt to the Tricore.
- Soft reset (generated reset)
  - Requested by soft reset event
  - Resets all registers except the reset indication bits.

This section describes the definition and controls depending on the reset source. **Figure 819** shows a simplified overview of the reset signals.

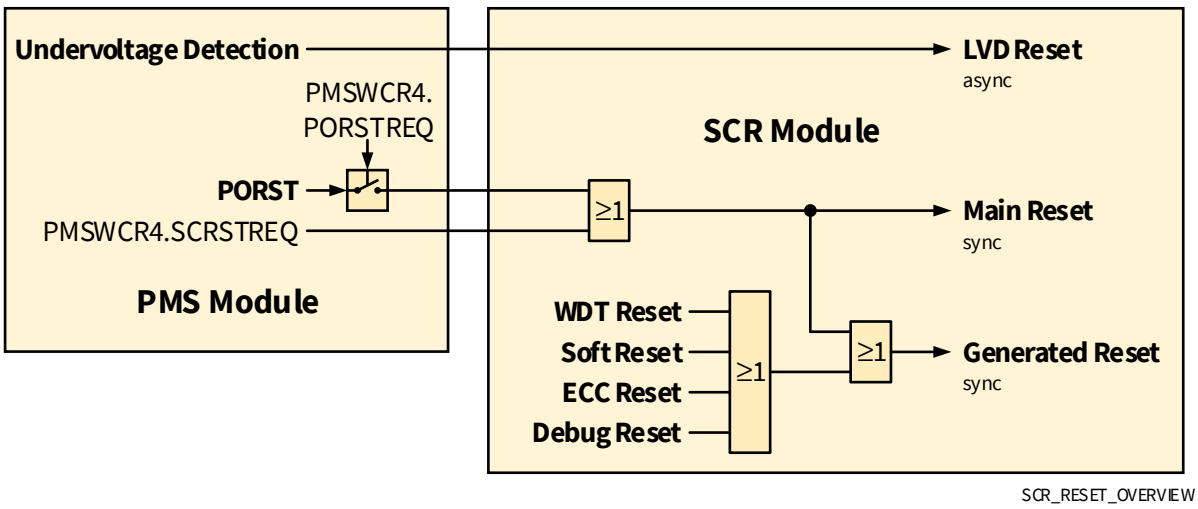


Figure 819 Simplified Reset Overview

Note: In the TC38x and TC39x, triggering of the PORST would reset the ports of the PMS. Some of them are shared with the SCR, meaning the communication of the SCR with the external world would be broken. In the TC35x and TC37x, the PORST does not reset any of the PMS pads, meaning the communication will be maintained.

49.7.2.1.1 LVD Reset (Undervoltage Reset)

The LVD reset, resets the SCR including the OCDS, and starts the correction in case of an ECC error.

An LVD reset will also trigger the Main and Generated Reset.

49.7.2.1.2 Main Reset (External Reset)

Main reset includes PORST and reset from Tricore via the PMSWCR4.SCRSTREQ bit.

A Main reset also triggers the Generated reset.

When the SCR is first powered up, the SCR enters User Mode 0 and start operation with the release of the system reset via the LVD Reset. User Mode 0 is used by CPUx to program the user code to the SCR internal XRAM. Once the XRAM is programmed, a new boot mode must be selected via PMSWCR4.SCRCFG. The Main Reset must then be issued as triggered by CPUx via bit PMSWCR4.SCRSTREQ to start running SCR in the new boot mode.

In User Mode 0, the 8-bit CPU starts to execute an infinite loop from the Boot ROM firmware and user code will not be executed. For more information on the other boot modes (User Mode 1 and Debug mode), refer to “Boot mode” section.

Note: Resetting SCR to enter a new boot mode is only possible via the reset that is triggered by bit PMSWCR4.SCRSTREQ from the main controller.

49.7.2.1.3 Generated Resets

Generated Reset is activated together with Main reset, and on a Watchdog Timer Reset, an ECC Reset, a Soft Reset or a Debug Reset.

Watchdog Timer Reset

The watchdog timer reset is an internal reset and can be disabled by bit RSTCON.WDTRSTEN. The Watchdog Timer (WDT) maintains a counter that must be refreshed or cleared periodically. If the WDT is not serviced correctly and in time, it will generate an NMI request to the CPU and then reset the device (WDT_RST) after a predefined time-out period if this reset type is enabled. When the reset is disabled, no reset will be triggered when the overflow event or out of boundary refresh event happens.

Bit RSTST.WDTRST is used to indicate the watchdog timer reset status.

Note: The event signal can be cleared by main reset.

Double Bit ECC Reset

The double bit ECC reset is an internal reset and can be disabled by bit RSTCON.ECCRSTEN (not adviced). The 8-bit data from the CPU is encoded with an Error Correction Code (ECC) before being stored in the XRAM memory. During a read access, data is retrieved from the memory and decoded for dynamic error detection and correction. If a double bit error is detected, it will generate a reset if this reset type is enabled. When the reset is disabled, no reset will be triggered when the ECC event happens.

Bit RSTST.ECCRST is used to indicate the ECC reset status.

Note: The event signal can be cleared by main reset.

Soft Reset

Soft reset is an internal reset that is caused by a software set of the soft reset request bit, SWRQ, in RSTCON register. It can be triggered by application software where necessary.

Bit RSTST.SOFTRST is used to indicate the soft reset status.

Note: The event signal can be cleared by main reset.

Debug Reset

A debug reset is a generated reset that is caused by debugger reset commands.

49.7.2.2 Module Reset Behavior

Table 573 gives an overview on how the various modules or functions of the SCR are affected with respect to the reset type. An “X” means that the module/function is reset to its default state.

Table 573 Effect of Reset on Modules/Functions

Module/ Function	LVD Reset	Main Reset	Generated Reset
CPU Core	X	X	X
SCU	X	X	X except reset indication bits
Peripherals	X	X	X
Debug System	X	Not affected	Not affected
Port Control	X	X	X
On-Chip Static RAM	Affected, unreliable	Not affected, reliable	Not affected, reliable
Clock System	X	X	X

49.7.2.3 Reset Register Description

RSTST register consist of the reset indication bits of a ECC reset, WDT reset and soft reset. **Table 575** shows the reset value of RSTST register after these events.

SCR Reset Status Register

SCU_RSTST
SCR Reset Status Register **($0F7_H$)** **Reset Value: Table 574**
RMAP: 0, PAGE: SCU_PAGE=0

7	6	5	4	3	2	1	0
0					**SOFTRST**	**WDTRST**	**ECCRST**
r					rwh	rwh	rwh

Field	Bits	Type	Description
ECCRST	0	rwh	**Double Bit ECC Reset Indication Bit** This bit can only be set by hardware and cleared by software. 0_B No ECC reset has occurred 1_B ECC reset has occurred
WDTRST	1	rwh	**Watchdog Timer Reset Indication Bit** This bit can only be set by hardware and cleared by software. 0_B No watchdog reset has occurred 1_B Watchdog reset has occurred
SOFTRST	2	rwh	**Soft Reset Indication Bit** This bit can only be set by hardware and cleared by software. 0_B No soft reset has occurred 1_B Soft reset has occurred
0	7:3	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 574 Reset Values of SCU_RSTST

Reset Type	Reset Value	Note
LVD Reset	XXXX $X000_B$	
Main Reset	---- -000_B	

Table 575 Reset Value of Register RSTST

Reset Source	Reset Value
Double bit ECC Reset	0000 $0XX1_B$
WDT Reset	0000 $0X1X_B$
Soft Reset	0000 $01XX_B$
Main Reset	0000 0000_B

Main Reset Status Register

SCU_MRSTST
Main Reset Status Register **(0F8$_H$)** **Reset Value: Table 576**
RMAP: 0, PAGE: SCU_PAGE=0

7	6	5	4	3	2	1	0
0						RST	SMURST
r						rwh	rwh

Field	Bits	Type	Description
SMURST	0	rwh	**SMU Reset Status Bit** This bit can only be set by hardware, and cleared by software. 0_B No reset issued by SMU since the last clearing of the bit 1_B Reset issued by SMU
RST	1	rwh	**Application Reset Status Bit** This bit can only be set by hardware, and cleared by software. 0_B Application reset is not triggered since the last clearing of the bit 1_B Application reset triggered
0	7:2	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 576 Reset Values of SCU_MRSTST

Reset Type	Reset Value	Note
LVD Reset	XXXX XX00$_B$	
Generated Reset	---- --00$_B$	

Reset Control Register

SCU_RSTCON
Reset Control Register **(0F9$_H$)** **Reset Value: Table 578**
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
SWRQ			0			WDTRSTEN	ECCRSTEN
rwh			r			rw	rw

Field	Bits	Type	Description
ECCRSTEN	0	rw	**Double Bit ECC Reset Enable Bit** 0_B ECC reset is disabled 1_B ECC reset is enabled
WDTRSTEN	1	rw	**Watchdog Reset Enable Bit** 0_B Watchdog reset is disabled 1_B Watchdog reset is enabled

Field	Bits	Type	Description
SWRQ	7	rwh	**Soft Reset Request** This bit is automatically cleared by hardware. The SWRQ bit is a protected bit. When the Protection Scheme is activated, this bit cannot be written directly. For more information on Protection Scheme, see Section "Miscellaneous Control". 0_B No action 1_B On set, soft reset is requested
0	6:2	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 577 Access Mode Restrictions of SCU_RSTCON sorted by descending priority

Mode Name	Access Mode		Description
otherwise	rh	SWRQ	
PASSWD.PROTECT_S = 0 (default)	rwh	SWRQ	

Table 578 Reset Values of SCU_RSTCON

Reset Type	Reset Value	Note
LVD Reset	$0XXX\ XX11_B$	
Generated Reset	$0\text{---}\ \text{--}10_B$	

49.7.3 Boot and Startup

This section describes the boot mode for SCR. After the first power-up, SCR will enter User Mode 0. Subsequently, bitfield PMSWCR4.SCRCFG is used to change the boot mode that SCR will enter after a reset is deasserted. A reset to the SCR can be triggered by a PMSWCR4.SCRSTREQ reset request to start it off in the chosen mode. If an internal reset happens after a change of PMSWCR4.SCRCFG, the SCR will also start in the mode specified by the SCRCFG value. **Table 579** shows the various boot modes available in SCR.

Table 579 Boot Modes

SCRCFG Value	Boot Mode	Descriptions
00_H	User Mode 0	Default mode after LVD reset with no user code in XRAM and CPUx can download code using this mode. User code in XRAM will not be executed
01_H	User Mode 1	User Code in XRAM at address 0000_H will be executed
02_H	OCDS Mode	Debug mode using SCR DAP0_0 pin and DAP1_0 pin
03_H		Debug mode using SCR DAP0_1 pin and DAP1_1 pin
04_H - 05_H		Debug mode using SCR SPD_0 pin
06_H - 07_H		Debug mode using SCR SPD_1 pin
$0A_H$ - $0B_H$		Debug mode using SOC DAP interface
$0C_H$ - $0F_H$		Debug mode using SOC SPD interface

At end of the startup routine, an interrupt is triggered to the main controller by setting bit NMICON.SCRINTTC to 1 with a value of 80_H in SCRINTEXCHG register depending on the boot mode. This is used to indicate that SCR has finished executing the startup code and is ready for CPUx to download code to XRAM in User Mode 0 or jump to user code in User Mode 1 or jump to debug monitor code in OCDS mode.

Note: To change the boot mode, SCR has to be enabled first. Changing of boot mode when SCR is disabled will have no effect.

49.7.3.1 Boot ROM Operating Mode

After a reset, the CPU will always start by executing the Boot ROM code which occupies the program memory address space 0000_H – $07FF_H$. The Boot ROM start-up procedure will first switch the address space for the Boot ROM to $D000_H$ – $D7FF_H$. The remaining Boot ROM start-up procedure will be executed from $D60X_H$. The memory organization of the SCR shown in this document is after the Boot ROM address switch where the different operating modes are executed.

49.7.3.1.1 User Mode 0

User Mode 0 is the default boot mode after an LVD reset where XRAM memory is not programmed. This mode allows the downloading of code to SC internal XRAM memory. In this mode, the CPU will only executed start-up code in Boot ROM. At the end of start-up code, an interrupt would be triggered to the main controller by setting bit NMICON.SCRINTTC to 1 with a value of 80_H in SCRINTEXCHG register. This is used to indicate that SCR is ready for CPUx to download code to XRAM.

No application software can be exectuted in User Mode 0

49.7.3.1.2 User Mode 1

If the User Mode 1 is selected, the Boot ROM will jump to program memory address 0000_H to execute the user code in the XRAM memory. To use this mode, the XRAM must be pre-loaded with user code. This is the normal operating mode of the SCR.

For the last 8 bytes of XRAM starting at address $1FF8_H$, user need to program 4 sets of pre-fixed bytes with each set containing 55_H followed by AA_H. User code will not be executed and SCR will enter an endless loop if the memory content does not match these data sequence. It is used to avoid an unintentional entry to User Mode 1. Before entering the endless loop, the start-up code will trigger an interrupt to the main controller by setting bit NMICON.SCRINTTC to 1 with a value of 81_H in SCRINTEXCHG register. If there is a match, the same interrupt is triggered to the main controller with a value of 80_H in SCRINTEXCHG register to indicate the execution of user code.

49.7.3.1.3 OCDS Mode with SCR DAP/SPD pin

If the OCDS mode with SCR DAP/SPD pin is selected, the debug mode will be entered for debugging program code. The OCDS hardware is initialized and a jump to program memory address 0000_H is performed next. The user code in the XRAM memory is executed and the debugging process may be started.

Note: The DAP/SPD pin must be directly accessible by SCR. The control of these pins must be given to SCR using bit Pn_PCSR.SELx enabling this boot mode.

49.7.3.1.4 OCDS Mode with SOC DAP/SPD pin

If the OCDS mode with SOC DAP/SPD pin is selected, the debug mode will be entered for debugging program code using the SOC DAP/SPD pins that are also used by the main debug system. The OCDS hardware is initialized and a jump to program memory address 0000_H is performed next. The user code in the XRAM memory is executed and the debugging process may be started.

49.7.4 Power Management

This section describes the features and functionality provided for power management of the SCR. In addition, the exiting of standby mode via SCR will be described in this section. Standby mode is one of the power saving mode in the SCR.

49.7.4.1 Overview

The SCR power-management system allows software to configure the various processing units so that they automatically adjust to draw the minimum necessary power for the application.

There are generally two power modes: Active Mode and Idle Mode. The operation of the system components in each of these states can be configured by software. The power modes provide flexible reduction of power consumption through a combination of techniques, including:

- Stopping the CPU clock
- Stopping the clocks of other system components individually
- Clock-speed reduction of some peripheral components

In Active mode, the CPU and peripherals is running in system frequency.

In Idle mode, the Core is stopped with its clock disabled. Peripherals, with respective input clocks not disabled, are still functional. The watchdog timer, however, must be disabled by user before the system enters into an idle

mode; otherwise, it may generate an internal reset when an overflow occurs and this will disrupt the idle mode. The idle mode is terminated automatically when any enabled interrupt signal is detected.

49.7.4.2 Functional Description

This section describes the power-save modes, their operations, and entry and exit. It also describes the respective behavior of the SCR system components.

49.7.4.2.1 Idle Mode

Software requests Idle Mode by setting the bit PCON.IDLE to 1.

The power management state machine (PMSM) posts an idle request signal to the CPU. The CPU finishes its current operation, sends an acknowledge signal back to the PMSM, and then enters into an inactive state in which the CPU clock (CCLK) is disabled.

The system returns to either active mode when there is an incoming interrupt request to an enabled interrupt node.

Interrupt to the CPU causes the CCLK to recover. Upon RETI instruction, the CPU will return to execute the next instruction following the instruction which sets the IDLE bit to 1.

Note: The user is advised to disable watchdog timer prior to entering the Idle Mode.

49.7.4.2.2 Peripheral Management

It is possible to disable the clock for some peripherals separately in order to save power. Refer to **“Peripheral Management Register” on Page 74**. Each bit in the register can be set to disable the clock to that module.

49.7.4.3 Exit Standby Mode via SCR

In standby mode of the system, SCR could be operating in either active mode or idle mode. Hence, SCR could be one of the wake-up source to exit standby mode. Before entering standby mode, the followings steps related to SCR need to be setup before issuing the standby request:

- Configure the wake-up signals related to SCR that would trigger the exit from standby mode via PMSWCR4 register
- Configure the clock selection for SCR in standby mode via SCRCLKSEL bit.

To exit from standby mode, one of the following wake-up sources from SCR could be used:

- WDT event
- SRAM ECC double bit error
- Software request
- WCAN WUF event
- RTC compare match event

Besides the specified events such as WDT overflow event, SRAM ECC error, WCAN event or RTC compare match event, other types of events from SSC, T0, T1, T2CCU and UART could issue a wake-up request via the bit SCRWKP in STDBYWKP register.

49.7.4.4 Register Description

Power Control Register

PCON
Power Control Register **($0D9_H$)** **Reset Value: Table 580**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
SMOD		**0**		**GF1**	**GF0**	**0**	**IDLE**
rw		r		rw	rw	r	rw

Field	Bits	Type	Description
IDLE	0	rw	**Idle Mode Enable** 0_B Do not enter Idle Mode 1_B Enter Idle Mode
GF0	2	rw	**General Purpose Flag Bit 0**
GF1	3	rw	**General Purpose Flag Bit 1**
SMOD	7	rw	**Double Baud Rate Enable** 0_B Do not double the baud rate of serial interface in mode 2 1_B Double the baud rate of serial interface in mode 2
0	1, 6:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 580 Reset Values of PCON

Reset Type	Reset Value	Note
Generated Reset	$0XXX\ 00X0_B$	
LVD Reset	$0\text{---}\ 00\text{–}0_B$	

Standby Mode Wake-Up Register

SCU_STDBYWKP
Standby Mode Wake-Up Register **($0FC_H$)** **Reset Value: Table 581**
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
	0		**ECCWKSEL**	**WDTWKSEL**	**WCANWKSEL**	**RTCWKSEL**	**SCRWKP**
	r		rw	rw	rw	rw	rwh

Field	Bits	Type	Description
SCRWKP	0	rwh	**Software Wake-Up from Standby Mode** This bit is automatically cleared by hardware. 0_B No request to trigger a wake-up from standby mode 1_B Trigger a wake-up from standby mode

Field	Bits	Type	Description
RTCWKSEL	1	rw	**RTC Wake-Up Select from Standby Mode** 0_B Wake-up from standby mode via RTC is not selected 1_B Wake-up from standby mode via RTC is selected
WCANWKSEL	2	rw	**WCAN Interrupt Wake-Up Select from Standby Mode** 0_B Wake-up from standby mode via WCAN Interrupt is not selected 1_B Wake-up from standby mode via WCAN Interrupt is selected
WDTWKSEL	3	rw	**WDT Wake-Up Select from Standby Mode** 0_B Wake-up from standby mode is not selected when WDT event occurred 1_B Wake-up from standby mode is selected when WDT event occurred
ECCWKSEL	4	rw	**RAM ECC Wake-Up Select from Standby Mode** 0_B Wake-up from standby mode is not selected when RAM ECC event occurred 1_B Wake-up from standby mode is selected when RAM ECC event occurred
0	7:5	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 581 Reset Values of SCU_STDBYWKP

Reset Type	Reset Value	Note
LVD Reset	XXX0 0000_B	
Generated Reset	---0 0000_B	

SCU Status Register

The status register indicates whether the device is currently in standby mode or not. Further information on standby has to be saved on TriCore side to the XRAM.

SCU Status Register

SCU_SR
SCU Status Register **($0F9_H$)** **Reset Value: Table 582**
RMAP: 0, PAGE: SCU_PAGE=0

7	6	5	4	3	2	1	0
0				**STBY**		**0**	
r				rh		r	

Field	Bits	Type	Description
STBY	3	rh	**Standby Mode Status** 0_B Standby Mode not entered 1_B Standby Mode entered
0	2:0, 7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 582 Reset Values of SCU_SR

Reset Type	Reset Value	Note
LVD Reset	XXXX 0XXX$_B$	
Generated Reset	---- 0---$_B$	

49.7.4.4.1 Peripheral Management Register

Each register bit controls one peripheral. When this bit is set, the request signal to gate off the clock to the respective module is asserted. The SCU will then synchronize the gating off of the clock to the peripheral. In SCR, the clock inputs of all peripherals as specified in the PMCON1 register are gated off upon power on reset (LVD reset).

Peripheral Management Control Register 1

SCU_PMCON1
Peripheral Management Control Register 1 ($0FB_H$) Reset Value: Table 583
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
0	**OCDS_DIS**	**LIN_DIS**	**WDT_DIS**	**WCAN_DIS**	**RTC_DIS**	**T2CCU_DIS**	**SSC_DIS**
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SSC_DIS	0	rw	**SSC Disable Request (active high)** 0_B SSC is in normal operation 1_B Request to disable the SSC (default)
T2CCU_DIS	1	rw	**T2CCU Disable Request (active high)** 0_B T2CCU is in normal operation 1_B Request to disable the T2CCU (default)
RTC_DIS	2	rw	**RTC Disable Request (active high)** 0_B RTC is in normal operation 1_B Request to disable the RTC (default)
WCAN_DIS	3	rw	**WCAN Disable Request (active high)** 0_B WCAN is in normal operation 1_B Request to disable the WCAN (default)
WDT_DIS	4	rw	**Watchdog Disable Request (active high)** 0_B Watchdog is in normal operation 1_B Request to disable the Watchdog (default)
LIN_DIS	5	rw	**LIN Disable Request (active high)** 0_B LIN baudrate generator and detector are in normal operation 1_B Request to disable the LIN baudrate generator and detector (default)
OCDS_DIS	6	rw	**OCDS Disable Request (active high)** 0_B OCDS and Debug System is in normal operation (this is the default value, and the clock to OCDS and Debug System is enabled by default) 1_B Request to disable the OCDS and Debug System (during normal operation, application may choose to gate the OCDS and Debug System clock to save power)
0	7	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 583 Reset Values of SCU_PMCON1

Reset Type	Reset Value	Note
LVD Reset	X011 1111_B	
Generated Reset	–011 1111_B	

49.7.5 Miscellaneous Control

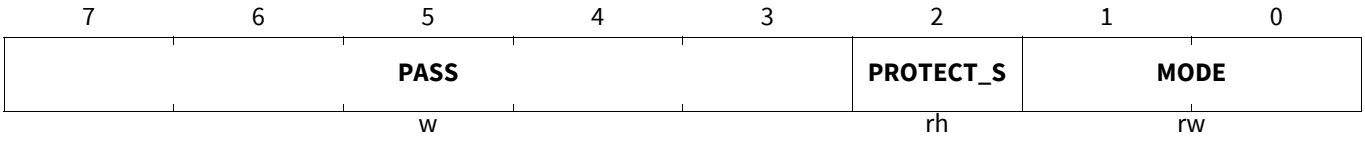
There are certain miscellaneous functions supported such as the bit protection scheme.

49.7.5.1 Bit Protection Register

The Bit-Protection Scheme disallows direct software writing of selected bits (i.e. Protected bits) by the SFR PASSWD. When the bit field MODE is 11_B, writing 10011_B to the bit field PASS opens access to writing of all protected bits and writing 10101_B to the bit field PASS closes access to writing of all protected bits. Note that access is opened for maximum 34 CCLKs if the “close access” password is not written. If “open access” password is written again before the end of 34 CCLK cycles, there will be a recount of 34 CCLK cycles.

Password Register

PASSWD
Password Register **(086_H)** **Reset Value: Table 584**
RMAP: X, PAGE: X



Field	Bits	Type	Description
MODE	1:0	rw	**Bit-Protection Scheme Control Bit** These two bits cannot be written directly. To change the value between 11_B and 00_B, the bitfield PASS must be written with 11000_B, only then the MODE[1:0] will be registered. 00_B **SCHEME_DIS**, Scheme Disabled 11_B **SCHEME_EN3**, Scheme Enabled **others**, Reserved; do not use
PROTECT_S	2	rh	**Bit-Protection Signal Status Bit** This bit shows the status of the protection. 0_B Software is able to write to all protected bits 1_B Software is unable to write to any protected bits (default)
PASS	7:3	w	**Password Bits** The Bit-Protection Scheme only recognizes three patterns. 13_H **PROT_OPEN**, Opens access to writing of all protected bits 15_H **PROT_CLOSE**, Closes access to writing of all protected bits 18_H **MODE_EN**, Enables writing of the bit field MODE

Table 584 Reset Values of PASSWD

Reset Type	Reset Value	Note
LVD Reset	07_H	
Generated Reset	07_H	

The list of protected bits is shown in **Table 585**.

Table 585 List of Protected Bits

Register	Bit Field
SCU_CMCON	OSCPD
SCU_RSTCON	SWRQ
WDTCON	WDTEN
CNT0 - CNT3 (RTC registers)	All bits in the registers

49.7.6 Revision History

Table 586 Revision History

Reference	Change to Previous Version	
V3.9		
	First Official Release of completely reworked SCR chapter	
	No change	
V4.0		
Page 56	Corrected bullet list item.	

49.8 Watchdog Timer (WDT)

49.8.1 Overview

The Watchdog Timer (WDT) provides a highly reliable and secure way to detect and recover from software or hardware failures. The WDT is reset at a regular interval that is predefined by the user. The CPU must service the WDT within this interval to prevent the WDT from causing generated reset. Hence, routine service of the WDT confirms that the system is functioning properly. This ensures that an accidental malfunction of the SCR will be aborted in a user-specified time period. The WDT reset is default enabled. It can also be disabled via RSTCON register.

The WDT is by default disabled.

In debug mode, the WDT is default suspended and stops counting (its debug suspend bit is default set i.e., MODSUSP.WDTSUSP = 1. Therefore during debugging, there is no need to refresh the WDT.

Features

- 16-bit Watchdog Timer
- Programmable reload value for upper 8 bits of timer
- Programmable window boundary
- Clock source from either the 70 kHz clock or the 100 MHz/DIV clock
- Selectable input frequency of $f_{WDTCLK}/2$ or $f_{WDTCLK}/128$

49.8.2 System Information

This section consist of the system information required to use the WDT.

49.8.2.1 Reset Effects

The Watchdog Timer maintains a counter which must be refreshed or cleared periodically. Otherwise, the counter will overflow and the watchdog reset will be asserted. The assertion of the reset could be disabled by setting bit RSTCON.WDTRSTEN. The occurrence of a WDT reset is indicated by the bit WDTRST in RSTST register.

49.8.2.2 Clocking Configuration

The WDT runs on either the 70 kHz clock or the 100 MHz/DIV clock. Bit WDT_CON.WDTCLK is used to select the WDT clock source, f_{WDTCLK}.

49.8.2.3 Interrupt Events and Assignment

Table 587 shows the non-maskable interrupt node assignment of the WDT interrupt source.

Table 587 WDT Events' Non-maskable Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
WDT Overflow	NMICON.NMIWDT	NMISR.FNMIWDT	73_H

49.8.2.4 Module Suspend Control

The timer in the WDT is by default suspended on entering debug mode. The WDT can be allowed to run in debug mode by clearing the bit WDTSUSP in SFR MODSUSP to 0. Refer to the definition of register MODSUSP.

49.8.3 Functional Description

The Watchdog Timer (WDT) is a 16-bit timer, which is incremented by a count rate of $f_{WDTCLK}/2$ or $f_{WDTCLK}/128$. The f_{WDTCLK} clock input is selectable between the 70 kHz clock or the 100 MHz/DIV clock using the WDTCLK bit. This 16-bit timer is realized as two concatenated 8-bit timers. The upper 8 bits of the WDT can be preset to a user-programmable value via a watchdog service access in order to vary the watchdog expire time. The lower 8 bits are reset on each service access. **Figure 820** shows the block diagram of the WDT unit.

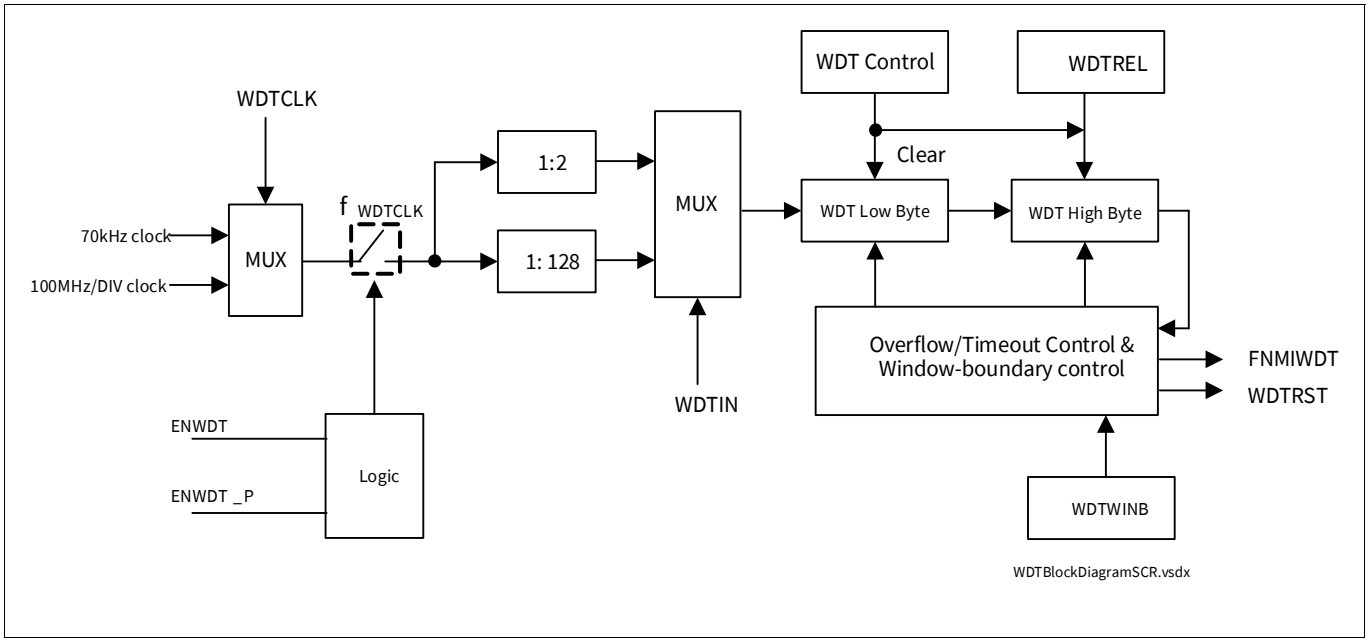


Figure 820 WDT Block Diagram

If the WDT is enabled by setting WDTEN to 1, the timer is set to a user-defined start value and begins counting up. It must be serviced before the counter overflows. Servicing is performed through refresh operation (setting bit WDTRS to 1). This reloads the timer with the start value, and normal operation continues.

If the WDT is not serviced before the timer overflows, a system malfunction is assumed and normal mode is terminated. A WDT NMI request (FNMIWDT) is then asserted and pre-warning is entered. The pre-warning lasts for 30_H count. During the pre-warning period, refreshing of the WDT is ignored and the WDT cannot be disabled. The user software shall not refresh the WDT upon entering the pre-warning period. A reset (WDTRST) of the SCR is imminent (if it is enabled by bit RSTCON.WDTRSTEN) and can no longer be avoided. The occurrence of a WDT reset is indicated by the bit WDTRST in RSTST register. If refresh happens at the same time an overflow occurs, WDT will not go into pre-warning period.

Note: In the SCR, it is possible to wake-up the main controller from standby mode when the WDT overflow event happened by setting bit STDBYWKP.WDTWKSEL to 1.

The WDT must be serviced periodically so that its count value will not overflow. Servicing the WDT clears the low byte and reloads the high byte with the preset value in bit field WDTREL. Servicing the WDT also clears the bit WDTRS.

The WDT has a “programmable window boundary”, which disallows any refresh during the WDT’s count-up. A refresh during this window-boundary constitutes an invalid access to the WDT and causes the WDT to activate WDTRST, although no NMI request is generated in this instance. The window boundary is from 0000_H to the value obtained from the concatenation of WDTWINB and 00_H. This feature can be enabled by WINBEN.

After being serviced, the WDT continues counting up from the value (<WDTREL> * 2^8). The time period for an overflow of the WDT is programmable in two ways:

- The input frequency to the WDT can be selected via bit WDTIN in register WDT_CON to be either $f_{WDTCLK}/2$ or $f_{WDTCLK}/128$.
- The reload value WDTREL for the high byte of WDT can be programmed in register WDT_REL.

The period P_{WDT} between servicing the WDT and the next overflow can be determined by the following formula:

$$P_{WDT} = \frac{2^{(1+<WDTIN>*6)} * (2^{16} - WDTREL * 2^{8})}{f_{PCLK}} \quad (49.1)$$

PWDT_formula.vsd

If the Window-Boundary Refresh feature of the WDT is enabled, the period during which a refresh is allowed is shortened if WDTWINB is greater than WDTREL. See also **Figure 821**. This period can be calculated by the same formula by replacing WDTREL with WDTWINB. In order for this feature to be useful, WDTWINB cannot be smaller than WDTREL.

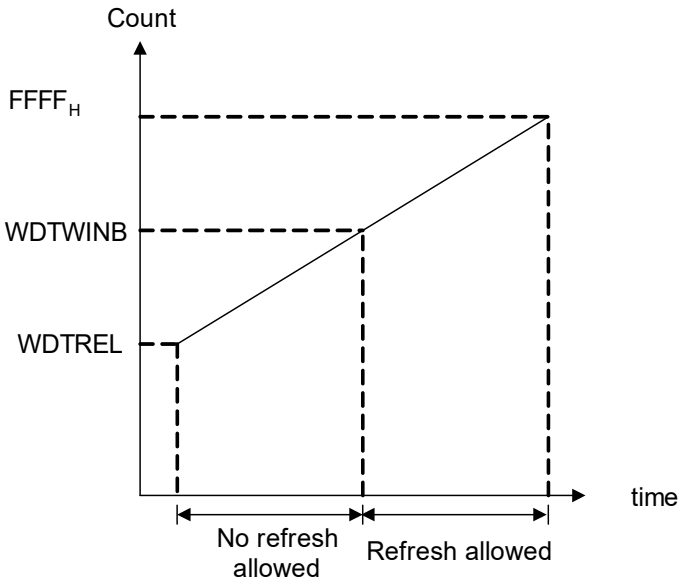


Figure 821 WDT Timing Diagram

Table 588 lists the possible ranges for the watchdog time which can be achieved using a certain module clock. Some numbers are rounded to 3 significant digits.

Table 588 Watchdog Time Ranges

Reload value in WDTREL	Prescaler for f_{WDT}					
	2 (WDTIN = 0)			128 (WDTIN = 1)		
	20 MHz	16 MHz	12 MHz	20 MHz	16 MHz	12 MHz
FF_H	25.5 µs	32.0 µs	42.67 µs	1.63 ms	2.05 ms	2.73 ms
$7F_H$	3.3 ms	4.13 ms	5.5 ms	211 ms	264 ms	352 ms
00_H	6.55 ms	8.19 ms	10.92 ms	419 ms	524 ms	699 ms

Note: For safety reasons, the user is advised to rewrite WDT_CON each time before the WDT is serviced.

Note: The Watchdog Timer can be suspended when OCDS enters Monitor Mode and has the Debug-Suspend signal activated, provided the respective suspend bit, WDTSUSP in SFR MODSUSP, is set. See Module Suspend Control section.

49.8.4 Registers Description

Five SFRs control the operations of the WDT. **Table 589** lists the addresses of these SFRs.

Table 589 Register Overview - WDT (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
WDT_CON	Watchdog Timer Control Register	0	X	081_H	83
WDT_H	Watchdog Timer High Byte	0	X	085_H	86
WDT_L	Watchdog Timer Low Byte	0	X	084_H	85
WDT_REL	Watchdog Timer Reload Register	0	X	082_H	85
WDT_WINB	Watchdog Window-Boundary Register	0	X	083_H	86

49.8.4.1 Watchdog Timer Registers

The Watchdog Timer Current Count Value is contained in the Watchdog Timer Register WDT_H and WDT_L, which are non-bitaddressable read-only register. The operation of the WDT is controlled by its bitaddressable WDT Control Register WDT_CON. This register also selects the input clock prescaling factor. The register WDT_REL specifies the reload value for the high byte of the timer. WDT_WINB specifies Watchdog Window-Boundary count value.

Watchdog Timer Control Register

WDT_CON
Watchdog Timer Control Register **(081_H)** **Reset Value: Table 591**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0		WINBEN	WDTPR	WDTCLK	WDTEN	WDTRS	WDTIN
r		rw	rh	rw	rw	rwh	rw

Field	Bits	Type	Description
WDTIN	0	rw	**Watchdog Timer Input Frequency Selection** 0_B Input frequency is fWDTCLK/2 1_B Input frequency is fWDTCLK/128
WDTRS	1	rwh	**WDT Refresh Start** Active high. Set to start refresh operation on the watchdog timer. Cleared automatically by hardware after it is set by software. Takes 3 wdtclk cycles to take effect. Has higher priority than WDTEN.

Field	Bits	Type	Description
WDTEN	2	rw	**WDT Enable** WDTEN is a protected bit. If the Protection Scheme is activated, then this bit cannot be written directly. See Section "Miscellaneous Control" for details on the protection scheme. Needs 3 wdtclk cycles to take effect. Has lower priority than WDTRS. *Note: Clearing WDTEN bit to 0 during Prewarning Mode (WDTPR = 1) has no effect.* 0_B WDT is disabled 1_B WDT is enabled
WDTCLK	3	rw	**Watchdog Timer Clock Source Selection** 0_B 70 kHz clock is selected 1_B 100 MHz/DIV clock is selected
WDTPR	4	rh	**Watchdog Prewarning Mode Flag** This bit is set to 1 when a Watchdog error is detected. The Watchdog Timer has issued an NMI trap and is in Prewarning Mode. A reset of the chip occurs after the pre-warning period has expired. 0_B Normal mode (default after reset) 1_B The Watchdog is operating in Prewarning Mode
WINBEN	5	rw	**Watchdog Window-Boundary Enable** 0_B Watchdog Window-Boundary feature is disabled (default) 1_B Watchdog Window-Boundary feature is enabled
0	7:6	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 590 Access Mode Restrictions of WDT_CON sorted by descending priority

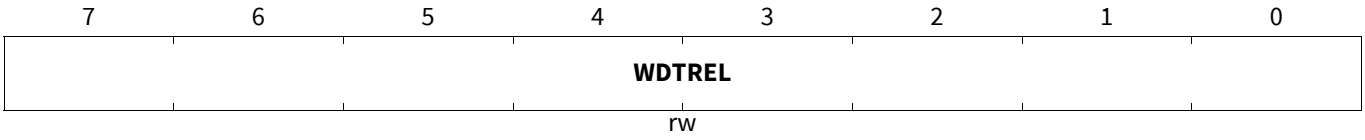
Mode Name	Access Mode		Description
otherwise	r	WDTEN	
PASSWD.PROTECT_S = 0 (default)	rw	WDTEN	

Table 591 Reset Values of WDT_CON

Reset Type	Reset Value	Note
LVD Reset	XX00 0000_B	
Generated Reset	–-00 0000_B	

Watchdog Timer Reload Register

WDT_REL
Watchdog Timer Reload Register **(082_H)** **Reset Value: Table 592**
RMAP: 0, PAGE: X



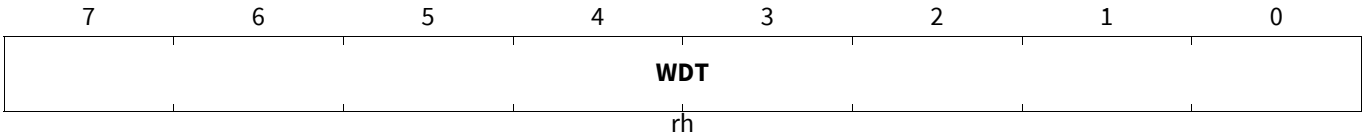
Field	Bits	Type	Description
WDTREL	7:0	rw	**Watchdog Timer Reload Value** Reload value for the high byte of the WDT. After writing WDTREL, it takes 3 fWDTCLK cycles to get WDTREL active. The delay has to be regarded for WDT enable and WDT refresh operation, as otherwise the previous WDTREL value is still active.

Table 592 Reset Values of WDT_REL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Watchdog Timer Low Byte

WDT_L
Watchdog Timer Low Byte **(084_H)** **Reset Value: Table 593**
RMAP: 0, PAGE: X



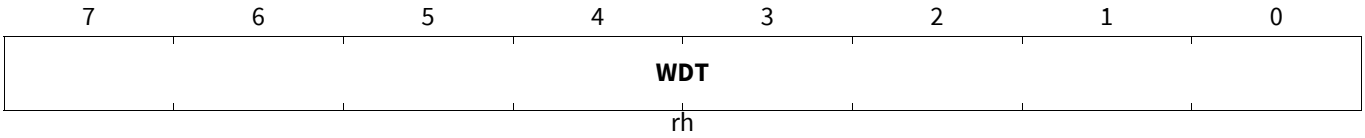
Field	Bits	Type	Description
WDT	7:0	rh	**Watchdog Timer Current Value, low byte**

Table 593 Reset Values of WDT_L

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Watchdog Timer High Byte

WDT_H
Watchdog Timer High Byte **(085_H)** **Reset Value: Table 594**
RMAP: 0, PAGE: X



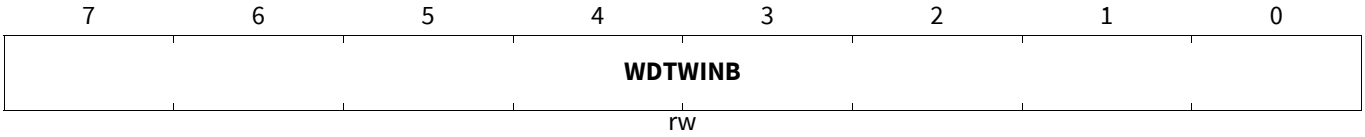
Field	Bits	Type	Description
WDT	7:0	rh	**Watchdog Timer Current Value, high byte**

Table 594 Reset Values of WDT_H

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Watchdog Window-Boundary Register

WDT_WINB
Watchdog Window-Boundary Register **(083_H)** **Reset Value: Table 595**
RMAP: 0, PAGE: X



Field	Bits	Type	Description
WDTWINB	7:0	rw	**Watchdog Window-Boundary Value** This value is programmable. Within this Window-Boundary range from 0000_H to (WDTWINB|00_H), the WDT cannot do a refresh, else it will cause a WDTRST to be asserted. WDTWINB is compared to WDTH.

Table 595 Reset Values of WDT_WINB

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.8.5 Revision History

Table 596 Revision History

Reference	Change to Previous Version	
v2.3		
	First Official Release of completely reworked SCR chapter	
	No change	

49.9 Interrupt System

The XC800 Core supports one non-maskable interrupt (NMI) and 14 maskable interrupt requests. In addition to the standard interrupt functions supported by the core, e.g., configurable interrupt priority and interrupt masking, the SCR interrupt system provides extended interrupt support capabilities such as the mapping of each interrupt vector to several interrupt sources to increase the number of interrupt sources supported, and additional status registers for detecting and identifying the interrupt source.

49.9.1 Interrupt Sources

The SCR supports 14 interrupt vectors with four priority levels. These interrupt vectors are assigned to the on-chip peripherals: Timer 0, Timer 1, UART, RTC, WCAN, and SSC are each assigned one dedicated interrupt vector. T2CCU and LIN interrupts has to share an interrupt vector. External interrupt 0, 1 and 2 are each assigned one dedicated interrupt vector. The remaining 13 external interrupts are shared among 4 interrupt vectors.

A non-maskable interrupt (NMI) with the highest priority is shared by the following:

- Watchdog Timer, warning before overflow
- OCDS, on user IRAM event
- RAM ECC error
- External NMI (via rising edge on EXTNMI pin)
- Main controller - This interrupt is triggered either when software sets PMSWCR2.TCINTREQ or when PMSWCR2.SMURST or PMSWCR2.RST are set.
- Wake-up interrupt: This interrupt is triggered when any of the wakeup inputs were triggered, and the VEXT supply did not ramp up even after the blanking time interval. Please see PMS chapter for additional details

Figure 822, **Figure 823**, and **Figure 824** give a general overview of the interrupt sources and nodes, and their corresponding control and status flags. **Figure 825** gives the corresponding overview for the NMI sources.

The details of the peripheral interrupt sources are described in the respective peripheral chapters, while the details of the external interrupt sources can be found in **Figure 835** through **Figure 838** in this chapter.

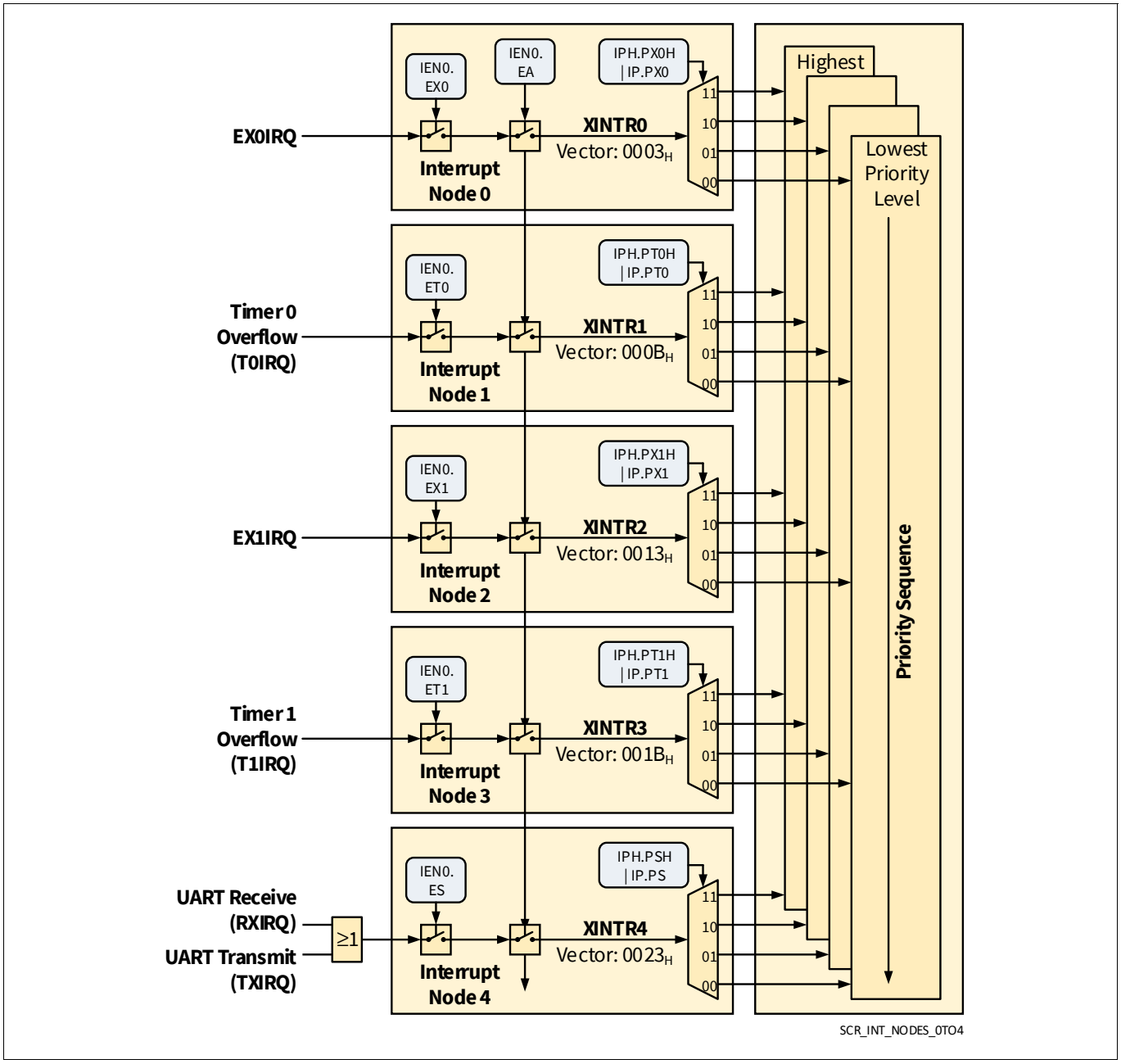


Figure 822 Interrupt Request Sources and Interrupt Nodes 0 through 4

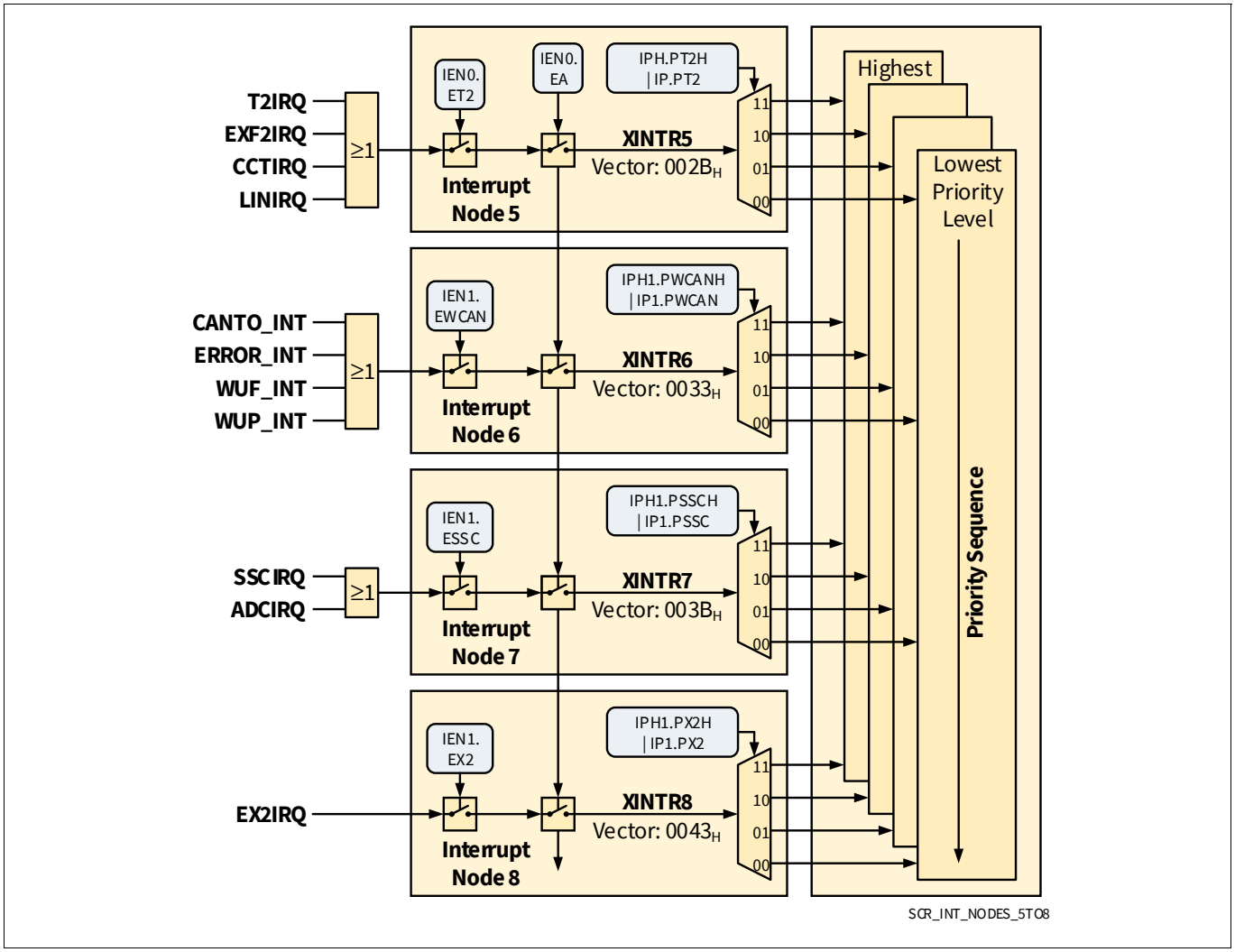


Figure 823 Interrupt Request and Interrupt Nodes 5 through 8

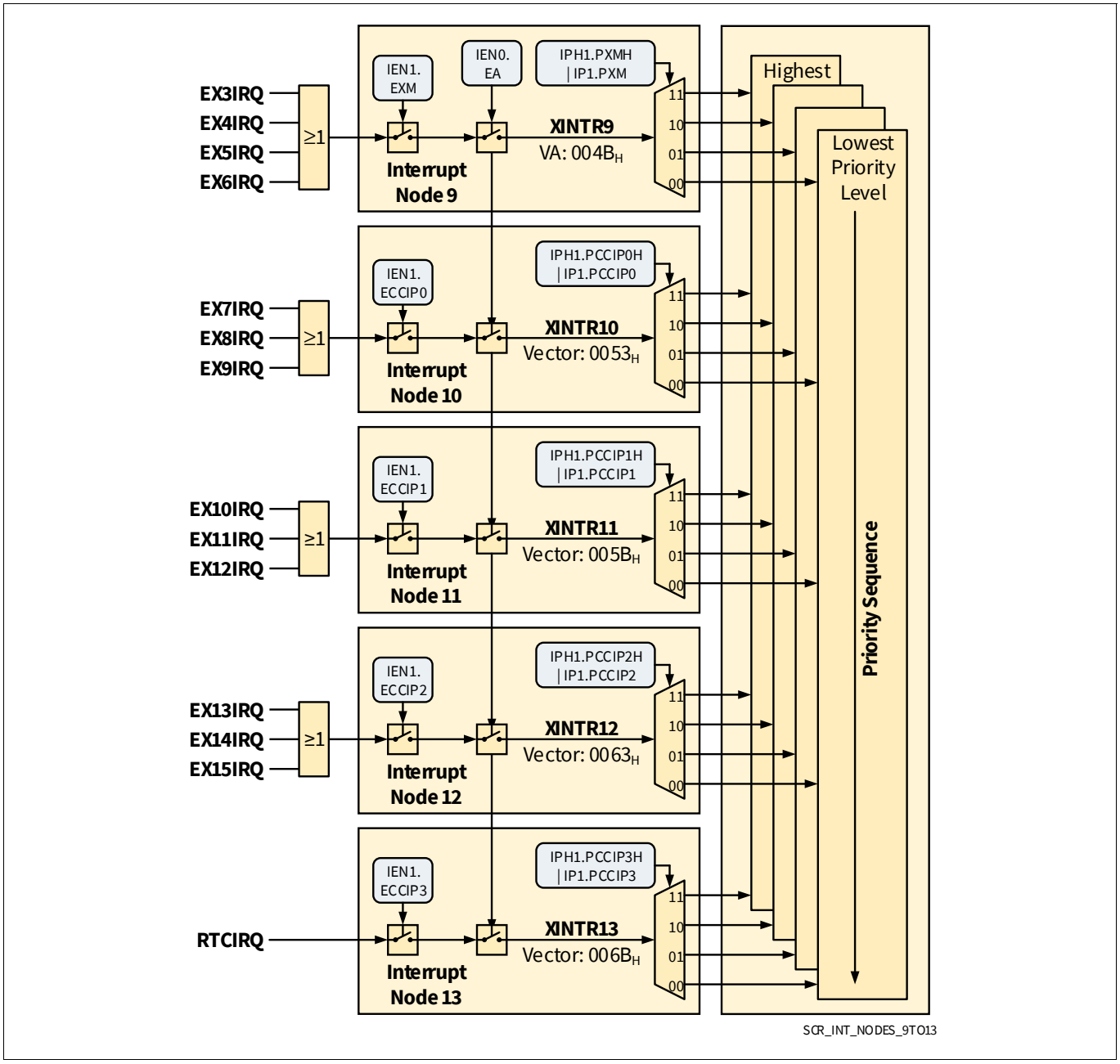


Figure 824 Interrupt Request and Interrupt Nodes 9 through 13

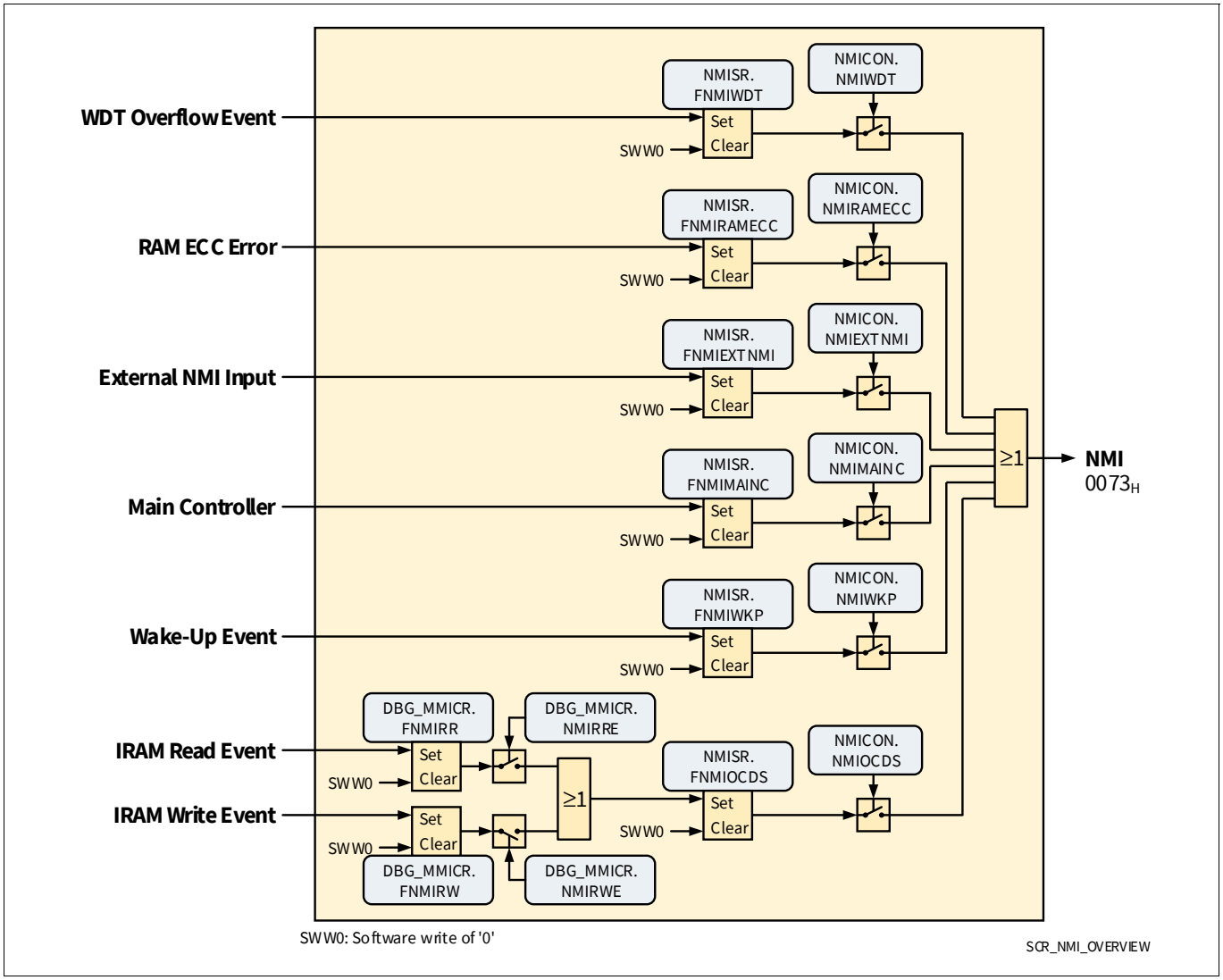


Figure 825 Non-Maskable Interrupt (NMI) Request Sources

49.9.1.1 Interrupt Source and Vector

Each interrupt event source has an associated interrupt vector address for the interrupt node it belongs to. This vector is accessed to service the corresponding interrupt node request. The interrupt service of each interrupt node can be individually enabled or disabled via an enable bit. The assignment of the SCR interrupt sources to the interrupt vector address and the corresponding interrupt node enable bits are summarized in **Table 597**.

Table 597 Interrupt Vector Address

Level	Interrupt Node	Vector Address	Assignment for SCR	Enable Bit	SFR
0	NMI	0073_H	Watchdog Timer NMI	NMIWDT	NMICON
			RAM Double Bit ECC Error NMI	NMIRAMECC	
			Wake-up event NMI	NMIWKP	
			External NMI via EXTNMI pin	NMIEXTNMI	
			OCDS NMI	NMIOCDS	
			Main Controller NMI	NMIMAINC	
1	XINTR0	0003_H	External Interrupt 0	EX0	IEN0
2	XINTR1	$000B_H$	Timer 0	ET0	
3	XINTR2	0013_H	External Interrupt 1	EX1	
4	XINTR3	$001B_H$	Timer 1	ET1	
5	XINTR4	0023_H	UART	ES	
6	XINTR5	$002B_H$	T2CCU	ET2	
			LIN		
7	XINTR6	0033_H	Wake-up CAN filter interrupt	EWCAN	IEN1
8	XINTR7	$003B_H$	ADC	ESSC	
			SSC		
9	XINTR8	0043_H	External Interrupt 2	EX2	
10	XINTR9	$004B_H$	External Interrupt 3	EXM	
			External Interrupt 4		
			External Interrupt 5		
			External Interrupt 6		
11	XINTR10	0053_H	External Interrupt 7	ECCIP0	
			External Interrupt 8		
			External Interrupt 9		
12	XINTR11	$005B_H$	External Interrupt 10	ECCIP1	
			External Interrupt 11		
			External Interrupt 12		
13	XINTR12	0063_H	External Interrupt 13	ECCIP2	
			External Interrupt 14		
			External Interrupt 15		
14	XINTR13	$006B_H$	RTC Interrupt	ECCIP3	

49.9.1.2 Interrupt Source and Priority

An interrupt that is currently being serviced can only be interrupted by a higher priority interrupt, but not by another interrupt of the same or lower priority. Hence, an interrupt of the highest priority cannot be interrupted by any other interrupt request.

If two or more requests of different priority levels are received simultaneously, the request with the highest priority is serviced first. If requests of the same priority are received simultaneously, an internal polling sequence determines which request is serviced first. Thus, within each priority level, there is a second priority structure determined by the polling sequence as shown in **Table 598**.

Table 598 Priority Structure within Interrupt Level

Source	Level
Non-Maskable Interrupt (NMI)	(highest)
External Interrupt 0	1
Timer 0	2
External Interrupt 1	3
Timer 1	4
UART	5
T2CCU / LIN	6
Wake-Up CAN Filter	7
A/D Converter	8
SSC	8
External Interrupt 2	9
External Interrupt [6:3]	10
External Interrupt [9:7]	11
External Interrupt [12:10]	12
External Interrupt [15:13]	13
Real-Time Clock	14

49.9.2 Interrupt Structure

An interrupt event source may be generated from the on-chip peripherals or from external. Detection of interrupt events is controlled by the respective on-chip peripherals. Interrupt status flags are available for determining which interrupt event has occurred, especially useful for an interrupt node which is shared by several event sources. Each interrupt node (except NMI) has a global enable/disable bit. In most cases, additional enable bits are provided for enabling/disabling particular interrupt events (provided for NMI events). No interrupt will be requested for any occurred event that has its interrupt enable bit disabled.

The SCR has, in general, two interrupt structures distinguished mainly by the manner in which the pending interrupt request (one per interrupt vector/node going directly to the core) is generated (due to the events) and cleared.

Common among these two interrupt structures is the interrupt masking bit, EA, which is used to globally enable or disable all interrupt requests (except NMI) to the core. Resetting bit EA to 0 only masks the pending interrupt requests from the core, but does not block the capture of incoming interrupt requests.

Note: The NMI node is similar to the other interrupt nodes, except for the exclusion of EA bit. Effectively, NMI node is non-maskable.

49.9.2.1 Interrupt Structure 1

For interrupt structure 1 (see **Figure 826**), the interrupt event will set the interrupt status flag which doubles as a pending interrupt request to the core. An active pending interrupt request will interrupt the core only if its corresponding interrupt node is enabled. Once an interrupt node is serviced (interrupt acknowledged), its pending interrupt request (represented by the interrupt status flag) may be automatically cleared by hardware (the core).

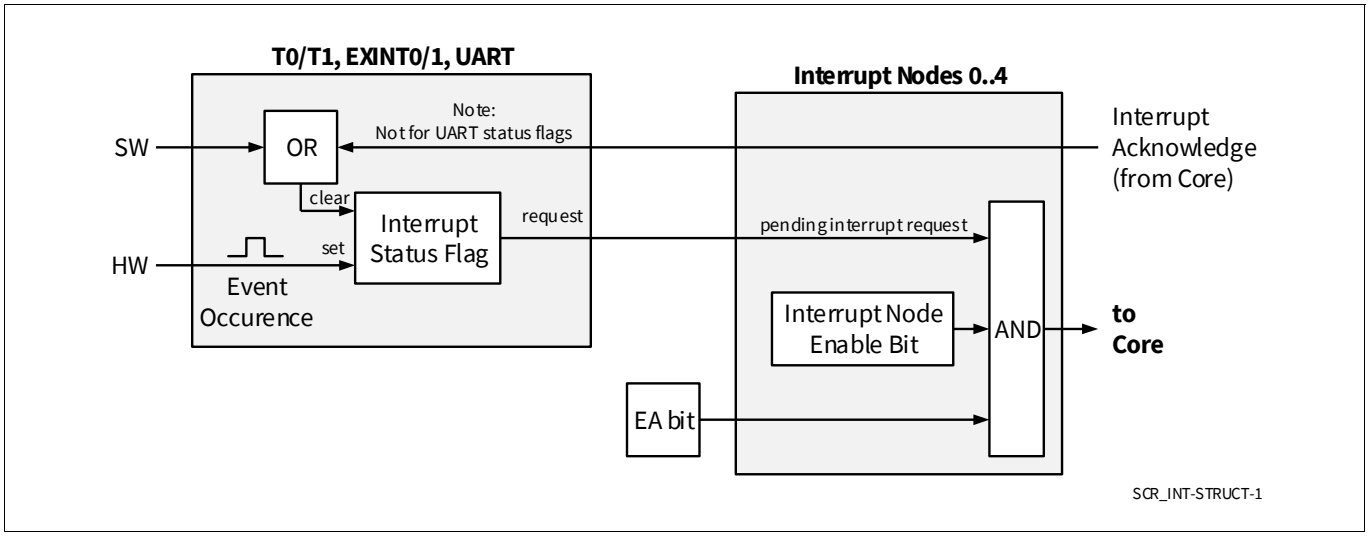


Figure 826 Interrupt Structure 1

For the SCR, interrupt sources of Timer 0, Timer 1, external interrupt 0 and external interrupt 1 (each having a dedicated interrupt node) will have their respective interrupt status flags TF0, TF1, IE0 and IE1 in register T01_TCON cleared by the core, once their corresponding pending interrupt request is serviced. In the case that an interrupt node is disabled (e.g. software polling is used), its interrupt status flag must be cleared by software, since the core will not be interrupted (and therefore the interrupt acknowledge is not generated).

For the UART which has its dedicated interrupt node, interrupt status flags RI and TI in register SCON will NOT be cleared by the core, even when its pending interrupt request is serviced. The UART interrupt status flags (and hence the pending interrupt request) can only be cleared by software.

49.9.2.2 Interrupt Structure 2

Interrupt structure 2 is implemented for interrupt nodes 5 through 13, relevant for the T2CCU, LIN, external interrupts 2 to 15, WCAN, SSC and RTC interrupt sources. For interrupt structure 2, the interrupt status flag does not directly drive the pending interrupt request. Instead, the interrupt nodes have a separate internal pending flag (FF).

An event generated by an associated interrupt source will set the internal pending flag. When the node is enabled, and the CPU is ready to service that interrupt request, the pending flag is automatically cleared by hardware (interrupt acknowledged) on entry into the service routine.

However, the status flag of the associated interrupt source is not automatically cleared; it must be cleared by software (preferably in the service routine).

Most of the interrupt nodes 5 through 13 have more than one associated interrupt source. The request signals of the sources are ORed for the set input of the pending flag. The clear input of the pending flag is either activated by the interrupt acknowledge, or when the status flags of ALL associated interrupt sources are cleared (inactive).

If one wants to clear the internal pending flag in an interrupt node with structure 2, special considerations need to be taken into account, depending on the associated interrupt source(s).

The interrupt sources differ slightly in the configuration of the request and the status lines. This is detailed in the following.

Interrupt Nodes 8 through 12

Figure 827 illustrates the configuration (indicated as Type a) for interrupt nodes 8 through 12, which are connected to the external interrupts EXINT2 through EXINT15. All interrupt sources connected to these nodes have the same configuration (Type a). These external interrupts do not have a separate interrupt enable control; instead, an event is disabled by selecting the '0'-input of its edge-detection multiplexer (see **Figure 836** through **Figure 838**).

Thus, in order to clear the pending FF in the interrupt node, all associated status flags connected to that node must be cleared by software.

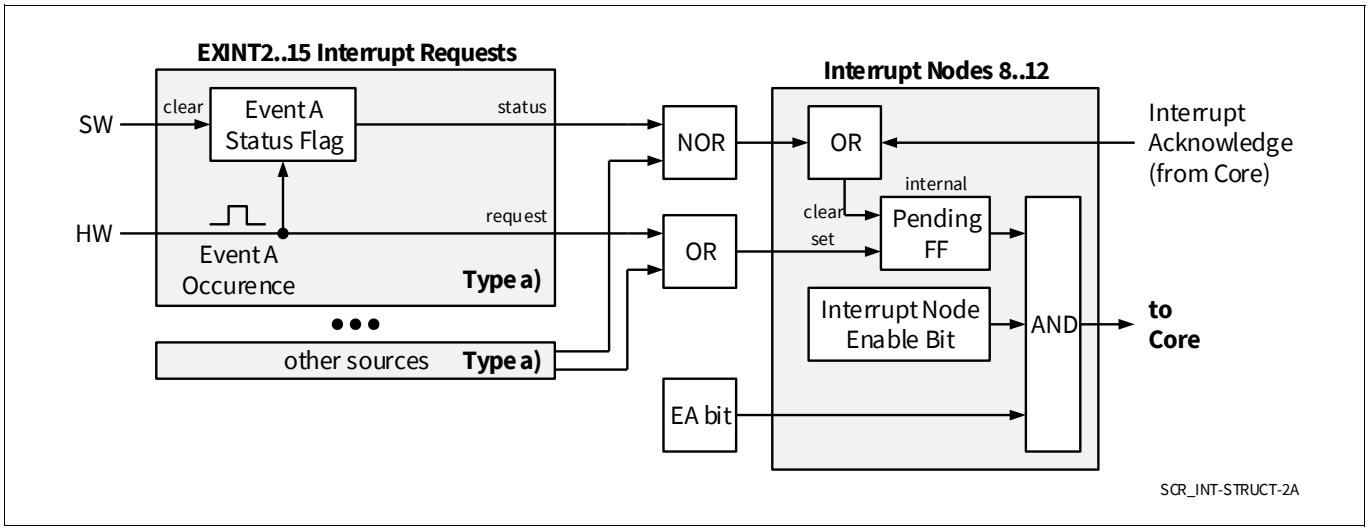


Figure 827 Interrupt Structure 2 for Nodes 8 through 12

Interrupt Node 5

Figure 828 illustrates the configurations for interrupt node 5. Here, two different types of interrupt source configurations (indicated as Type b and Type c) are connected to the same node.

- For the LIN and the CCTIRQ (of T2CCU) interrupt sources, the request line as well as the status line are enabled/disabled via the associated interrupt enable bit. Thus, when disabling the interrupt source, also the status line to the clear input of the interrupt node's pending flag FF is deactivated.
- For the two interrupt sources T2IRQ and EXF2IRQ of the T2CCU, only the request line is enabled/disabled via the associated interrupt enable bit. Thus, when disabling the interrupt source, the status bit needs to be cleared by software in order to be able to clear the pending flag in the interrupt node.

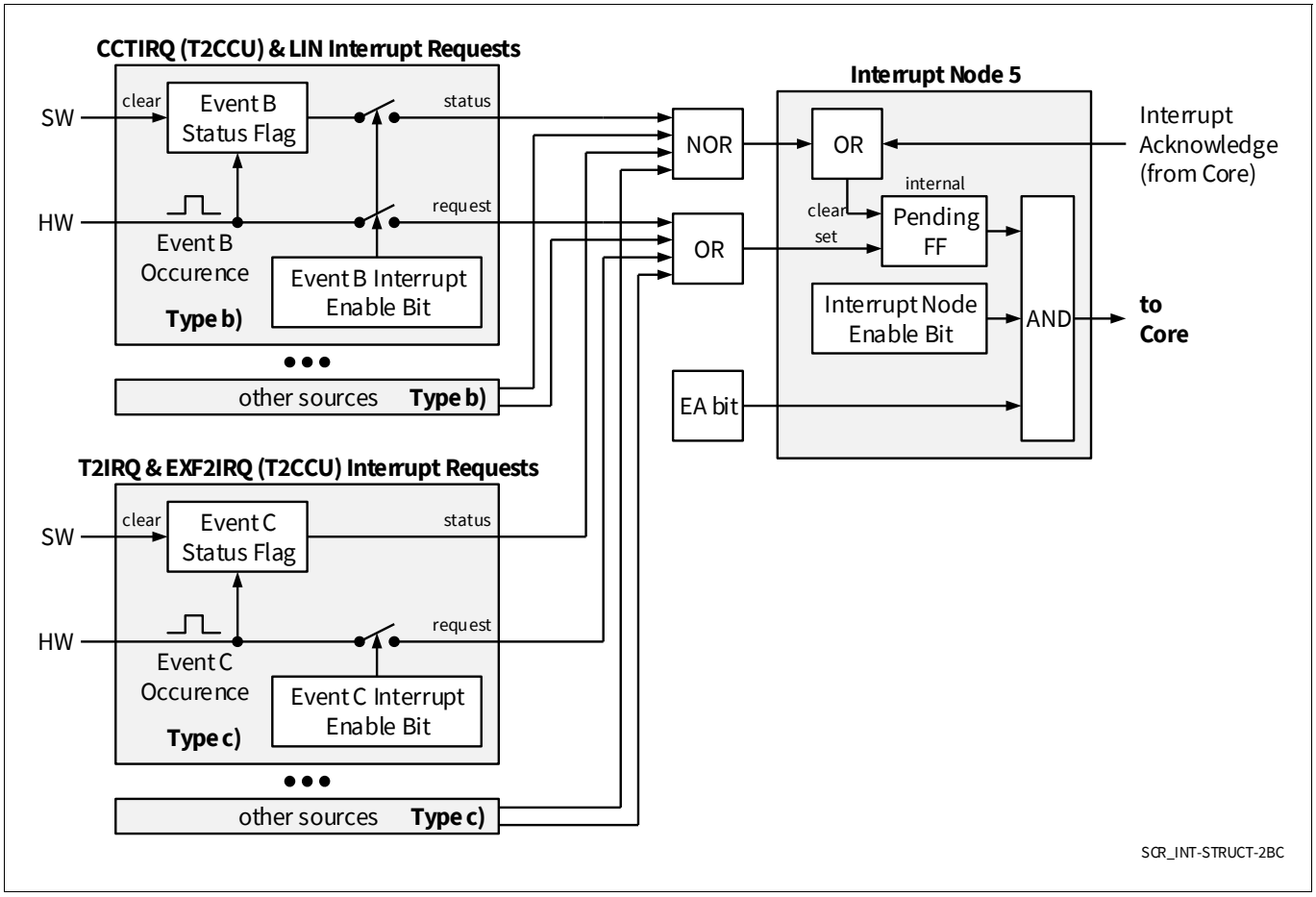


Figure 828 Interrupt Structure 2 for Node 5

Interrupt Node 6

Figure 829 illustrates the configuration for interrupt node 6, which is connected to the WCAN interrupt sources. Only the request line is enabled/disabled via the associated interrupt enable bit. Thus, when disabling the interrupt source, all active status bits of the WCAN module need to be cleared by software in order to be able to clear the pending flag in the interrupt node.

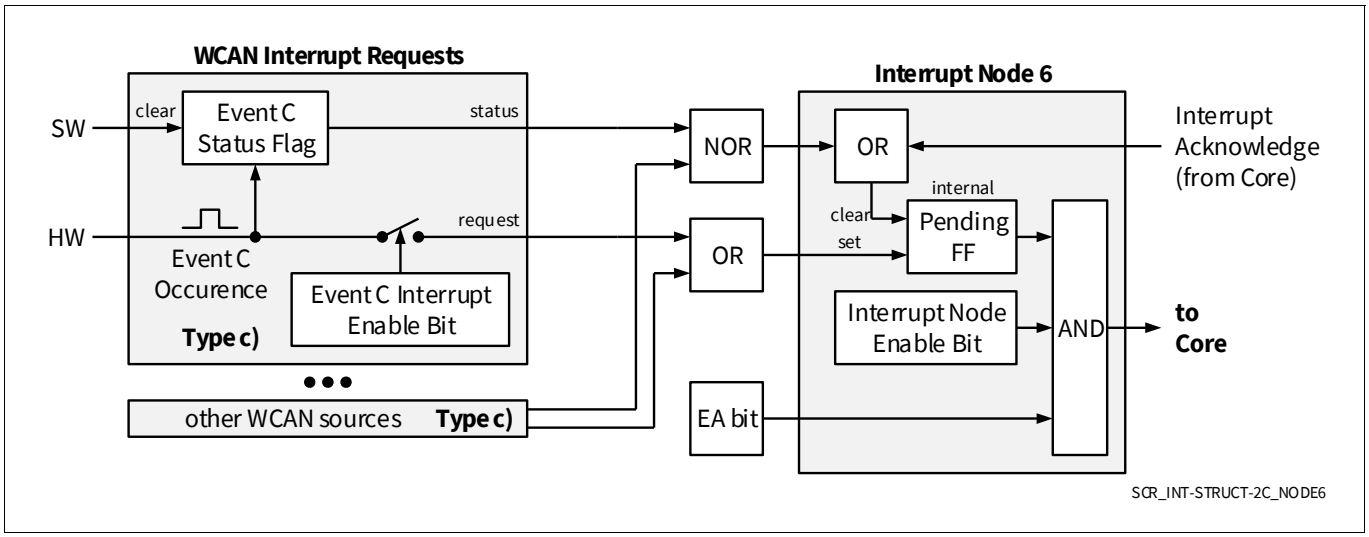


Figure 829 Interrupt Structure 2 for Node 6

Interrupt Node 13

Figure 830 illustrates the configuration for interrupt node 13, which is solely connected to the RTC interrupt source. Only the request line is enabled/disabled via the associated interrupt enable bit. Thus, when disabling the interrupt source, the status bit needs to be cleared by software in order to be able to clear the pending flag in the interrupt node.

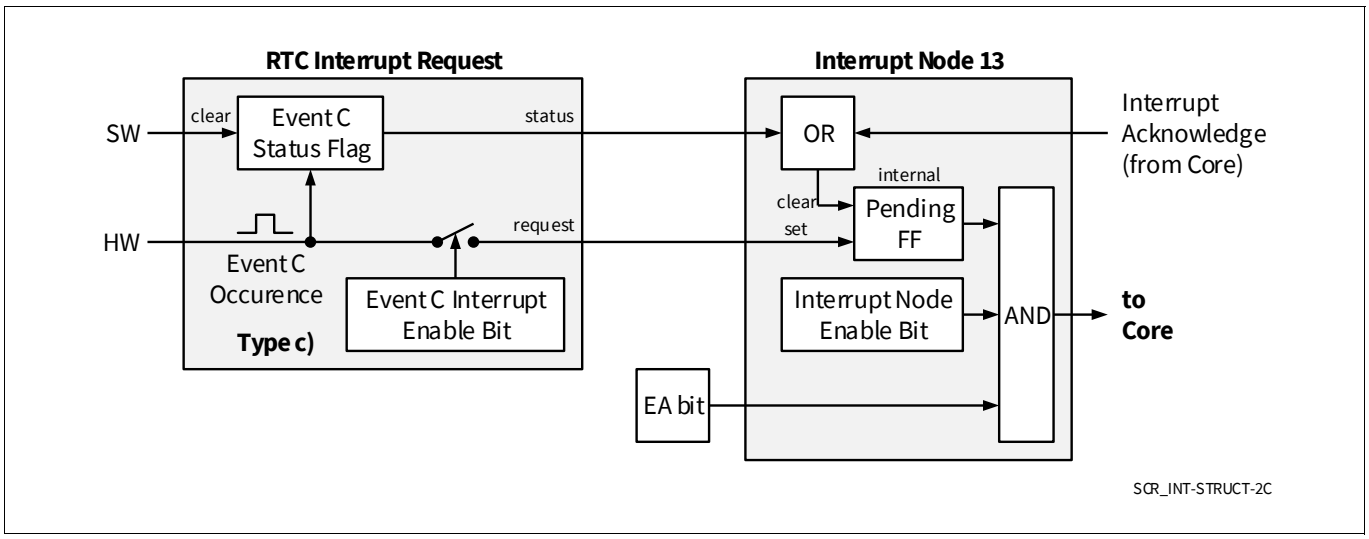


Figure 830 Interrupt Structure 2 for Node 13

Interrupt Node 7

Figure 831 illustrates the configuration for interrupt node 7, which is connected to the SSC and ADCOMP interrupt sources. These interrupt sources are handled by a separate block in the interrupt module, controlled by registers SCU_MODIEN, IRCON1, and IEN1. Here, the status flag is only set when the interrupt event is enabled.

In order to clear the pending FF in the interrupt node, all associated status flags connected to that node must be cleared by software.

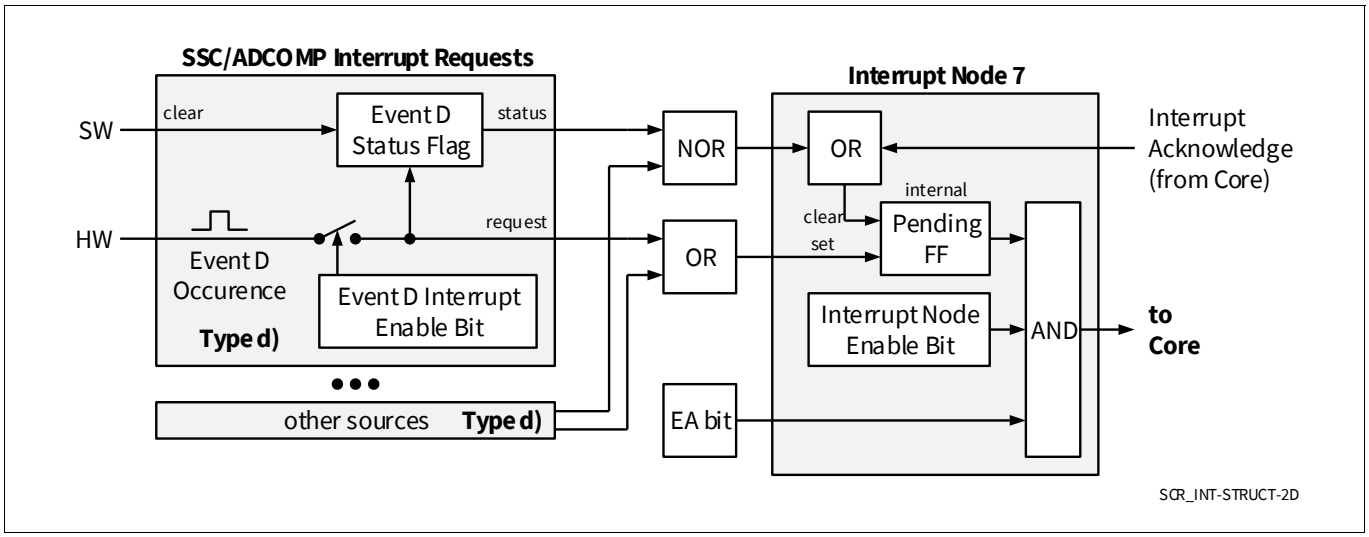


Figure 831 Interrupt Structure 2 for Node 7

49.9.3 Interrupt Handling

The interrupt request signals are sampled at phase 2 in each CPU cycle. The sampled requests are then polled during the following CPU cycle. If one interrupt node request was active at phase 2 of the preceding cycle, the

polling cycle will find it and the interrupt system will generate a LCALL to the node's service routine, provided this hardware-generated LCALL is not blocked by any of the following conditions:

1. An interrupt of equal or higher priority is already in progress.
2. The current (polling) cycle is not in the final cycle of the instruction in progress.
3. The instruction in progress is RETI or any write access to registers IEN0/IEN1 or IP/IPH or IP1/IPH1.

Any of these three conditions will block the generation of the LCALL to the interrupt service routine. Condition 2 ensures that the instruction in progress is completed before vectoring to any service routine. Condition 3 ensures that if the instruction in progress is RETI or any write access to registers IEN0/IEN1 or IP/IPH or IP1/IPH1, then at least one more instruction will be executed before any interrupt is vectored to; this delay guarantees that changes of the interrupt status can be observed by the CPU.

The polling cycle is repeated with each CPU cycle, and the values polled are the values that were present at phase 2 of the previous CPU cycle. Note that if any interrupt flag is active but its node interrupt request was not responded to for one of the conditions already mentioned, and if the flag is no longer active at a later time when servicing the interrupt node, the corresponding interrupt source will not be serviced. In other words, the fact that the interrupt flag was once active but not serviced is not remembered. Every polling cycle interrogates only the pending interrupt requests.

The processor acknowledges an interrupt request by executing a hardware generated LCALL to the corresponding service routine. In some cases, hardware also clears the flag that generated the interrupt, while in other cases, the flag must be cleared by the user's software. The hardware-generated LCALL pushes the contents of the Program Counter (PC) onto the stack (but it does not save the PSW) and reloads the PC with an address that depends on the interrupt node being vectored to, as shown in the **Table 597**.

Program execution returns to the next instruction after calling the interrupt when the RETI instruction is encountered. The RETI instruction informs the processor that the interrupt routine is no longer in progress, then pops the two top bytes from the stack and reloads the PC. Execution of the interrupted program continues from the point where it was stopped. Note that the RETI instruction is important because it informs the processor that the program has left the current interrupt priority level. A simple RET instruction would also have returned execution to the interrupted program, but it would have left the interrupt control system on the assumption that an interrupt was still in progress. In this case, no interrupt of the same or lower priority level would be acknowledged.

49.9.4 Interrupt Response Time

Due to an interrupt event of (the various sources of) an interrupt node, its corresponding request signal will be sampled active at phase 2 in every CPU cycle. The value is not polled by the circuitry until the next CPU cycle. If the request is active and conditions are right for it to be acknowledged, a hardware subroutine call to the requested service routine will be the next instruction to be executed. The call itself takes two CPU cycles. Thus, a minimum of three complete CPU cycles will elapse from activation of the interrupt request to the beginning of execution of the first instruction of the service routine as shown in **Figure 832**.

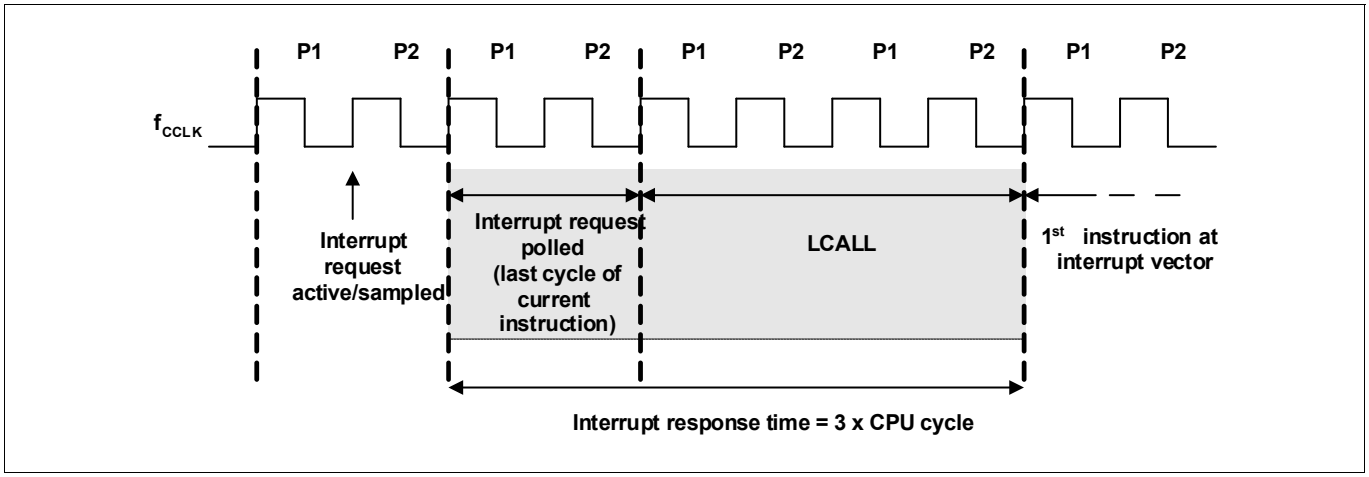


Figure 832 Minimum Interrupt Response Time

A longer response time would be obtained if the request is blocked by one of the three previously listed conditions:

1. If an interrupt of equal or higher priority is already in progress, the additional wait time will depend on the nature of the other interrupt's service routine.
2. If the instruction in progress is not in its final cycle, the additional wait time cannot be more than three CPU cycles since the longest instructions (MUL and DIV) are only four CPU cycles long. See **Figure 833**.
3. If the instruction in progress is RETI or a write access to registers IEN0, IEN1 or IP(H), IP1(H), the additional wait time cannot be more than five cycles (a maximum of one more CPU cycle to complete the instruction in progress, plus four CPU cycles to complete the next instruction, if the instruction is MUL or DIV). See **Figure 834**.

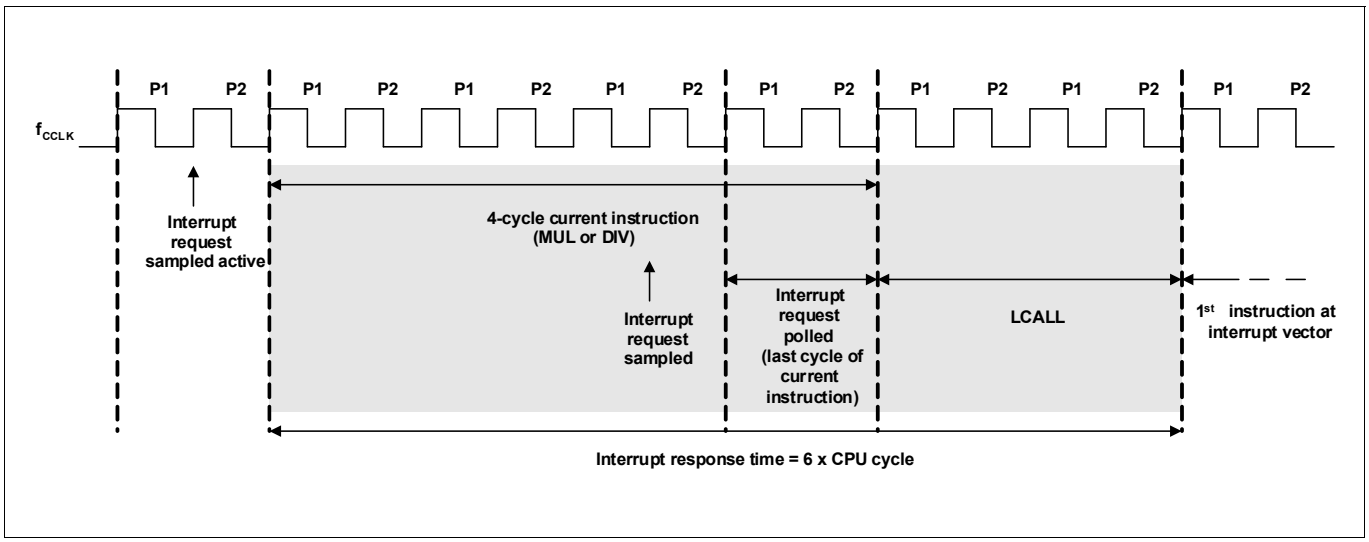


Figure 833 Interrupt Response Time for Condition 2

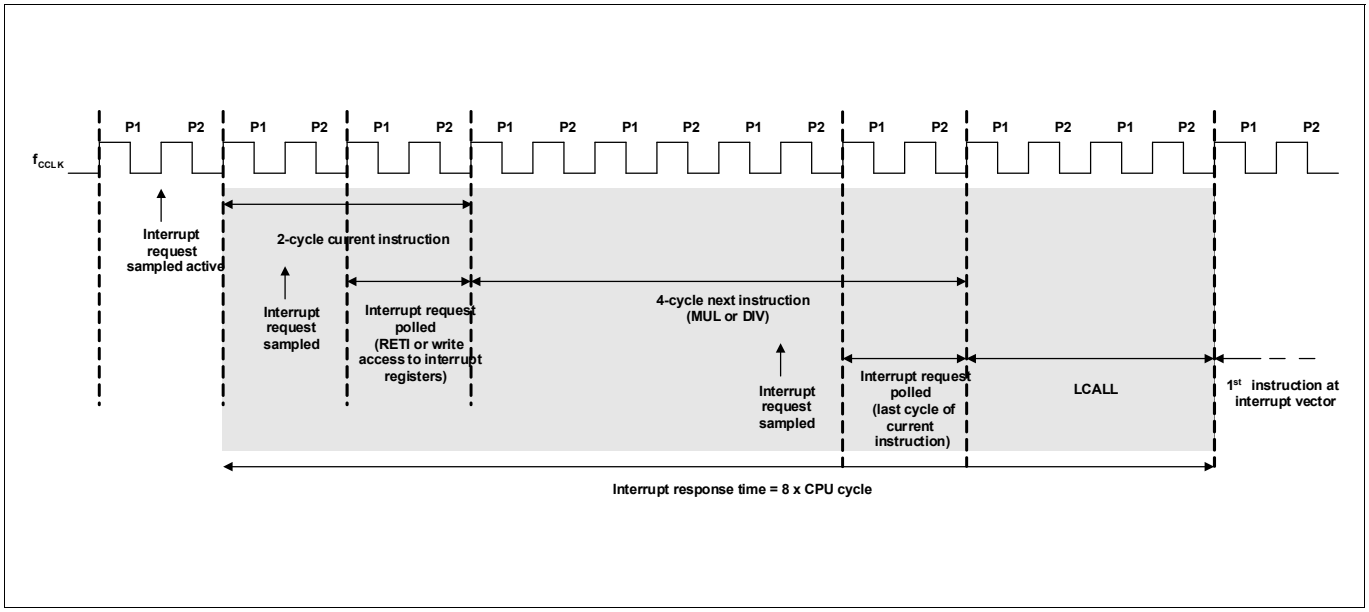


Figure 834 Interrupt Response Time for Condition 3

Thus in a single interrupt system, the response time is always more than three CPU cycles and less than nine CPU cycles (wait states are not considered). When considering wait states, the interrupt response time will be extended depending on the user instructions (also the hardware generated LCALL) being executed during the interrupt response time (shaded region in **Figure 833** and **Figure 834**).

49.9.5 Interrupt from/to TriCore CPUx

There is an interrupt signal from main SoC TriCore CPUx to trigger an 8-bit NMI event. It is triggered by setting a bit in register (PMSWCR2.TCINTREQ) that is located in the main SCU. The interrupt data exchange register, TCINTEXCHG, is used to indicate the type of SOC events that causes this interrupt.

In SCR, there is also an interface to trigger an interrupt to CPUx. This is done via bit NMICON.SCRINTTC. The type of SCR events that causes this interrupt to CPUx can be stored in register SCRINTEXCHG.

49.9.6 Registers Description

Interrupt Special Function Registers or bits are used for interrupt configuration such as node enable, external interrupt control, interrupt flags and interrupt priority setting.

Table 599 lists the SFRs and corresponding addresses.

Table 599 Register Map

Address	Register
RMAP = 0 or 1	
88_H	SYSCON0
$D8_H$	IEN0
$D1_H$	IEN1
DC_H	IP
$D2_H$	IPH
DB_H	IP1
$D3_H$	IPH1

Table 599 Register Map (cont'd)

Address	Register
$C9_H$	T01_TCON
BA_H	UART_SCON
RMAP = 0, SCU Page = 0	
$F2_H$	IRCON0
$F3_H$	IRCON1
$F4_H$	IRCON2
RMAP = 0, SCU Page = 1	
$F2_H$	NMISR
$F3_H$	NMICON
$F4_H$	EXICON0
$F5_H$	EXICON1
$F6_H$	EXICON2
$F7_H$	EXICON3
FA_H	SCU_MODIEN

49.9.6.1 Interrupt Node Enable Registers

Each interrupt node can be individually enabled or disabled by setting or clearing the corresponding bit in the interrupt enable registers IEN0 and IEN1. Register IEN0 also contains the global interrupt masking bit (EA), which can be cleared to block all pending interrupt requests at once.

The NMI interrupt vector is shared by a number of sources, each of which can be enabled or disabled individually via register NMICON.

After reset, the enable bits in IEN0, IEN1 and NMICON are cleared to 0. This implies that all interrupt nodes are disabled by default.

Interrupt Enable Register 0

IEN0
Interrupt Enable Register 0 **($0D8_H$)** **Reset Value: Table 600**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
EA	**0**	**ET2**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**
rw	r	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
EX0	0	rw	**Interrupt Node XINTR0 Enable** 0_B XINTR0 is disabled 1_B XINTR0 is enabled
ET0	1	rw	**Interrupt Node XINTR1 Enable** 0_B XINTR1 is disabled 1_B XINTR1 is enabled

Field	Bits	Type	Description
EX1	2	rw	**Interrupt Node XINTR2 Enable** 0_B XINTR2 is disabled 1_B XINTR2 is enabled
ET1	3	rw	**Interrupt Node XINTR3 Enable** 0_B XINTR3 is disabled 1_B XINTR3 is enabled
ES	4	rw	**Interrupt Node XINTR4 Enable** 0_B XINTR4 is disabled 1_B XINTR4 is enabled
ET2	5	rw	**Interrupt Node XINTR5 Enable** 0_B XINTR5 is disabled 1_B XINTR5 is enabled
EA	7	rw	**Global Interrupt Mask** 0_B All pending interrupt requests (except NMI) are blocked from the core 1_B Pending interrupt requests are not blocked from the core
0	6	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 600 Reset Values of IEN0

Reset Type	Reset Value	Note
Generated Reset	$0X00\ 0000_B$	
LVD Reset	$0–00\ 0000_B$	

Interrupt Enable Register 1

IEN1
Interrupt Enable Register 1 **($0D1_H$)** **Reset Value: Table 601**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
ECCIP3	**ECCIP2**	**ECCIP1**	**ECCIP0**	**EXM**	**EX2**	**ESSC**	**EWCAN**
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
EWCAN	0	rw	**Interrupt Node XINTR6 Enable** 0_B XINTR6 is disabled 1_B XINTR6 is enabled
ESSC	1	rw	**Interrupt Node XINTR7 Enable** 0_B XINTR7 is disabled 1_B XINTR7 is enabled
EX2	2	rw	**Interrupt Node XINTR8 Enable** 0_B XINTR8 is disabled 1_B XINTR8 is enabled

Field	Bits	Type	Description
EXM	3	rw	**Interrupt Node XINTR9 Enable** 0_B XINTR9 is disabled 1_B XINTR9 is enabled
ECCIP0	4	rw	**Interrupt Node XINTR10 Enable** 0_B XINTR10 is disabled 1_B XINTR10 is enabled
ECCIP1	5	rw	**Interrupt Node XINTR11 Enable** 0_B XINTR11 is disabled 1_B XINTR11 is enabled
ECCIP2	6	rw	**Interrupt Node XINTR12 Enable** 0_B XINTR12 is disabled 1_B XINTR12 is enabled
ECCIP3	7	rw	**Interrupt Node XINTR13 Enable** 0_B XINTR13 is disabled 1_B XINTR13 is enabled

Table 601 Reset Values of IEN1

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

NMI Control Register

NMICON
NMI Control Register **($0F3_H$)** **Reset Value: Table 602**
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
SCRINTTC	**NMIWKP**	**0**	**NMIMAINC**	**NMIOCDS**	**NMIEXTNMI**	**NMIRAMECC**	**NMIWDT**
rwh	rw	r	rw	rw	rw	rw	rw

Field	Bits	Type	Description
NMIWDT	0	rw	**Watchdog Timer NMI Enable** 0_B WDT NMI is disabled 1_B WDT NMI is enabled
NMIRAMECC	1	rw	**RAM Double bit ECC Error NMI Enable** 0_B RAM double bit ECC error NMI is disabled 1_B RAM double bit ECC error NMI is enabled

Field	Bits	Type	Description
NMIEXTNMI	2	rw	**External NMI (via EXTNMI pin) NMI Enable** *Note: It is recommended to clear the NMISR.FNMIEXTNMI flag before enabling external NMI.* 0_B External NMI (via EXTNMI pin) is disabled 1_B External NMI (via EXTNMI pin) is enabled
NMIOCDS	3	rw	**OCDS NMI Enable** 0_B OCDS NMI is disabled 1_B OCDS NMI is enabled
NMIMAINC	4	rw	**Main Controller NMI Enable** 0_B Main Controller NMI is disabled 1_B Main Controller NMI is enabled
NMIWKP	6	rw	**Wake-Up NMI Enable** 0_B Wake-Up NMI is disabled 1_B Wake-Up NMI is enabled
SCRINTTC	7	rwh	**CPUx interrupt Enable** Trigger an interrupt to CPUx. This bit will be reset automatically to 0. 0_B No interrupt event is triggered 1_B A CPUx interrupt is triggered
0	5	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 602 Reset Values of NMICON

Reset Type	Reset Value	Note
LVD Reset	$00X0\ 0000_B$	
Generated Reset	$00–0\ 0000_B$	

49.9.6.2 External Interrupt Control Registers

The 16 external interrupts, EXINT[15:0], are driven into the SCR from the ports. The external interrupt pin assignment for SCR is shown in **Table 603**. External interrupts can be positive, negative, or double-edge triggered. Registers EXICON0 through EXICON3 specify the active edge for an external interrupt. Among the external interrupts, external interrupt 0 and external interrupt 1 can be selected to bypass edge detection in the SCU, for direct feed-through to the core. The signal to the core can be further programmed to either low-level or negative transition activated, by bits IT0 and IT1 in the TCON register. However, for edge detection in SCU, T01_TCON.IT0/1 must be set to falling edge triggered. A detected active edge event will generate internally two CCLK cycle low pulse for detection by the core.

If the external interrupt is positive (negative) edge triggered, the external source must hold the request pin low (high) for at least one CCLK cycle, and then hold it high (low) for at least one CCLK cycle to ensure that the transition is recognized. If edge detection is bypassed for external interrupt 0 and/or external interrupt 1, the external source must hold the request pin “high” or “low” for at least two CCLK cycles.

External interrupts 3 to 15 share 4 interrupt nodes. Therefore, in addition to the corresponding interrupt node enable, each external interrupt 3 to 15 may be disabled individually; they are disabled by default after reset.

Table 603 External interrupt [15:0] Pin Functions and Selection (Attention SCR ports)

Pin	Function	Description	Selected By
SCR_P01.1	EXINT0A	External Interrupt Input 0	MODPISEL1.EXINT0IS = 00_B
SCR_P00.4	EXINT0B		MODPISEL1.EXINT0IS = 01_B
SCR_P01.4	EXINT0C		MODPISEL1.EXINT0IS = 10_B
SCR_P01.2	EXINT0D		MODPISEL1.EXINT0IS = 11_B
SCR_P00.1	EXINT1A	External Interrupt Input 1	MODPISEL1.EXINT1IS = 00_B
SCR_P00.5	EXINT1B		MODPISEL1.EXINT1IS = 01_B
SCR_P01.5	EXINT1C		MODPISEL1.EXINT1IS = 10_B
ESR1[1)]	EXINT1D		MODPISEL1.EXINT1IS = 11_B
SCR_P00.2	EXINT2A	External Interrupt Input 2	MODPISEL1.EXINT2IS = 00_B
SCR_P00.6	EXINT2B		MODPISEL1.EXINT2IS = 01_B
SCR_P01.6	EXINT2C		MODPISEL1.EXINT2IS = 10_B
P14.1[1)]	EXINT2D		MODPISEL1.EXINT2IS = 11_B
SCR_P00.0	EXINT3A	External Interrupt Input 3	MODPISEL2.T2CC0EXINT3IS = 00_B
SCR_P01.0	EXINT3B		MODPISEL2.T2CC0EXINT3IS = 01_B
SCR_P00.6	EXINT3C		MODPISEL2.T2CC0EXINT3IS = 10_B
SCR_P00.1	EXINT4A	External Interrupt Input 4	MODPISEL2.T2CC1EXINT4IS = 00_B
SCR_P01.1	EXINT4B		MODPISEL2.T2CC1EXINT4IS = 01_B
SCR_P00.7	EXINT4C		MODPISEL2.T2CC1EXINT4IS = 10_B
SCR_P00.2	EXINT5A	External Interrupt Input 5	MODPISEL2.T2CC2EXINT5IS = 00_B
SCR_P01.2	EXINT5B		MODPISEL2.T2CC2EXINT5IS = 01_B
SCR_P01.6	EXINT5C		MODPISEL2.T2CC2EXINT5IS = 10_B
SCR_P00.3	EXINT6A	External Interrupt Input 6	MODPISEL2.T2CC3EXINT6IS = 00_B
SCR_P01.3	EXINT6B		MODPISEL2.T2CC3EXINT6IS = 01_B
SCR_P01.7	EXINT6C		MODPISEL2.T2CC3EXINT6IS = 10_B

Table 603 External interrupt [15:0] Pin Functions and Selection (Attention SCR ports) (cont'd)

Pin	Function	Description	Selected By
SCR_P00.4	EXINT7	External Interrupt Input 7	-
SCR_P00.5	EXINT8	External Interrupt Input 8	-
SCR_P00.6	EXINT9	External Interrupt Input 9	-
SCR_P00.7	EXINT10	External Interrupt Input 10	-
SCR_P01.4	EXINT11	External Interrupt Input 11	-
SCR_P01.5	EXINT12	External Interrupt Input 12	-
SCR_P01.6	EXINT13	External Interrupt Input 13	-
SCR_P01.7	EXINT14	External Interrupt Input 14	-
P14.1[1)]	EXINT15	External Interrupt Input 15	-

1) The pad control of this pin is in the 32-bit Ports module (via PMS). The input function is available after reset, this pin can be used as external interrupt input.

Note: Several external interrupts support alternative input signals. When switching inputs, the active edge/level trigger select and the level on the associated pins should be considered to prevent unintentional interrupt generation.

The following figures show block diagrams and connection overviews of External Interrupts 0 through 15.

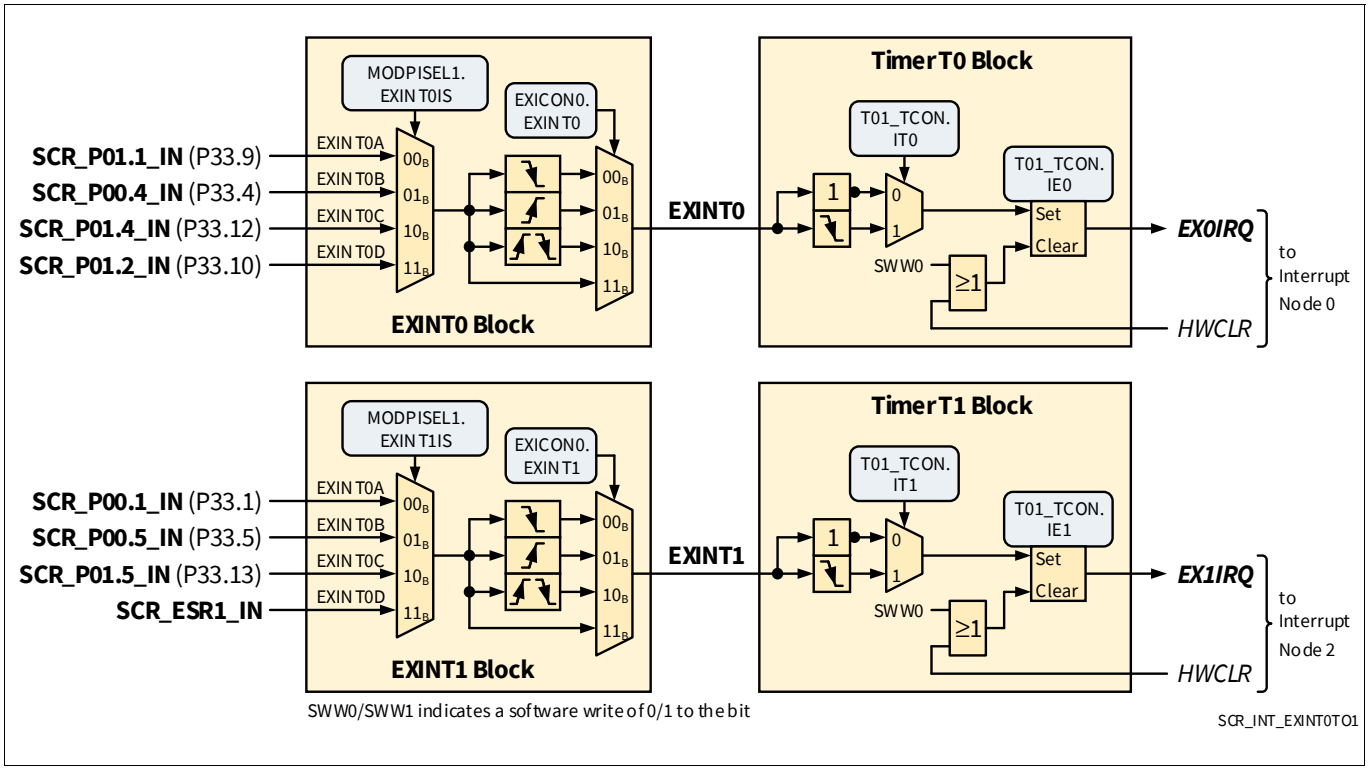


Figure 835 Block Diagram and Connections of External Interrupts EXINT0 and EXINT1

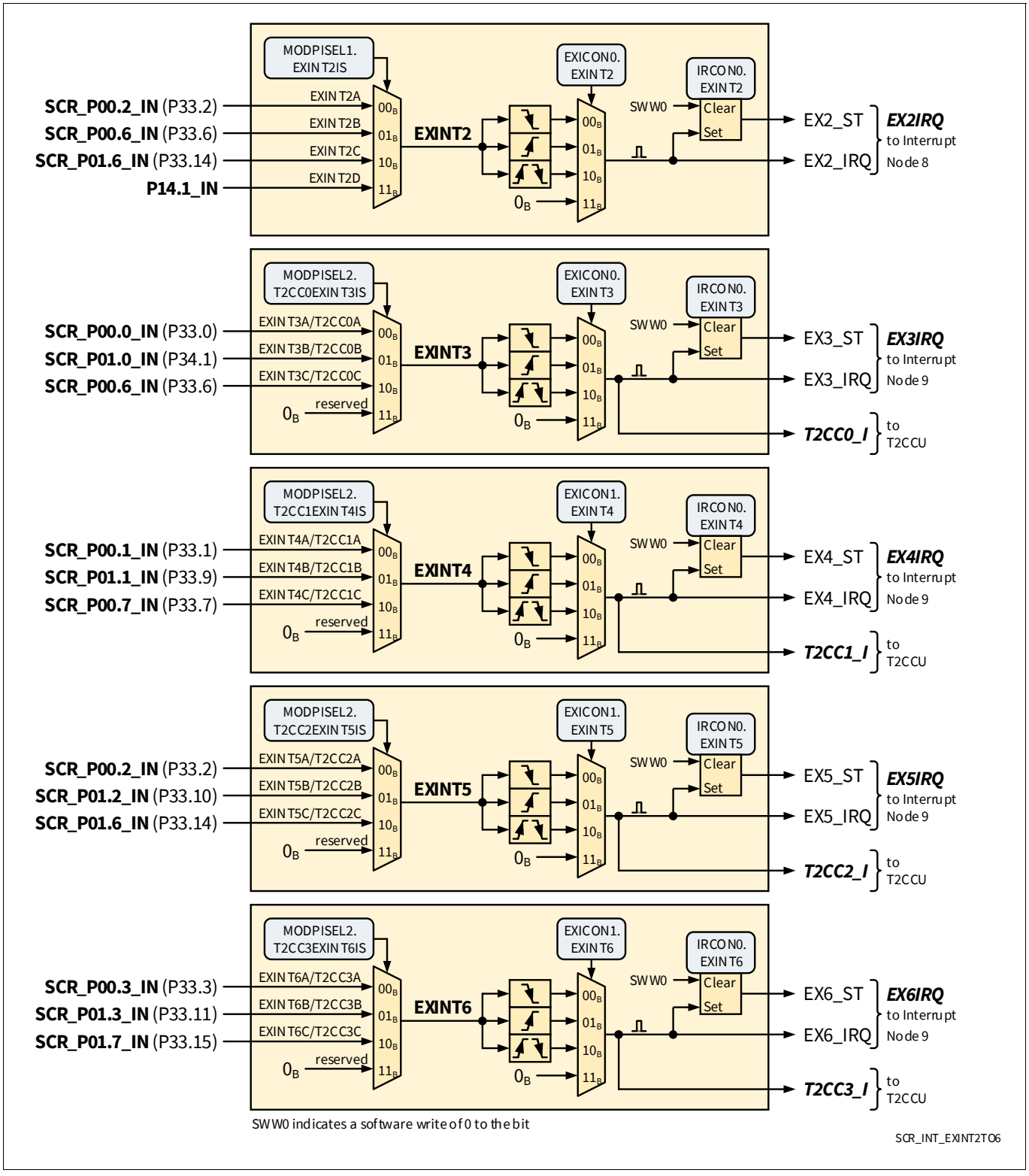


Figure 836 Block Diagram and Connections of External Interrupts EXINT2 through EXINT6

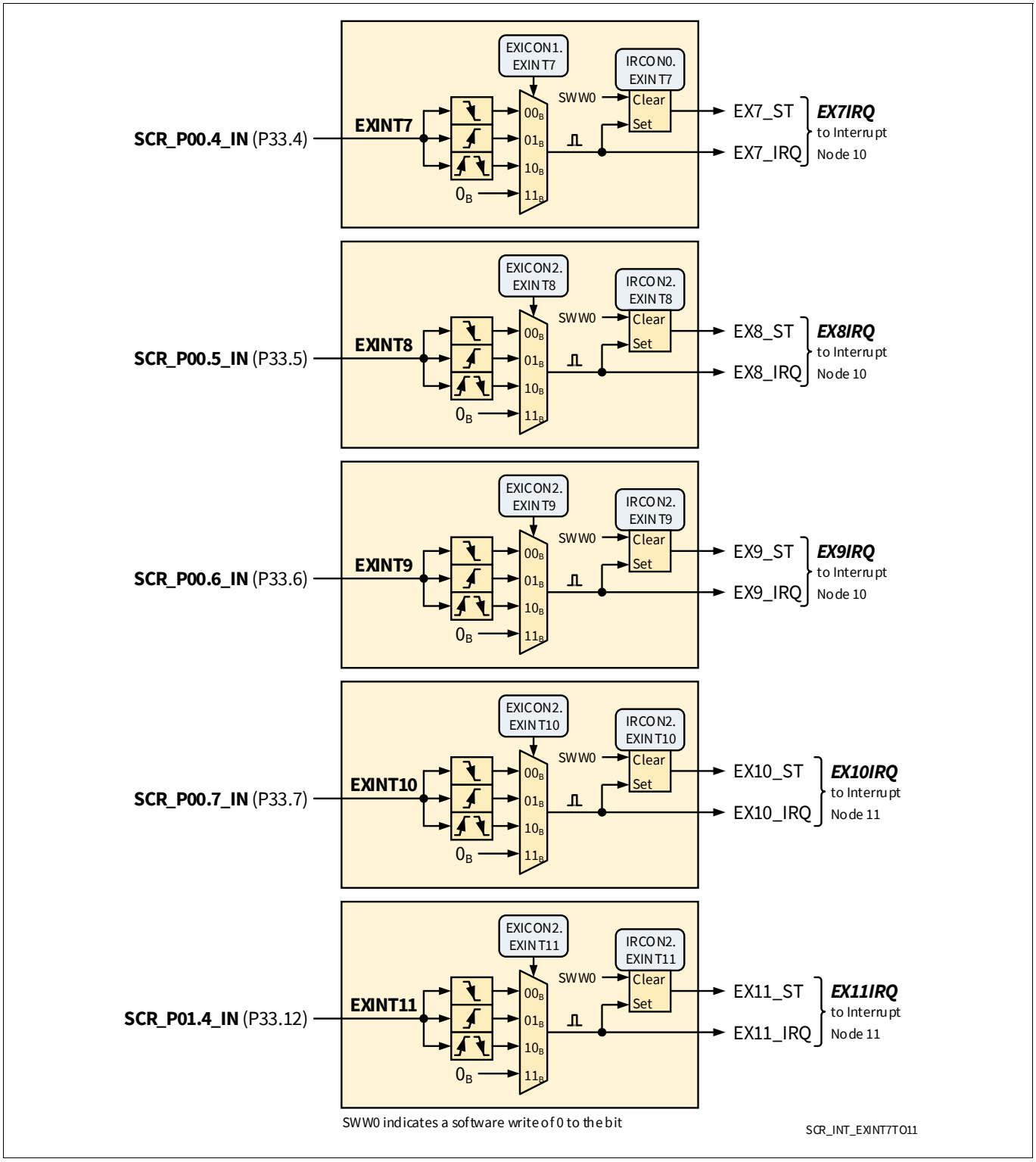


Figure 837 Block Diagram and Connections of External Interrupts EXINT7 through EXINT11

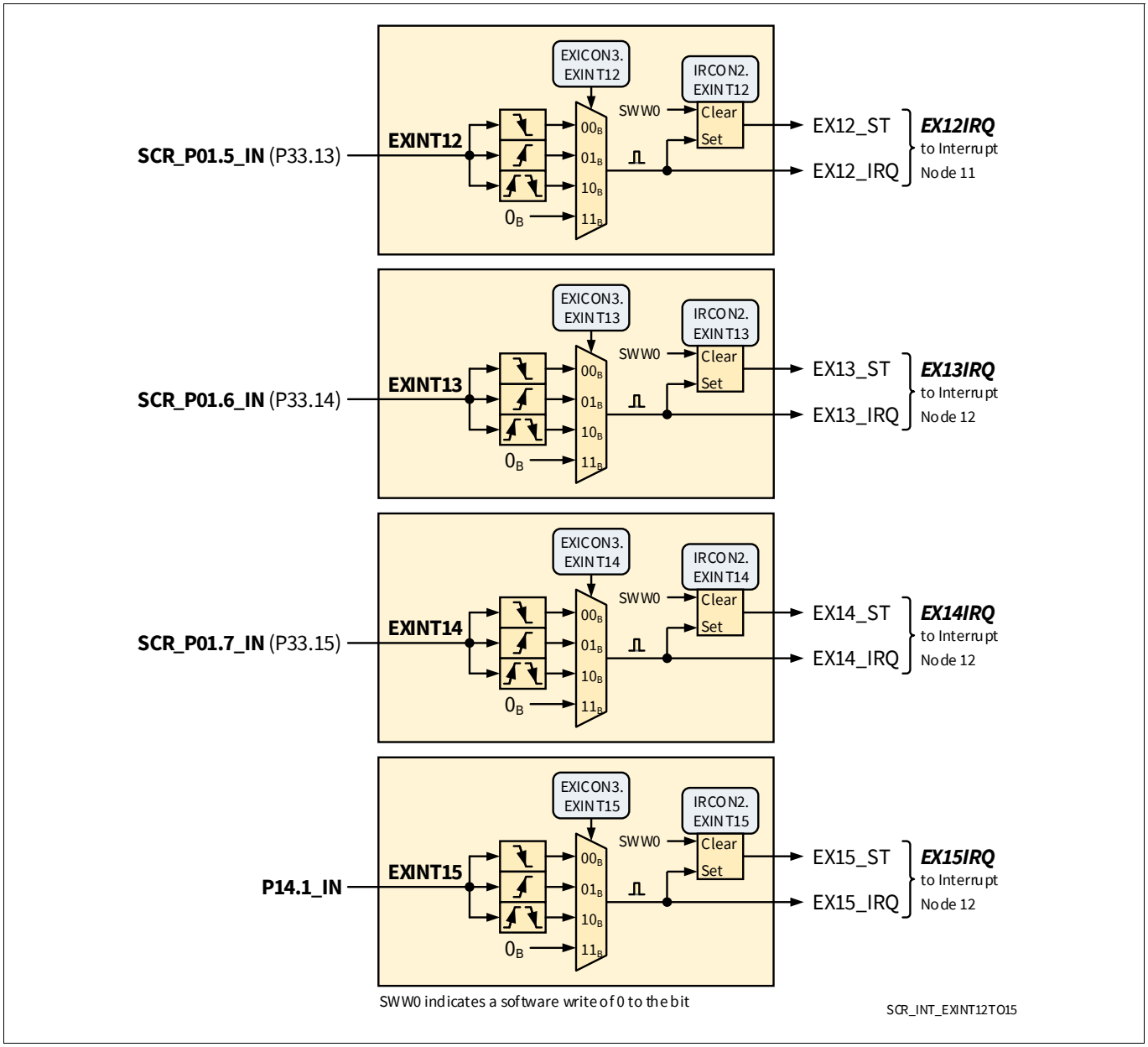


Figure 838 Block Diagram and Connections of External Interrupts EXINT12 through EXINT15

External Interrupt Control Register 0

EXICON0
External Interrupt Control Register 0 ($0F4_H$) Reset Value: Table 604
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
EXINT3		EXINT2		EXINT1		EXINT0	
rw		rw		rw		rw	

Field	Bits	Type	Description
EXINT0	1:0	rw	**External Interrupt 0 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **BYPASS**, Bypass the edge detection in SCU. The input signal directly feeds to the core.
EXINT1	3:2	rw	**External Interrupt 1 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **BYPASS**, Bypass the edge detection in SCU. The input signal directly feeds to the core.
EXINT2	5:4	rw	**External Interrupt 2 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 2 is disabled
EXINT3	7:6	rw	**External Interrupt 3 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 3 is disabled

Table 604 Reset Values of EXICON0

Reset Type	Reset Value	Note
LVD Reset	$F0_H$	
Generated Reset	00_H	

External Interrupt Control Register 1

EXICON1
External Interrupt Control Register 1 ($0F5_H$) Reset Value: Table 605
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
EXINT7		**EXINT6**		**EXINT5**		**EXINT4**	
rw		rw		rw		rw	

Field	Bits	Type	Description
EXINT4	1:0	rw	**External Interrupt 4 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 4 is disabled
EXINT5	3:2	rw	**External Interrupt 5 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 5 is disabled
EXINT6	5:4	rw	**External Interrupt 6 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 6 is disabled
EXINT7	7:6	rw	**External Interrupt 7 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 7 is disabled

Table 605 Reset Values of EXICON1

Reset Type	Reset Value	Note
LVD Reset	FF_H	
Generated Reset	00_H	

External Interrupt Control Register 2

EXICON2
External Interrupt Control Register 2 **($0F6_H$)** **Reset Value: Table 606**
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
EXINT11		**EXINT10**		**EXINT9**		**EXINT8**	
rw		rw		rw		rw	

Field	Bits	Type	Description
EXINT8	1:0	rw	**External Interrupt 8 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 8 is disabled
EXINT9	3:2	rw	**External Interrupt 9 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 9 is disabled
EXINT10	5:4	rw	**External Interrupt 10 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 10 is disabled
EXINT11	7:6	rw	**External Interrupt 11 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 11 is disabled

Table 606 Reset Values of EXICON2

Reset Type	Reset Value	Note
LVD Reset	FF_H	
Generated Reset	00_H	

External Interrupt Control Register 3

EXICON3
External Interrupt Control Register 3 **($0F7_H$)** **Reset Value: Table 607**
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
EXINT15		**EXINT14**		**EXINT13**		**EXINT12**	
rw		rw		rw		rw	

Field	Bits	Type	Description
EXINT12	1:0	rw	**External Interrupt 12 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 12 is disabled
EXINT13	3:2	rw	**External Interrupt 13 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 13 is disabled
EXINT14	5:4	rw	**External Interrupt 14 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 14 is disabled
EXINT15	7:6	rw	**External Interrupt 15 Trigger Select** 00_B **INT_FALL_EDGE**, Interrupt on falling edge 01_B **INT_RISE_EDGE**, Interrupt on rising edge 10_B **INT_BOTH_EDGE**, Interrupt on both, rising and falling edge 11_B **DISABLE**, External interrupt 15 is disabled

Table 607 Reset Values of EXICON3

Reset Type	Reset Value	Note
LVD Reset	FF_H	
Generated Reset	00_H	

Timer 0/1 Control Register

The Register T01_TCON includes also external interrupt control flags. Please see **"Timer 0/1 Control Register" on Page 117**.

49.9.6.3 Interrupt Flag Registers

The interrupt flags for the different interrupt sources are located in several Special Function Registers. In case of software and hardware access to a flag bit at the same time, hardware will have higher priority.

Interrupt Request Register 0

IRCON0
Interrupt Request Register 0 **($0F2_H$)** **Reset Value: Table 608**
RMAP: 0, PAGE: SCU_PAGE=0

7	6	5	4	3	2	1	0
EXINT7	**EXINT6**	**EXINT5**	**EXINT4**	**EXINT3**	**EXINT2**	**0**	
rwh	rwh	rwh	rwh	rwh	rwh	r	

Field	Bits	Type	Description
EXINTm (m=2-7)	m	rwh	**Interrupt Flag for External Interrupt x** This bit is set by hardware and can only be cleared by software. 0_B Interrupt request has not occurred 1_B Interrupt request has occurred
0	1:0	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 608 Reset Values of IRCON0

Reset Type	Reset Value	Note
LVD Reset	0000 00XX_B	
Generated Reset	0000 00$--_B$	

Interrupt Request Register 1

IRCON1
Interrupt Request Register 1 **($0F3_H$)** **Reset Value: Table 609**
RMAP: 0, PAGE: SCU_PAGE=0

7	6	5	4	3	2	1	0
0	**RFR**	**RER**	**ADCIR**	**0**	**RIR**	**TIR**	**EIR**
r	rwh	rwh	rwh	r	rwh	rwh	rwh

Field	Bits	Type	Description
EIR	0	rwh	**Error Interrupt Flag for SSC** This bit is set by hardware, and can only be cleared by software. 0_B Interrupt request has not occurred 1_B Interrupt request has occurred
TIR	1	rwh	**Transmit Interrupt Flag for SSC** This bit is set by hardware, and can only be cleared by software. 0_B Interrupt request has not occurred 1_B Interrupt request has occurred
RIR	2	rwh	**Receive Interrupt Flag for SSC** This bit is set by hardware, and can only be cleared by software. 0_B Interrupt request has not occurred 1_B Interrupt request has occurred
ADCIR	4	rwh	**Interrupt Flag for ADCOMP unit** This bit is set by hardware, and can only be cleared by software. 0_B Interrupt request has not occurred 1_B Interrupt request has occurred
RER	5	rwh	**Receive FIFO Empty Interrupt Flag for SSC** This bit is set by hardware, and can only be cleared by software. 0_B Interrupt request has not occurred 1_B Interrupt request has occurred
RFR	6	rwh	**Receive FIFO Full Interrupt Flag for SSC** This bit is set by hardware, and can only be cleared by software. 0_B Interrupt request has not occurred 1_B Interrupt request has occurred
0	3, 7	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 609 Reset Values of IRCON1

Reset Type	Reset Value	Note
LVD Reset	X000 $X000_B$	
Generated Reset	–000 $–000_B$	

Interrupt Request Register 2

IRCON2
Interrupt Request Register 2 **($0F4_H$)** **Reset Value: Table 610**
RMAP: 0, PAGE: SCU_PAGE=0

7	6	5	4	3	2	1	0
EXINT15	**EXINT14**	**EXINT13**	**EXINT12**	**EXINT11**	**EXINT10**	**EXINT9**	**EXINT8**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
EXINTm (m=8-15)	m-8	rwh	**Interrupt Flag for External Interrupt x** This bit is set by hardware and can only be cleared by software. 0_B Interrupt request has not occurred 1_B Interrupt request has occurred

Table 610 Reset Values of IRCON2

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Timer 0/1 Control Register

T01_TCON
Timer 0/1 Control Register **($0C9_H$)** **Reset Value: Table 611**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**
rwh	rw	rwh	rw	rwh	rw	rwh	rw

Field	Bits	Type	Description
IT0	0	rw	**External Interrupt 0 Level/Edge Trigger Control** 0_B Low level triggered external interrupt 0 is selected 1_B Falling edge triggered external interrupt 0 is selected
IE0	1	rwh	**External Interrupt 0 Flag** Set by hardware when external interrupt 0 event is detected. Cleared by hardware when the processor vectors to interrupt routine. Can also be set/cleared by software.
IT1	2	rw	**External Interrupt 1 Level/Edge Trigger Control** 0_B Low level triggered external interrupt 1 is selected 1_B Falling edge triggered external interrupt 1 is selected
IE1	3	rwh	**External Interrupt 1 Flag** Set by hardware when external interrupt 1 event is detected. Cleared by hardware when the processor vectors to interrupt routine. Can also be set/cleared by software.
TR0	4	rw	**Timer 0 Run Control** 0_B Timer is halted 1_B Timer runs

Field	Bits	Type	Description
TF0	5	rwh	**Timer 0 Overflow Flag** Set by hardware on Timer/Counter 0 overflow. Cleared by hardware when processor vectors to interrupt routine. Can also be set/cleared by software.
TR1	6	rw	**Timer 1 Run Control** *Note: Timer 1 Run Control also affects TH0, if Timer 0 operates in Mode 3.* 0_B Timer is halted 1_B Timer runs
TF1	7	rwh	**Timer 1 Overflow Flag** Set by hardware on Timer/Counter 1 overflow. Cleared by hardware when processor vectors to interrupt routine. Can also be set/cleared by software. *Note: TF1 is set by TH0 instead, if Timer 0 operates in Mode 3.*

Table 611 Reset Values of T01_TCON

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

Serial Channel Control Register

UART_SCON
Serial Channel Control Register ($0BA_H$) Reset Value: Table 612
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
SM01	**SM01**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**
rw	rw	rw	rw	rw	rwh	rwh	rwh

Field	Bits	Type	Description
RI	0	rwh	**Receive Interrupt Flag** This is set by hardware at the end of the 8th bit on mode 0, or at the half point of the stop bit in modes 1, 2, and 3. This bit can also be set by software. Must be cleared by software. *Note: In case of an ongoing reception and a mode change at the very same time, the flag RI will have an undefined behaviour, as it is dependent from the point in time of the mode change.*

Field	Bits	Type	Description
TI	1	rwh	**Transmit Interrupt Flag** This is set by hardware at the end of the 8th bit in mode 0, or at the beginning of the stop bit in modes 1, 2, and 3. This bit can also be set by software. Must be cleared by software. *Note: If a transmission has been started and TI is still 0, a further write to UART_SBUF will destroy the ongoing transmission. If TI is set to 1, the next transmission can be started.*
RB8	2	rwh	**Serial Port Receiver Bit 9** In modes 2 and 3, this is the 9th data bit received. In mode 1, this is the stop bit received. In mode 0, this bit is not used.
TB8	3	rw	**Serial Port Transmitter Bit 9** In modes 2 and 3, this is the 9th data bit sent.
REN	4	rw	**Enable Receiver of Serial Port** *Note: The operating mode of the UART must not be changed while bit REN is set (or while a transmission is in progress). Before changing the mode of the UART, bit REN shall be cleared to 0.* 0_B Serial reception is disabled 1_B Serial reception is enabled
SM2	5	rw	**Enable Serial Port Multiprocessor Communication in Modes 2 and 3** In mode 2 or 3, if SM2 is set to 1, RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 is set to 1, RI will not be activated if a valid stop bit (RB8) was not received. In mode 0, SM2 should be 0. *Note: The operating mode of the UART must not be changed while bit REN is set or while a transmission is in progress. Before changing the mode of the UART, make sure that no transmissions are in progress, and that bit REN is cleared to 0.*
SM01	7:6	rw	**Serial Port Operating Mode Selection** *Note: The operating mode of the UART must not be changed while bit REN is set or while a transmission is in progress. Before changing the mode of the UART, make sure that no transmissions are in progress, and that bit REN is cleared to 0.* 00_B **MODE_0**, 8-bit shift register, fixed baud rate (fPCLK/2) 01_B **MODE_1**, 8-bit UART, variable baud rate 10_B **MODE_2**, 9-bit UART, fixed baud rate (fPCLK/64 or fPCLK/32) 11_B **MODE_3**, 9-bit UART, variable baud rate

Table 612 Reset Values of UART_SCON

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

NMI Status Register

NMISR
NMI Status Register **($0F2_H$)** **Reset Value: Table 613**
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
0	**FNMIWKP**	**0**	**FNMIMAINC**	**FNMIOCDS**	**FNMIEXTNMI**	**FNMIRAMECC**	**FNMIWDT**
r	rwh	r	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
FNMIWDT	0	rwh	**Watchdog Timer NMI Flag** 0_B No watchdog NMI has occurred 1_B WDT pre-warning has occurred
FNMIRAMECC	1	rwh	**RAM Double bit ECC Error NMI Flag** 0_B No RAM ECC NMI has occurred 1_B RAM ECC NMI has occurred
FNMIEXTNMI	2	rwh	**External NMI (via EXTNMI pin) Flag** *Note: It is recommended to clear the NMISR.FNMIEXTNMI flag before enabling external NMI.* 0_B No external NMI (via EXTNMI pin) has occurred 1_B External NMI (via EXTNMI pin) has occurred
FNMIOCDS	3	rwh	**OCDS NMI Flag** 0_B No OCDS NMI has occurred 1_B JTAG-receiving or user interrupt request in Monitor Mode has occurred
FNMIMAINC	4	rwh	**Main Controller NMI Flag** 0_B No Main controller NMI has occurred 1_B Main controller NMI has occurred
FNMIWKP	6	rwh	**Wake-Up NMI Flag** 0_B No wake-up NMI has occurred 1_B Wake-up NMI has occurred
0	5, 7	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 613 Reset Values of NMISR

Reset Type	Reset Value	Note
LVD Reset	X0X0 0000_B	
Main Reset	–0–0 0000_B	

Note: NMISR register can only be cleared by software or reset to the default value after the Main Reset. Its value is retained on any other resets. This allows the cause of the previous NMI to be checked.

49.9.6.4 Interrupt Priority Registers

Each interrupt node can be individually programmed to one of the four priority levels available. Two pairs of Interrupt Priority Registers are available to program the priority level of the each interrupt vector. The first pair of Interrupt Priority Registers are SFRs IP and IPH. The second pair of Interrupt Priority Registers are SFRs IP1 and IPH1.

The corresponding bits in each pair of Interrupt Priority Registers select one of the four priority levels as shown in **Table 614**.

Table 614 Interrupt Priority Level Selection

IPH.x / IPH1.x	IP.x / IP1.x	Priority Level
0	0	Level 0 (lowest)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (highest)

*Note: NMI always has the highest priority (above Level 3); it does not use the priority level selection shown in **Table 614**.*

Interrupt Priority Register

IP
Interrupt Priority Register **($0DC_H$)** **Reset Value: Table 615**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
0		**PT2**	**PS**	**PT1**	**PX1**	**PT0**	**PX0**
r		rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PX0	0	rw	**Priority Level Low Bit for Interrupt Node XINTR0**
PT0	1	rw	**Priority Level Low Bit for Interrupt Node XINTR1**
PX1	2	rw	**Priority Level Low Bit for Interrupt Node XINTR2**
PT1	3	rw	**Priority Level Low Bit for Interrupt Node XINTR3**
PS	4	rw	**Priority Level Low Bit for Interrupt Node XINTR4**
PT2	5	rw	**Priority Level Low Bit for Interrupt Node XINTR5**
0	7:6	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 615 Reset Values of IP

Reset Type	Reset Value	Note
Generated Reset	$XX00\ 0000_B$	
LVD Reset	$--00\ 0000_B$	

Interrupt Priority High Register

IPH
Interrupt Priority High Register **($0D2_H$)** **Reset Value: Table 616**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
0		**PT2H**	**PSH**	**PT1H**	**PX1H**	**PT0H**	**PX0H**
r		rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PX0H	0	rw	**Priority Level High Bit for Interrupt Node XINTR0**
PT0H	1	rw	**Priority Level High Bit for Interrupt Node XINTR1**
PX1H	2	rw	**Priority Level High Bit for Interrupt Node XINTR2**
PT1H	3	rw	**Priority Level High Bit for Interrupt Node XINTR3**
PSH	4	rw	**Priority Level High Bit for Interrupt Node XINTR4**
PT2H	5	rw	**Priority Level High Bit for Interrupt Node XINTR5**
0	7:6	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 616 Reset Values of IPH

Reset Type	Reset Value	Note
Generated Reset	$XX00\ 0000_B$	
LVD Reset	$--00\ 0000_B$	

Interrupt Priority 1 Register

IP1
Interrupt Priority 1 Register **($0DB_H$)** **Reset Value: Table 617**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
PCCIP3	**PCCIP2**	**PCCIP1**	**PCCIP0**	**PXM**	**PX2**	**PSSC**	**PWCAN**
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PWCAN	0	rw	**Priority Level Low Bit for Interrupt Node XINTR6**
PSSC	1	rw	**Priority Level Low Bit for Interrupt Node XINTR7**
PX2	2	rw	**Priority Level Low Bit for Interrupt Node XINTR8**
PXM	3	rw	**Priority Level Low Bit for Interrupt Node XINTR9**
PCCIP0	4	rw	**Priority Level Low Bit for Interrupt Node XINTR10**
PCCIP1	5	rw	**Priority Level Low Bit for Interrupt Node XINTR11**
PCCIP2	6	rw	**Priority Level Low Bit for Interrupt Node XINTR12**

Field	Bits	Type	Description
PCCIP3	7	rw	**Priority Level Low Bit for Interrupt Node XINTR13**

Table 617 Reset Values of IP1

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

Interrupt Priority 1 High Register

IPH1
Interrupt Priority 1 High Register ($0D3_H$) Reset Value: Table 618
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
PCCIP3H	**PCCIP2H**	**PCCIP1H**	**PCCIP0H**	**PXMH**	**PX2H**	**PSSCH**	**PWCANH**
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PWCANH	0	rw	**Priority Level High Bit for Interrupt Node XINTR6**
PSSCH	1	rw	**Priority Level High Bit for Interrupt Node XINTR7**
PX2H	2	rw	**Priority Level High Bit for Interrupt Node XINTR8**
PXMH	3	rw	**Priority Level High Bit for Interrupt Node XINTR9**
PCCIP0H	4	rw	**Priority Level High Bit for Interrupt Node XINTR10**
PCCIP1H	5	rw	**Priority Level High Bit for Interrupt Node XINTR11**
PCCIP2H	6	rw	**Priority Level High Bit for Interrupt Node XINTR12**
PCCIP3H	7	rw	**Priority Level High Bit for Interrupt Node XINTR13**

Table 618 Reset Values of IPH1

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

49.9.7 Interrupt Events Related Registers

There are some interrupt events related registers, which can be used to enable specific interrupts, or to exchange data to and from the main controller. These are described here.

49.9.7.1 Interrupt Event Enable Control

The five interrupt events of SSC module are of interrupt structure 2. As there is no enable/disable bit(s) for these interrupt events within the module, bits are defined in the SCU register SCU_MODIEN to provide for this purpose.

Peripheral Interrupt Enable Register

SCU_MODIEN
Peripheral Interrupt Enable Register **($0F8_H$)** **Reset Value: Table 619**
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
	0		**FFEN**	**FEEN**	**RIREN**	**TIREN**	**EIREN**
	r		rw	rw	rw	rw	rw

Field	Bits	Type	Description
EIREN	0	rw	**SSC Error Interrupt Enable** 0_B Error interrupt is disabled 1_B Error interrupt is enabled
TIREN	1	rw	**SSC Transmit Interrupt Enable** 0_B Transmit interrupt is disabled 1_B Transmit interrupt is enabled
RIREN	2	rw	**SSC Receive Interrupt Enable** 0_B Receive interrupt is disabled 1_B Receive interrupt is enabled
FEEN	3	rw	**SSC Receive FIFO Empty Interrupt Enable** 0_B Receive FIFO empty interrupt is disabled 1_B Receive FIFO empty interrupt is enabled
FFEN	4	rw	**SSC Receive FIFO Full Interrupt Enable** 0_B Receive FIFO full interrupt is disabled 1_B Receive FIFO full interrupt is enabled
0	7:5	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 619 Reset Values of SCU_MODIEN

Reset Type	Reset Value	Note
LVD Reset	XXX0 0111_B	
Generated Reset	---0 0000_B	

49.9.7.2 Interrupt Data Exchange Registers

Two registers are used to exchange interrupt data between main controller and SCR as described in Section 49.9.5, “Interrupt from/to TriCore CPUx” on Page 101.

SCR Interrupt Data Exchange Register

SCRINTEXCHG
SCR Interrupt Data Exchange Register (0F5$_H$) Reset Value: Table 620
RMAP: 0, PAGE: SCU_PAGE=0

7	6	5	4	3	2	1	0
SCRINT							
rwh							

Field	Bits	Type	Description
SCRINT	7:0	rwh	**Data Exchange from SCR to Main Controller** Fast data exchange between SCR and main controller. The data can be read via PMSWCR2.SCRINT in the main SCU.

Table 620 Reset Values of SCRINTEXCHG

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	$--_H$	

TriCore Interrupt Data Exchange Register

TCINTEXCHG
TriCore Interrupt Data Exchange Register (0F6$_H$) Reset Value: Table 621
RMAP: 0, PAGE: SCU_PAGE=0

7	6	5	4	3	2	1	0
TCINT							
rh							

Field	Bits	Type	Description
TCINT	7:0	rh	**Data Exchange from Main Controller to SCR** Fast data exchange between main controller and SCR. The data can only be read in SCR.

Table 621 Reset Values of TCINTEXCHG

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.9.8 Interrupt Flag Overview

The interrupt events have interrupt flags that are located in different SFRs. **Table 622** shows the corresponding SFR to which each interrupt flag belongs. Detailed information on the interrupt flags is provided in the respective peripheral chapters.

Table 622 Location of the Interrupt Flags

Interrupt Event	Interrupt Flag	SFR
Timer 0 Overflow	TF0	T01_TCON
Timer 1 Overflow	TF1	T01_TCON
Timer2 Overflow	TF2	T2_T2CON
Timer2 External Event	EXF2	T2_T2CON
UART Receive	RI	UART_SCON
UART Transmit	TI	UART_SCON
LIN End of Synch Byte	EOFSYN	LINST
LIN Synch Byte Error	ERRSYN	LINST
External Interrupt 0	IE0	T01_TCON
External Interrupt 1	IE1	T01_TCON
External Interrupt 2	EXINT2	IRCON0
External Interrupt 3	EXINT3	IRCON0
External Interrupt 4	EXINT4	IRCON0
External Interrupt 5	EXINT5	IRCON0
External Interrupt 6	EXINT6	IRCON0
External Interrupt 7	EXINT7	IRCON0
External Interrupt 8	EXINT8	IRCON2
External Interrupt 9	EXINT9	IRCON2
External Interrupt 10	EXINT10	IRCON2
External Interrupt 11	EXINT11	IRCON2
External Interrupt 12	EXINT12	IRCON2
External Interrupt 13	EXINT13	IRCON2
External Interrupt 14	EXINT14	IRCON2
External Interrupt 15	EXINT15	IRCON2
T2CCU CCT Overflow	CCTOVF	T2CCU_CCTCON
WCAN WUF detection interrupt	WUF	WCAN_INTESTAT1
WCAN error overflow interrupt	SYSERR	WCAN_INTESTAT0
WCAN WUP detection interrupt	WUP	WCAN_INTESTAT0
WCAN time-out interrupt	CANTO	WCAN_INTESTAT0
RTC compare/wakeup interrupt	CFRTC	RTC_CON
ADCOMP Service Request	ADCIR	IRCON1
SSC Error	EIR	IRCON1
SSC Transmit	TIR	IRCON1

Table 622 Location of the Interrupt Flags (cont'd)

Interrupt Event	Interrupt Flag	SFR
SSC Receive	RIR	IRCON1
SSC Receive FIFO Full	RFR	IRCON1
SSC Receive FIFO Empty	RER	IRCON1
Watchdog Timer NMI	FNMIWDT	NMISR
RAM Double bit ECC Error NMI	FNMIRAMECC	NMISR
Wake-Up Event NMI	FNMIWKP	NMISR
External NMI (via EXTNMI pin) NMI	FNMIEXTNMI	NMISR
OCDS NMI	FNMIOCDS	NMISR
Main Controller NMI	FNMIMAINC	NMISR

49.9.9 Revision History

Table 623 Revision History

Reference	Change to Previous Version	Change Request
V3.6		
	First Official Release of completely reworked SCR chapter	
	No change	

49.10 General Purpose I/O Ports and Peripheral I/O Lines (Ports)

The SCR has a maximum of 16 general purpose I/O ports with 8-bit peripheral I/O lines that are shared with the 32-bit peripherals I/O lines. It is organized into two parallel ports. Internally within the SCR, these ports are always referred to as SCR_P00 and SCR_P01. Here, within the SCR, the terms SCR_P00 and SCR_P01 do not refer to the main-SoC ports P00 and P01. Instead, they are actually mapped to pins in ports P33 and P34.1 in this platform. This particular pin assignment of these functions is shown in **Table 624**.

Upon reset, the 32-bit Ports module has the full control of all the pads. During run mode, the control of these 16 pins can be switched to SCR if necessary via the 32-bit Ports register Pn_PCSR.

A separate set of pad control registers as shown in **Table 625** are available in SCR to control these ports.

In **Table 624**, the transfer from a TC26x device to a TC3xx device is shown.

Table 624 Pin assignment for Ports in SCR

SCR Ports	Main SoC Ports / Pins
SCR_P00.0 - SCR_P00.7	P33.0 - P33.7
SCR_P01.0	P34.1
SCR_P01.1 - SCR_P01.7	P33.9 - P33.15

Note: Not all pins described in this chapter might be available for a specific device.

49.10.1 Basic Port Operation

Figure 839 is a general block diagram of an SCR GPIO port slice.

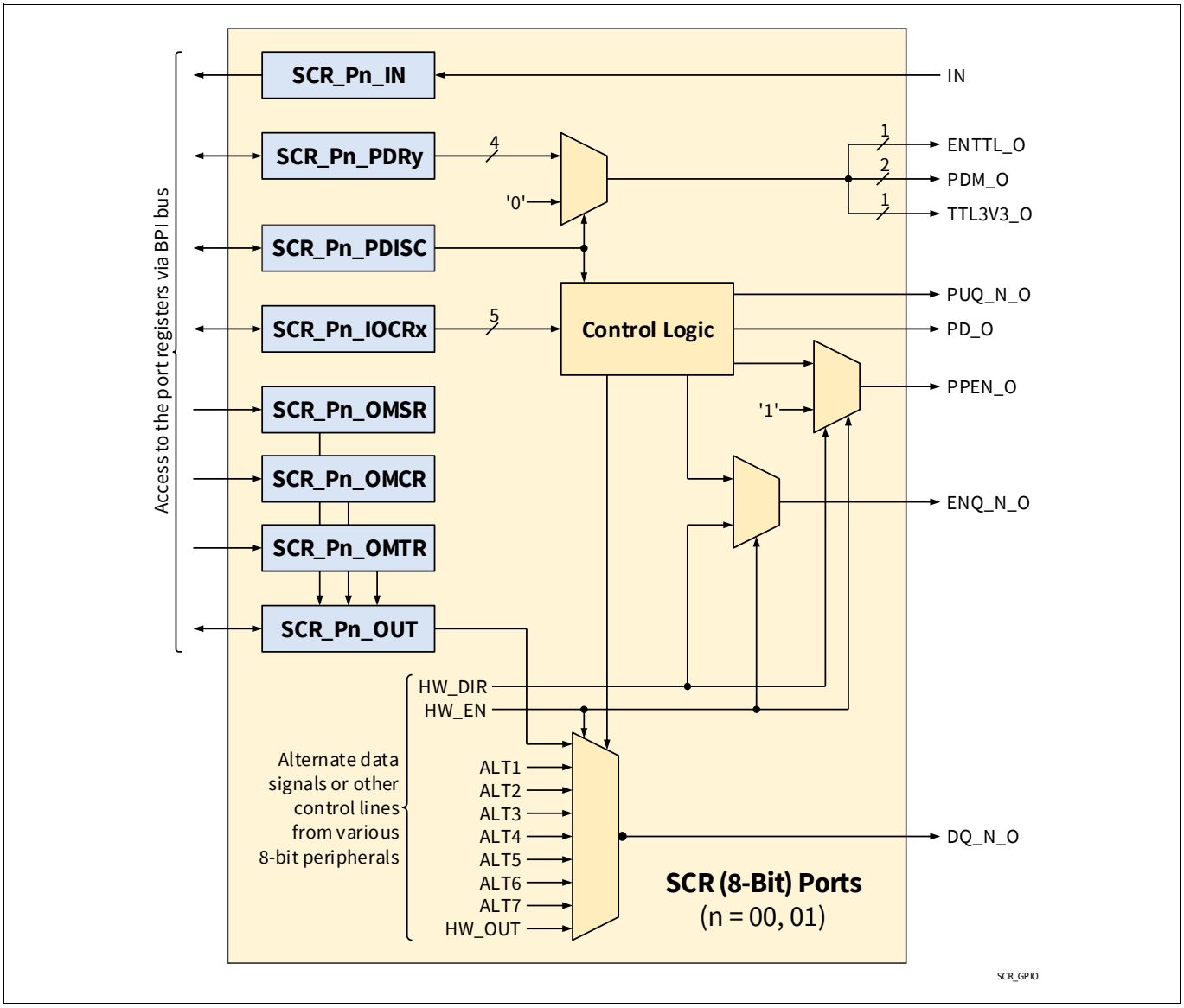


Figure 839 General Structure of a Port Pin

Each port line has a number of control and data bits, enabling very flexible usage of the line. Each port pin can be configured for input or output operation. In input mode (default after reset), the output driver is switched off (high-impedance). The actual voltage level present at the port pin is translated into a logical 0 or 1 via a Schmitt-Trigger device and can be read via the read-only register SCR_Pn_IN (n=00, 01). Input signals are connected directly to the various inputs of the peripheral units (AltDataIn). The function of the input line from the pin to the input register SCR_Pn_IN and to AltDataIn is independent of whether the port pin operates as input or output.

This means that when the port is in output mode, the level of the pin can be read by software via SCR_Pn_IN or a peripheral can use the pin level as an input.

In output mode, the output driver is activated and drives the value supplied through the multiplexer to the port pin. Switching between input and output mode is accomplished through the SCR_Pn_IOCR register, which enables or disables the output driver. If a peripheral unit uses a GPIO port line as a bi-directional I/O line, register SCR_Pn_IOCR has to be written for input or output selection. The SCR_Pn_IOCR register further controls the driver type of the output driver, and determines whether an internal weak pull-up, pull-down, or without input

pull device is alternatively connected to the pin when used as an input. This offers additional advantages in an application.

The output multiplexer in front of the output driver selects the signal source for the GPIO line when used as output. If the pin is used as general-purpose output, the multiplexer is switched by software (SCR_Pn_IOCR register) to the Output Data Register SCR_Pn_OUT. Software can set, clear, or toggle the bit in SCR_Pn_OUT through separate SCR_Pn_OMSR, SCR_Pn_OMCR, or SCR_Pn_OMTR registers. The manipulation of the control bits in these registers can directly influence the state of the port pin. If the on-chip peripheral units use the pin for output signals, the alternate output lines ALT1 to ALT7 can be switched via the multiplexer to the output driver. The data written into the output register SCR_Pn_OUT by software can be used as input data to an on-chip peripheral. This enables, for example, peripheral tests via software without external circuitry.

When selected as general-purpose output line, the logic state of each port pin can be changed individually by programming the pin-related bits in the Output Modification Set Register SCR_Pn_OMSR, Output Modification Clear Register SCR_Pn_OMCR, or the Output Modification Toggle Register SCR_Pn_OMTR.

When selected as general-purpose output line, the actual logic level at the pin can be examined through reading Pn_IN and compared against the applied output level (either applied through software via the output register SCR_Pn_OUT, or via an alternate output function of a peripheral unit). This can be used to detect some electrical failures at the pin caused through external circuitry. In addition, software-supported arbitration schemes can be implemented in this way using the open-drain configuration and an external wired-And circuitry. Collisions on the external communication lines can be detected when a high level (1) is output, but a low level (0) is seen when reading the pin value via the input register SCR_Pn_IN.

49.10.2 Port Register Overview

The individual control and data bits of each parallel port are implemented in a number of 8-bit registers. Bits with the same meaning and function are assembled together in the same register. The registers configure and use the port as general purpose I/O or alternate function input/output. For some ports, not all registers are implemented. The availability of the registers in the specific ports is described separately. The definition of these registers are similar to the 32-bit format.

The registers are shown in **Figure 840**.

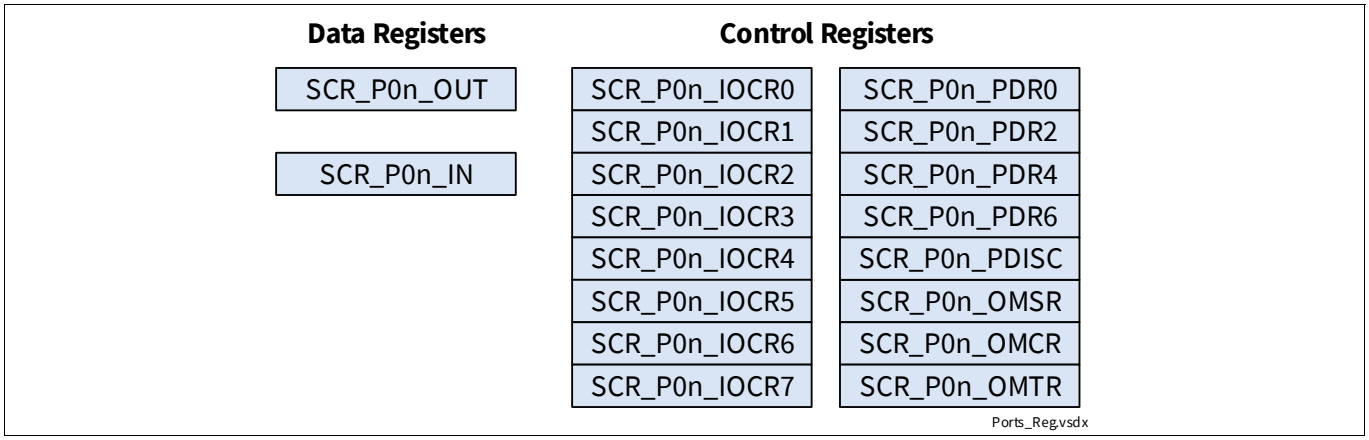


Figure 840 SCR Port Registers

The Port SFRs are located in the standard memory area (RMAP = 0) and the addresses are listed in **Table 625**.

Table 625 Register Overview - PORTS (sorted by Name)

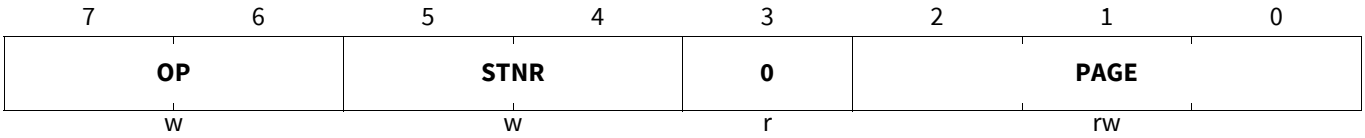
Short Name	Description	RMAP	PAGE	Offset Address	Page Number
IO_PAGE	Page Register for Port SFRs	0	X	$08F_H$	134
P0n_IN	Port P0n Input Register	0	IO_PAGE=0	091_H+n*8	147
P0n_IOCRk	Port P0n Input/Output Control Register k	0	IO_PAGE=1	090_H+n*8+k	136
P0n_OMCR	Port P0n Output Modification Clear Register	0	IO_PAGE=0	093_H+n*8	145
P0n_OMSR	Port P0n Output Modification Set Register	0	IO_PAGE=0	092_H+n*8	144
P0n_OMTR	Port P0n Output Modification Toggle Register	0	IO_PAGE=0	094_H+n*8	146
P0n_OUT	Port P0n Output Register	0	IO_PAGE=0	090_H+n*8	143
P0n_PDISC	Port P0n Pin Function Decision Control Register	0	IO_PAGE=2	095_H+n*8	142
P0n_PDR0	Port P0n Pad Driver Mode 0 Register	0	IO_PAGE=2	090_H+n*8	139
P0n_PDR2	Port P0n Pad Driver Mode 2 Register	0	IO_PAGE=2	091_H+n*8	139
P0n_PDR4	Port P0n Pad Driver Mode 4 Register	0	IO_PAGE=2	092_H+n*8	140
P0n_PDR6	Port P0n Pad Driver Mode 6 Register	0	IO_PAGE=2	093_H+n*8	140

49.10.2.1 Port Paging Register

Page Register for Port SFRs

Page register for Port registers of SCU.

IO_PAGE
Page Register for Port SFRs **($08F_H$)** **Reset Value: Table 626**
RMAP: 0, PAGE: X



Field	Bits	Type	Description
PAGE	2:0	rw	**Page Bits** When written, the value indicates the new page address. When read, the value indicates the currently active page = addr[y:x+1]. 000_B **PAGE0**, ST0 is selected 001_B **PAGE1**, ST1 is selected … 111_B **PAGE7**, ST1 is selected
STNR	5:4	w	**Storage Number** This number indicates which storage bitfield is the target of the operation defined by bit OP. If OP = 10_B, the contents of PAGE are saved in STx before being overwritten with the new value. If OP = 11_B, the contents of PAGE are overwritten by the contents of STx. The value written to bitfield PAGE is ignored. 00_B **MOD_ST0**, ST0 is selected 01_B **MOD_ST1**, ST1 is selected 10_B **MOD_ST2**, ST2 is selected 11_B **MOD_ST3**, ST3 is selected
OP	7:6	w	**Operation** 00_B **PAGE_MANUAL0**, Manual page mode. The value of STNR is ignored and PAGE is directly written. 01_B **PAGE_MANUAL1**, Manual page mode. The value of STNR is ignored and PAGE is directly written 10_B **PAGE_SAVE**, New page programming with automatic page saving. The value written to the bit positions of PAGE is stored. In parallel, the former contents of PAGE are saved in the storage bit field STx indicated by STNR. 11_B **PAGE_RESTORE**, Automatic restore page action. The value written to the bit positions PAGE is ignored and instead, PAGE is overwritten by the contents of the storage bit field STx indicated by STNR.
0	3	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 626 Reset Values of IO_PAGE

Reset Type	Reset Value	Note
LVD Reset	0000 X000$_B$	
Generated Reset	0000 –000$_B$	

49.10.2.2 Port Input/Output Control Registers

The port input/output control registers select the digital output and input driver functionality and characteristics of a GPIO port pin. Port direction (input or output), pull-up, pull-down, or no pull devices for inputs, and push-pull or open-drain functionality for outputs can be selected by the corresponding bit field PC.

Each 8-bit wide port input/output control register controls one GPIO port line:

- Register SCR_P00_IOCR0 controls the SCR_P00.0 (P33.0) port line, register SCR_P01_IOCR0 controls the SCR_P01.0 port line (P34.1).
- Registers SCR_P00/01_IOCR1 control the SCR_P00.1 (P33.1) and SCR_P01.1 (P33.9) port lines.
- Accordingly, the other registers control the port lines SCR_P00.2 through SCR_P00.7 (P33L), and SCR_P01.2 through SCR_P01.7 (P33H), respectively.

The diagrams below show the register layouts of the port input/output control registers with the PCx bit fields. One PCx bit field controls exactly one port line Pn.x.

Note: The reset values of 10_H and 00_H for SCR_Pn_IOCRx registers represents input pull-up and no input pull device (tri-state mode) being activated, respectively. The switching of the intended mode of the device is controlled by HWCFG6. When a cold reset is activated and HWCFG6=1, the pins are set to input pull-up mode as long as PORST is activated. If HWCFG6=0, the pins have the default state of tri-state mode. The pad state can also be configured by software through PMSWCR5.TRISTREQ bit. In the event of a warm reset or wake-up from standby mode, PMSWCR5.TRISTREQ is not affected by reset, hence SCR_Pn_IOCRx registers have the reset values configured as per the last state of the TRISTREQ bit.

Port P0n Input/Output Control Register k

P0n_IOCRk (k=0-7;n=0-1)
Port P0n Input/Output Control Register k (090$_H$+n*8+k) **Reset Value: Table 627**
RMAP: 0, PAGE: IO_PAGE=1

7	6	5	4	3	2	1	0
PC					0		
rw					r		

Field	Bits	Type	Description
PC	7:3	rw	**Port Control for Port n Pin k** This bit field determines the Port n line k functionality according to the coding table (see Table "PCx_Coding"). 10_H **PP_GPO**, … $1F_H$ **OD_ALT_FUNC7**,
0	2:0	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 627 Reset Values of P0n_IOCRk (k=0-7;n=0-1)

Reset Type	Reset Value	Note
LVD Reset	0000 0XXX$_B$	
Generated Reset	0000 0–––$_B$	

Port Control Coding

Table 628 describes the coding of the PCx bit fields that determine the port line functionality.

Table 628 PCx Coding

PCx[4:0]	I/O	Output Characteristics	Selected Pull-up / Pull-down / Selected Output Function
$0XX00_B$	Input		No input pull device connected, tri-state mode
$0XX01_B$			Input pull-down device connected
$0XX10_B$			Input pull-up device connected
$0XX11_B$			No input pull device connected, tri-state mode
10000_B	Output	Push-pull	General-purpose output
10001_B			Alternate output function 1
10010_B			Alternate output function 2
10011_B			Alternate output function 3
10100_B			Alternate output function 4
10101_B			Alternate output function 5
10110_B			Alternate output function 6
10111_B			Alternate output function 7
11000_B		Open-drain	General-purpose output
11001_B			Alternate output function 1
11010_B			Alternate output function 2
11011_B			Alternate output function 3
11100_B			Alternate output function 4
11101_B			Alternate output function 5
11110_B			Alternate output function 6
11111_B			Alternate output function 7

49.10.2.3 Pad Driver Mode Register

Overview

The pad structure of the SCR GPIO lines offers the possibility to select the output driver strength and the slew rate. These two parameters are controlled by the PDx bit fields in the pad driver mode registers SCR_Pn_PDRy (y=0, 2, 4, 6), independently from input/output and pull-up/pull-down control functionality as programmed in the Pn_IOCR register. SCR_Pn_PDRy registers are assigned to each port.

The assignment of each port pin to one of these pad classes is shown in the port configuration figures. Further details about pad driver classes that are available in the AURIX™ TC3xx Platform are summarized in the Target Data Sheet/Data Sheet.

Depending on the assigned pad class, the 3-bit wide pad driver mode selection bit fields PDx in the pad driver mode registers SCR_Pn_PDRy make it possible to select the port line functionality as shown in the tables below.

Table 629 Pad Driver Mode Selection- RFast Pads

PDx.1	PDx.0	Speed Grade	Driver Setting
0	0	1	Strong Driver, Sharp Edge (“ss”)
0	1	2	Strong Driver, Medium Edge (“sm”)
1	0	3	Medium Driver (“m”)
1	1	4	RGMII Driver

Table 630 Pad Driver Mode Selection- Fast Pads

PDx.1	PDx.0	Speed Grade	Driver Setting
0	0	1	Strong Driver, Sharp Edge (“ss”)
0	1	2	Strong Driver, Medium Edge (“sm”)
1	X	3	Medium Driver (“m”)

Table 631 Pad Driver Mode Selection- Slow Pads

PDx.1	PDx.0	Speed Grade	Driver Setting
X	0	1	Medium Driver, Sharp Edge (“sm”)
X	1	2	Medium Driver (“m”)

Note: AURIX™ TC3xx Platform Data Sheet describes the DC characteristics of all pad classes.

TTL/Automotive Input Selection

The input function can operate with different VIH and VIL levels depending on the pad supply voltage and the selection done by the PLx bits of the Pn_PDRx registers, as shown in **Table 632**.

PLx.1 additionally also changes the pull-up and pull-down resistors.

Table 632 Pad Level Selection

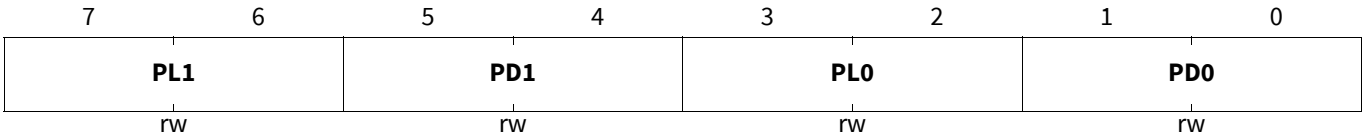
PLx.1	PLx.0	Input Levels
0	X	Automotive level “AL”
1	0	TTL level for 5 V supply
1	1	TTL level for 3.3 V supply

Pad Driver Mode Registers

This is the general description of the PDR registers. Each port contains its own specific PDR registers, described additionally at each port, that can contain between one and eight PDx fields for PDRy (y=0, 2, 4, 6) registers, respectively. Each PDx field controls one pin. For coding of PDx, see **Table 629**, **Table 630** and **Table 631**. Similarly, each PLx bit controls one pin. For coding of PLx, see **Table 632**.

Port P0n Pad Driver Mode 0 Register

P0n_PDR0 (n=0-1)
Port P0n Pad Driver Mode 0 Register **(090_H+n*8)** **Reset Value: Table 633**
RMAP: 0, PAGE: IO_PAGE=2



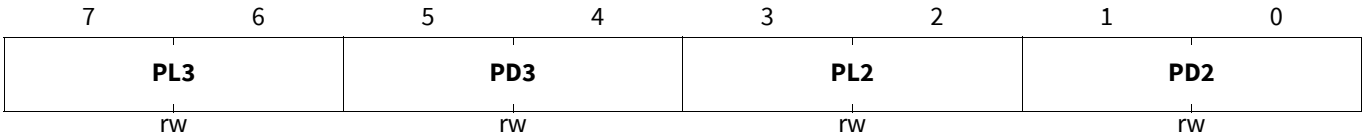
Field	Bits	Type	Description
PDm (m=0-1)	4*m+1:4*m	rw	**Pad Driver Mode for Port n Pin m**
PLm (m=0-1)	4*m+3:4*m+2	rw	**Pad Level Selection for Port n Pin m**

Table 633 Reset Values of P0n_PDR0 (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Port P0n Pad Driver Mode 2 Register

P0n_PDR2 (n=0-1)
Port P0n Pad Driver Mode 2 Register **(091_H+n*8)** **Reset Value: Table 634**
RMAP: 0, PAGE: IO_PAGE=2



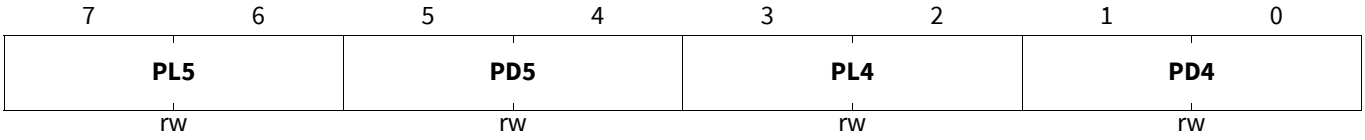
Field	Bits	Type	Description
PDm (m=2-3)	4*m-7:4*m-8	rw	**Pad Driver Mode for Port n Pin m**
PLm (m=2-3)	4*m-5:4*m-6	rw	**Pad Level Selection for Port n Pin m**

Table 634 Reset Values of P0n_PDR2 (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Port P0n Pad Driver Mode 4 Register

P0n_PDR4 (n=0-1)
Port P0n Pad Driver Mode 4 Register (092_H+n*8) Reset Value: Table 635
RMAP: 0, PAGE: IO_PAGE=2



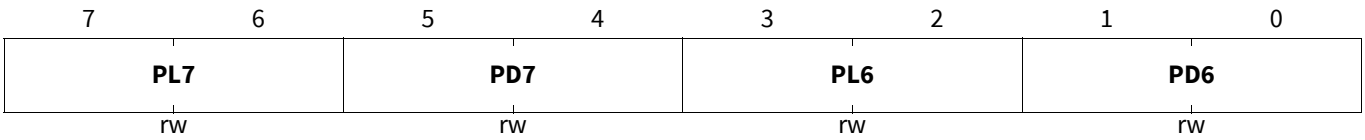
Field	Bits	Type	Description
PDm (m=4-5)	4*m-15:4*m-16	rw	**Pad Driver Mode for Port n Pin m**
PLm (m=4-5)	4*m-13:4*m-14	rw	**Pad Level Selection for Port n Pin m**

Table 635 Reset Values of P0n_PDR4 (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Port P0n Pad Driver Mode 6 Register

P0n_PDR6 (n=0-1)
Port P0n Pad Driver Mode 6 Register (093_H+n*8) Reset Value: Table 636
RMAP: 0, PAGE: IO_PAGE=2



Field	Bits	Type	Description
PDm (m=6-7)	4*m-23:4*m-24	rw	**Pad Driver Mode for Port n Pin m**
PLm (m=6-7)	4*m-21:4*m-22	rw	**Pad Level Selection for Port n Pin m**

Table 636 Reset Values of P0n_PDR6 (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.10.2.4 Pin Function Decision Control Register

The pad structure of the SCR GPIO lines offers the possibility to disable/enable port pad or select digital input. For the selection of digital input, the parameters defined for Class S pads must be met. This feature can be controlled by individual bits in the SCR_Pn_PDISC register, independently from input/output and pull-up/pull-down control functionality as programmed in the SCR_Pn_IOCR register. One SCR_Pn_PDISC register is assigned to each port.

Note: After reset, all SCR_P0n_PDISC registers have the reset value of FF_H.

Port P0n Pin Function Decision Control Register

P0n_PDISC (n=0-1)
Port P0n Pin Function Decision Control Register(095_H+n*8) **Reset Value: Table 637**
RMAP: 0, PAGE: IO_PAGE=2

7	6	5	4	3	2	1	0
PDIS7	**PDIS6**	**PDIS5**	**PDIS4**	**PDIS3**	**PDIS2**	**PDIS1**	**PDIS0**
rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
PDISm (m=0-7)	m	rw	**Pin Function Decision Control for Pin m** This bit selects the function of the port pad. 0_B Pad Pn.m is enabled 1_B Pad Pn.m is disabled

Table 637 Reset Values of P0n_PDISC (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	FF_H	
Generated Reset	FF_H	

49.10.2.5 Port Output Register

The port output register determines the value of a GPIO pin when it is selected by SCR_P0n_IOCRx as output. Writing a 0 to a SCR_P0n_OUT.Px (x = 0-7) bit position delivers a low level at the corresponding output pin. A high level is output when the corresponding bit is written with a 1. Note that the bits of SCR_P0n_OUT.Px can be individually set, cleared, or toggled by writing appropriate values into the port output modification set register SCR_P0n_OMSR, or port output modification clear register SCR_P0n_OMCR, or the port output toggle register SCR_P0n_OMTR, respectively.

Note: SCR_P01_OUT.P0 is connected to P34.1 instead of P33.8

Port P0n Output Register

P0n_OUT (n=0-1)
Port P0n Output Register **(090_H+n*8)** **Reset Value: Table 638**
RMAP: 0, PAGE: IO_PAGE=0

7	6	5	4	3	2	1	0
P7	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
Pm (m=0-7)	m	rwh	**Port n Output Bit m** This bit determines the level at the output pin Pn.m if the output is selected as GPIO output. Pn.m can also be set or cleared by control bits of the SCR_P0n_OMSR, SCR_P0n_OMCR or SCR_P0n_OMTR registers. 0_B The output level of Pn.m is 0 1_B The output level of Pn.m is 1

Table 638 Reset Values of P0n_OUT (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.10.2.6 Port Output Modification Set Register

The port output modification set register contains control bits that make it possible to individually set the logic state of a single port line by manipulating the output register.

Port P0n Output Modification Set Register

P0n_OMSR (n=0-1)
Port P0n Output Modification Set Register (092$_H$+n*8) **Reset Value: Table 639**
RMAP: 0, PAGE: IO_PAGE=0

7	6	5	4	3	2	1	0
PS7	**PS6**	**PS5**	**PS4**	**PS3**	**PS2**	**PS1**	**PS0**
w	w	w	w	w	w	w	w

Field	Bits	Type	Description
PSm (m=0-7)	m	w	**Port n Set Bit m** Setting this bit will set the corresponding bit in the port output register SCR_Pn_OUT. Read as 0. 0_B No operation 1_B Sets Pn_OUT.Pm

Table 639 Reset Values of P0n_OMSR (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Note: Register SCR_Pn_OMSR is virtual and does not contain any flip-flops. A read action delivers the value of 0. A 1-bit write behaves as an 8-bit write padded with zeros.

49.10.2.7 Port Output Modification Clear Register

The port output modification clear register contains control bits that make it possible to individually clear the logic state of a single port line by manipulating the output register.

Port P0n Output Modification Clear Register

P0n_OMCR (n=0-1)
Port P0n Output Modification Clear Register (093$_H$+n*8) **Reset Value: Table 640**
RMAP: 0, PAGE: IO_PAGE=0

7	6	5	4	3	2	1	0
PCL7	**PCL6**	**PCL5**	**PCL4**	**PCL3**	**PCL2**	**PCL1**	**PCL0**
w	w	w	w	w	w	w	w

Field	Bits	Type	Description
PCLm (m=0-7)	m	w	**Port n Clear Bit m** Setting this bit will clear the corresponding bit in the port output register SCR_Pn_OUT. Read as 0. 0_B No operation 1_B Clears SCR_Pn_OUT.Pm

Table 640 Reset Values of P0n_OMCR (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Note: Register SCR_Pn_OMCR is virtual and does not contain any flip-flops. A read action delivers the value of 0. A 1-bit write behaves as an 8-bit write padded with zeros.

49.10.2.8 Port Output Modification Toggle Register

The port output modification toggle register contains control bits that make it possible to individually toggle the logic state of a single port line by manipulating the output register.

Port P0n Output Modification Toggle Register

P0n_OMTR (n=0-1)
Port P0n Output Modification Toggle Register (094_H+n*8) **Reset Value: Table 641**
RMAP: 0, PAGE: IO_PAGE=0

7	6	5	4	3	2	1	0
PTL7	**PTL6**	**PTL5**	**PTL4**	**PTL3**	**PTL2**	**PTL1**	**PTL0**
w	w	w	w	w	w	w	w

Field	Bits	Type	Description
PTLm (m=0-7)	m	w	**Port n Toggle Bit m** Setting this bit will toggle the corresponding bit in the port output register SCR_Pn_OUT. Read as 0. 0_B No operation 1_B Toggles SCR_Pn_OUT.Pm

Table 641 Reset Values of P0n_OMTR (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Note: Register SCR_Pn_OMTR is virtual and does not contain any flip-flops. A read action delivers the value of 0. A 1-bit write behaves as an 8-bit write padded with zeros.

49.10.2.9 Port Input Register

The logic level of a GPIO pin can be read via the read-only port input register SCR_P0n_IN. Reading the SCR_P0n_IN register always returns the current logical value at the GPIO pin independently whether the pin is selected as input or output.

Port P0n Input Register

P0n_IN (n=0-1)
Port P0n Input Register **(091_H+n*8)** **Reset Value: Table 642**
RMAP: 0, PAGE: IO_PAGE=0

7	6	5	4	3	2	1	0
P7	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
Pm (m=0-7)	m	rh	**Port n Input Bit m** This bit reflects the level at the input pin Pn.m. 0_B The input level of Pn.m is 0 1_B The input level of Pn.m is 1

Table 642 Reset Values of P0n_IN (n=0-1)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.10.3 SCR Functions in Port SCR_P00 (P33L)

This section describes the SCR functions of Port SCR_P00, which is mapped on P33L.

49.10.3.1 SCR Port SCR_P00 Configuration

The SCR has functions on Port lines SCR_P00.0 - SCR_P00.7 (P33.0 - P33.7). It can be alternatively used for UART, WCAN, SSC, T0/T1, T2CCU and External Interrupt input and output functions.

49.10.3.2 Port SCR_P00 Function Table

Table 643 summarizes the SCR I/O control selection functions of the 8 Port SCR_P00 lines, while **Figure 841** provides an overview of the SCR_P00 connections to other modules.

Table 643 Port P00 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Selection	
				Reg./Bit Field	Value
SCR_P00.0 (P33.0)	I	General-purpose input	SCR_P00_IN.P0	SCR_P00_ IOCR0.PC	$0XXXX_B$
		SSC input	MTSRC		
		External interrupt 3 / T2CCU input	EXINT3A/T2CC0A		
	O	General-purpose output	SCR_P00_OUT.P0		$1X000_B$
		UART output	TXD_O		$1X001_B$
		SSC output	MTSR_O		$1X010_B$
		T2CCU output	T2CC0_O		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
SCR_P00.1 (P33.1)	I	General-purpose input	SCR_P00_IN.P1	SCR_P00_ IOCR1.PC	$0XXXX_B$
		SSC input	MRSTC		
		External interrupt 4 / T2CCU input	EXINT4A/T2CC1A		
		External interrupt 1	EXINT1A		
		T0 input	T0A		
		T2CCU input	T2EXA		
	O	General-purpose output	SCR_P00_OUT.P1		$1X000_B$
		SSC output	MRST_O		$1X001_B$
		T2CCU output	T2CC1_O		$1X010_B$
		Reserved	-		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$

Table 643 Port P00 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Selection	
				Reg./Bit Field	Value
SCR_P00.2 (P33.2)	I	General-purpose input	SCR_P00_IN.P2	SCR_P00_ IOCR2.PC	$0XXXX_B$
		External interrupt 5 / T2CCU input	EXINT5A/T2CC2A		
		External interrupt 2	EXINT2A		
		WCAN input	WCANRXDE		
		Debug input 0	DAP0_1		
	O	General-purpose output	SCR_P00_OUT.P2		$1X000_B$
		T2CCU output	T2CC2_O		$1X001_B$
		SSC output	SCLK_O		$1X010_B$
		T2CCU output	EXF2_O		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
SCR_P00.3 (P33.3)	I	General-purpose input	SCR_P00_IN.P3	SCR_P00_ IOCR3.PC	$0XXXX_B$
		External interrupt 6 / T2CCU input	EXINT6A/T2CC3A		
		T2CCU input	T2A		
		Debug input	DAP1_1/SPD_1		
	O	General-purpose output	SCR_P00_OUT.P3		$1X000_B$
		T2CCU output	T2CC3_O		$1X001_B$
		Reserved	-		$1X010_B$
		Reserved	-		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
	DIR	Debug output 1; ENx	DAP1_1/SPD_1	HW_OUT	

Table 643 Port P00 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Selection	
				Reg./Bit Field	Value
SCR_P00.4 (P33.4)	I	General-purpose input	SCR_P00_IN.P4	SCR_P00_ IOCR4.PC	$0XXXX_B$
		External interrupt 7	EXINT7		
		External interrupt 0	EXINT0B		
		T1 input	T1B		
		UART input	RXDC		
		ADCOMP input	ADCOMPCH0		
	O	General-purpose output	SCR_P00_OUT.P4		$1X000_B$
		T2CCU output	T2CC4_O		$1X001_B$
		UART output	RXD_O		$1X010_B$
		Reserved	-		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
SCR_P00.5 (P33.5)	I	General-purpose input	SCR_P00_IN.P5	SCR_P00_ IOCR5.PC	$0XXXX_B$
		External interrupt 8	EXINT8		
		External interrupt 1	EXINT1B		
		WCAN input	WCANRXDD		
		SSC input	MRSTA		
		T0 input	T0B		
		T2CCU input	T2EXB		
		ADCOMP input	ADCOMPCH1		
	O	General-purpose output	SCR_P00_OUT.P5		$1X000_B$
		T2CCU output	T2CC5_O		$1X001_B$
		SSC output	MRST_O		$1X010_B$
		Reserved	-		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$

Table 643 Port P00 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Selection	
				Reg./Bit Field	Value
SCR_P00.6 (P33.6)	I	General-purpose input	SCR_P00_IN.P6	SCR_P00_ IOCR6.PC	$0XXXX_B$
		External interrupt 3 /T2CCU input	EXINT3C/T2CC0C		
		External interrupt 9	EXINT9		
		External interrupt 2	EXINT2B		
		SSC input	MTSRA		
		ADCOMP input	ADCOMPCH2		
		Debug input	DAP0_0		
	O	General-purpose output	SCR_P00_OUT.P6		$1X000_B$
		T2CCU output	T2CC0_O		$1X001_B$
		SSC output	MTSR_O		$1X010_B$
		T2CCU output	EXF2_O		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
SCR_P00.7 (P33.7)	I	General-purpose input	SCR_P00_IN.P7	SCR_P00_ IOCR7.PC	$0XXXX_B$
		External interrupt 4 / T2CCU input	EXINT4C/T2CC1C		
		External interrupt 10	EXINT10		
		T2CCU input	T2B		
		SSC input	SCLKA		
		WCAN input	WCANRXDB		
		ADCOMP input	ADCOMPCH3		
		Debug input	DAP1_0/SPD_0		
	O	General-purpose output	SCR_P00_OUT.P7		$1X000_B$
		T2CCU output	T2CC1_O		$1X001_B$
		UART output	TXD_O		$1X010_B$
		SSC output	SCLK_O		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
	DIR	Debug output 1; ENx	DAP1_0/SPD_0	HW_OUT	

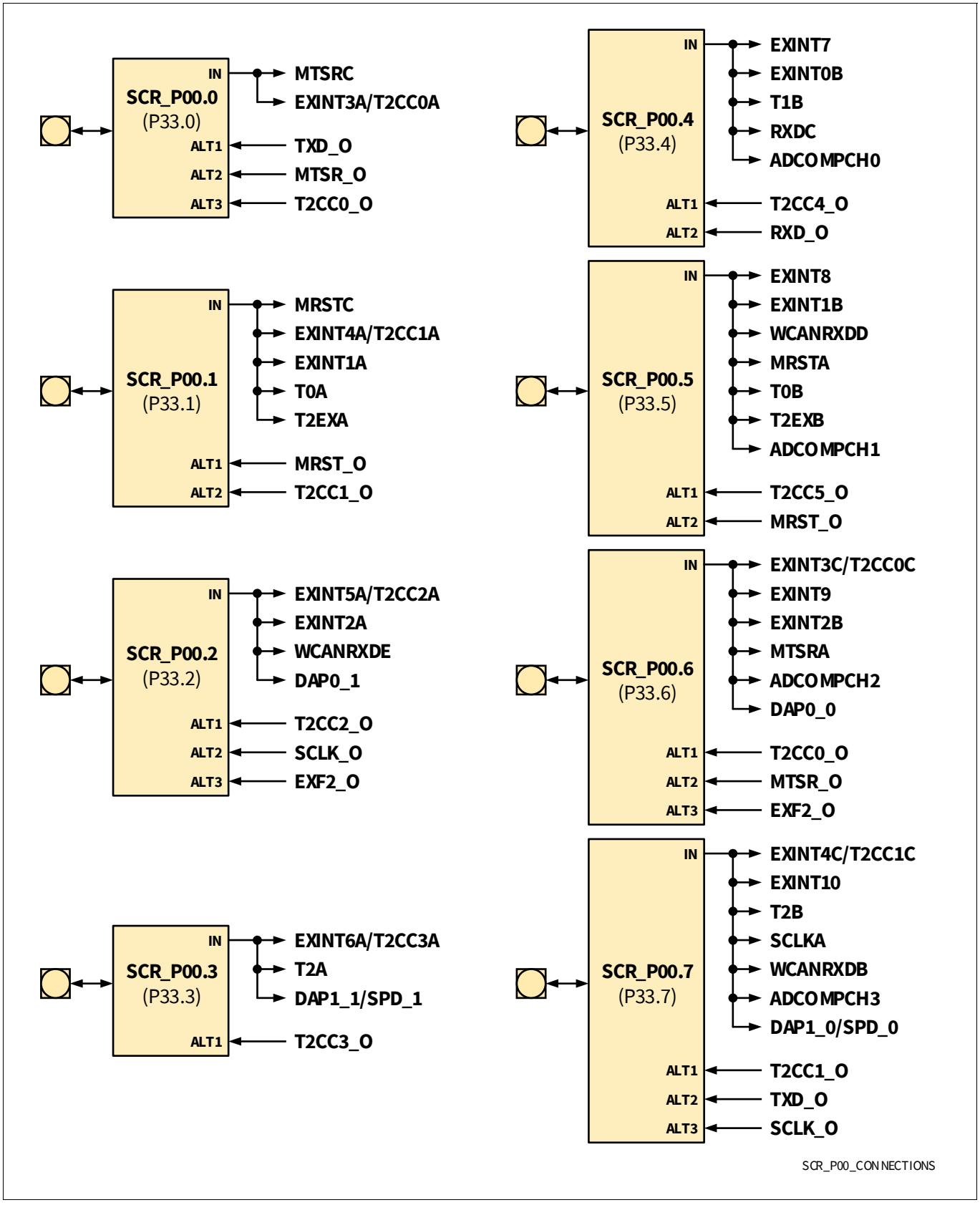


Figure 841 SCR_P00 Connection Overview

49.10.3.3 Port SCR_P00 Registers

The following registers are available on Port SCR_P00 (P33L).

The Port SFRs are located in the standard memory area (RMAP = 0).

Table 644 Register Overview - Port SCR_P00 (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
IO_PAGE	Page Register for Port SFRs	0	X	$08F_H$	134
P00_IN	Port P00 Input Register	0	0	091_H	147
P00_IOCR0	Port P00 Input/Output Control Register 0	0	1	090_H	136
P00_IOCR1	Port P00 Input/Output Control Register 1	0	1	091_H	136
P00_IOCR2	Port P00 Input/Output Control Register 2	0	1	092_H	136
P00_IOCR3	Port P00 Input/Output Control Register 3	0	1	093_H	136
P00_IOCR4	Port P00 Input/Output Control Register 4	0	1	094_H	136
P00_IOCR5	Port P00 Input/Output Control Register 5	0	1	095_H	136
P00_IOCR6	Port P00 Input/Output Control Register 6	0	1	096_H	136
P00_IOCR7	Port P00 Input/Output Control Register 7	0	1	097_H	136
P00_OMCR	Port P00 Output Modification Clear Register	0	0	093_H	145
P00_OMSR	Port P00 Output Modification Set Register	0	0	092_H	144
P00_OMTR	Port P00 Output Modification Toggle Register	0	0	094_H	146
P00_OUT	Port P00 Output Register	0	0	090_H	143
P00_PDISC	Port P00 Pin Function Decision Control Register	0	2	095_H	142
P00_PDR0	Port P00 Pad Driver Mode 0 Register	0	2	090_H	139
P00_PDR2	Port P00 Pad Driver Mode 2 Register	0	2	091_H	139
P00_PDR4	Port P00 Pad Driver Mode 4 Register	0	2	092_H	140
P00_PDR6	Port P00 Pad Driver Mode 6 Register	0	2	093_H	140

49.10.4 SCR Functions in Port SCR_P01 (P34.1 and P33H)

This section describes the SCR functions of Port SCR_P01, which is mapped on P34.1 and P33H.

49.10.4.1 SCR Port SCR_P01 Configuration

The SCR has functions on Port lines SCR_P01.0 - SCR_P01.7 (P34.1, P33.9 - P33.15). It can be alternatively used for UART, WCAN, SSC, T0/T1, T2CCU and External Interrupt input and output functions.

49.10.4.2 Port SCR_P01 Function Table

Table 645 summarizes the SCR I/O control selection functions of the 8 Port SCR_P01 lines.

Table 645 Port P01 Functions

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Selection	
				Reg./Bit Field	Value
SCR_P01.0 (P34.1)	I	General-purpose input	SCR_P01_IN.P0	SCR_P01_ IOCR0.PC	$0XXXX_B$
		UART input	RXDD		
		External interrupt 3 /T2CCU input	EXINT3B/T2CC0B		
	O	General-purpose output	SCR_P01_OUT.P0		$1X000_B$
		UART output	RXD_O		$1X001_B$
		T2CCU output	T2CC0_O		$1X010_B$
		Reserved	-		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
SCR_P01.1 (P33.9)	I	General-purpose input	SCR_P01_IN.P1	SCR_P01_ IOCR1.PC	$0XXXX_B$
		SSC input	SCLKC		
		External interrupt 4 / T2CCU input	EXINT4B/T2CC1B		
		External interrupt 0	EXINT0A		
		T1 input	T1A		
		External NMI	EXTNMI		
		T2CCU input	T2EXC		
	O	General-purpose output	SCR_P01_OUT.P1		$1X000_B$
		UART output	TXD_O		$1X001_B$
		SSC output	SCLK_O		$1X010_B$
		T2CCU output	T2CC1_O		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$

Table 645 Port P01 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Selection	
				Reg./Bit Field	Value
SCR_P01.2 (P33.10)	I	General-purpose input	SCR_P01_IN.P2	SCR_P01_ IOCR2.PC	$0XXXX_B$
		UART input	RXDA		
		WCAN input	WCANRXDA		
		External interrupt 5 / T2CCU input	EXINT5B/T2CC2B		
		External interrupt 0	EXINT0D		
		T2CCU input	T2C		
		External clock input to RTC, limited to 32 kHz and 32.768 kHz oscillators *Note: This input option is NOT available for TC39x and TC38x devices.*	RTC32_IN		
	O	General-purpose output	SCR_P01_OUT.P2		$1X000_B$
		UART output	RXD_O		$1X001_B$
		T2CCU output	T2CC2_O		$1X010_B$
		Reserved	-		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
SCR_P01.3 (P33.11)	I	General-purpose input	SCR_P01_IN.P3	SCR_P01_ IOCR3.PC	$0XXXX_B$
		External interrupt 6 / T2CCU input	EXINT6B/T2CC3B		
		T2CCU input	T2EXH		
		SSC input	SCLKB		
	O	General-purpose output	SCR_P01_OUT.P3		$1X000_B$
		SSC output	SCLK_O		$1X001_B$
		T2CCU output	T2CC3_O		$1X010_B$
		Reserved	-		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$

Table 645 Port P01 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Selection	
				Reg./Bit Field	Value
SCR_P01.4 (P33.12)	I	General-purpose input	SCR_P01_IN.P4	SCR_P01_ IOCR4.PC	$0XXXX_B$
		UART input	RXDG		
		WCAN input	WCANRXDG		
		External interrupt 11	EXINT11		
		T2CCU input	T2EXE		
		External interrupt 0	EXINT0C		
		T1 input	T1C		
		SSC input	MTSRB		
	O	General-purpose output	SCR_P01_OUT.P4		$1X000_B$
		UART output	TXD_O		$1X001_B$
		T2CCU output	T2CC4_O		$1X010_B$
		SSC output	MTSR_O		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
SCR_P01.5 (P33.13)	I	General-purpose input	SCR_P01_IN.P5	SCR_P01_ IOCR5.PC	$0XXXX_B$
		UART input	RXDB		
		WCAN input	WCANRXDC		
		SSC input	MRSTB		
		External interrupt 12	EXINT12		
		External interrupt 1	EXINT1C		
		T0 input	T0C		
	O	General-purpose output	SCR_P01_OUT.P5		$1X000_B$
		UART output	RXD_O		$1X001_B$
		SSC output	MRST_O		$1X010_B$
		T2CCU output	T2CC5_O		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$

Table 645 Port P01 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Selection	
				Reg./Bit Field	Value
SCR_P01.6 (P33.14)	I	General-purpose input	SCR_P01_IN.P6	SCR_P01_ IOCR6.PC	$0XXXX_B$
		External interrupt 5 / T2CCU input	EXINT5C/T2CC2C		
		External interrupt 13	EXINT13		
		External interrupt 2	EXINT2C		
	O	General-purpose output	SCR_P01_OUT.P6		$1X000_B$
		UART output	TXD_O		$1X001_B$
		T2CCU output	T2CC2_O		$1X010_B$
		Reserved	-		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$
SCR_P01.7 (P33.15)	I	General-purpose input	SCR_P01_IN.P7	SCR_P01_ IOCR7.PC	$0XXXX_B$
		UART input	RXDE		
		External interrupt 6 / T2CCU input	EXINT6C/T2CC3C		
		External interrupt 14	EXINT14		
		T2CCU input	T2EXD		
	O	General-purpose output	SCR_P01_OUT.P7		$1X000_B$
		UART output	RXD_O		$1X001_B$
		T2CCU output	T2CC3_O		$1X010_B$
		Reserved	-		$1X011_B$
		Reserved	-		$1X100_B$
		Reserved	-		$1X101_B$
		Reserved	-		$1X110_B$
		Reserved	-		$1X111_B$

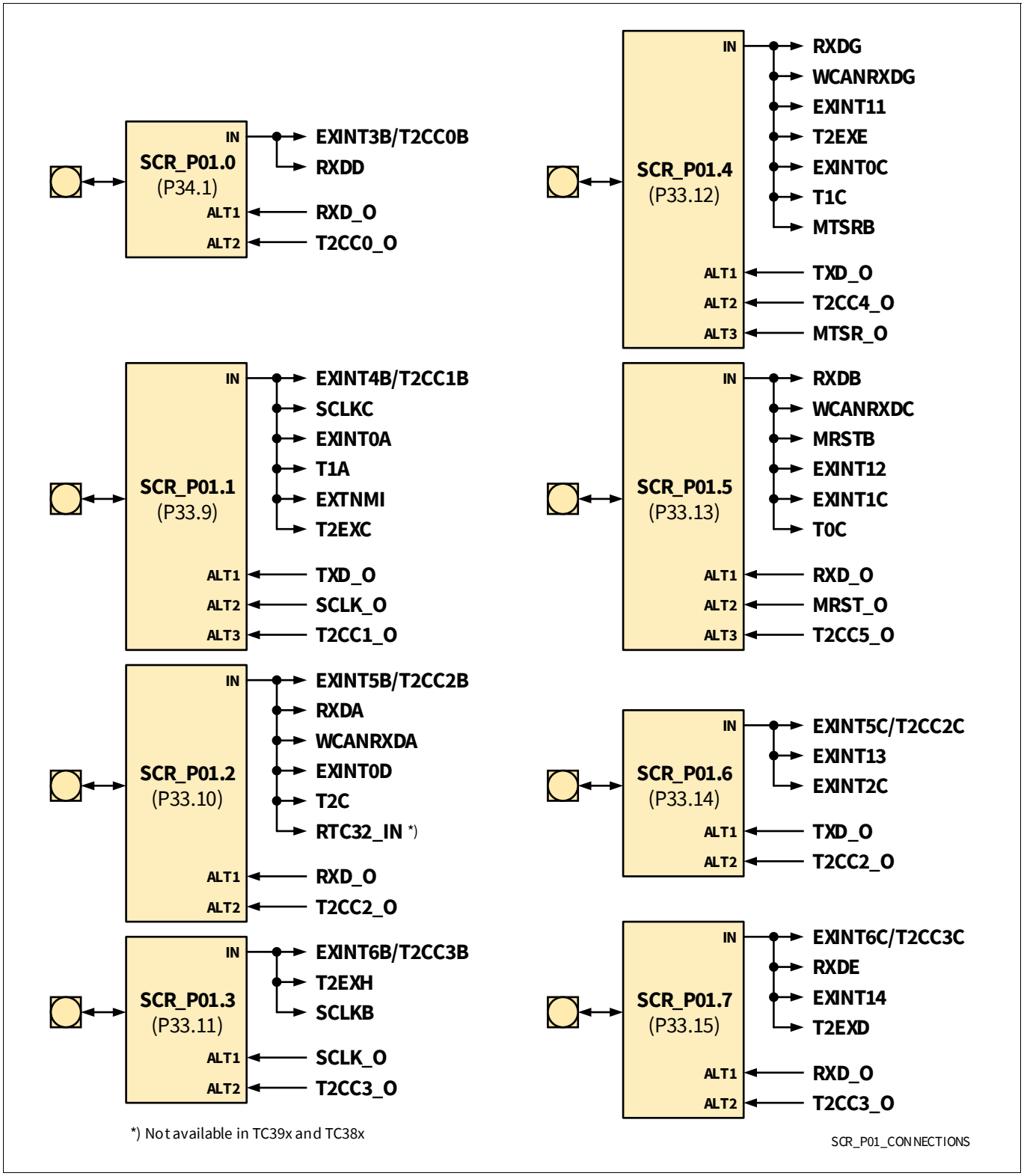


Figure 842 SCR_P01 Connection Overview

49.10.4.3 Port SCR_P01 Registers

The following registers are available on Port SCR_P01 (P34.1 and P33H).

The Port SFRs are located in the standard memory area (RMAP = 0).

Note: P34.1 is used instead of P33.8.

Table 646 Register Overview - Port SCR_P01 (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
IO_PAGE	Page Register for Port SFRs	0	X	$08F_H$	134
P01_IN	Port P01 Input Register	0	0	099_H	147
P01_IOCR0	Port P01 Input/Output Control Register 0	0	1	098_H	136
P01_IOCR1	Port P01 Input/Output Control Register 1	0	1	099_H	136
P01_IOCR2	Port P01 Input/Output Control Register 2	0	1	$09A_H$	136
P01_IOCR3	Port P01 Input/Output Control Register 3	0	1	$09B_H$	136
P01_IOCR4	Port P01 Input/Output Control Register 4	0	1	$09C_H$	136
P01_IOCR5	Port P01 Input/Output Control Register 5	0	1	$09D_H$	136
P01_IOCR6	Port P01 Input/Output Control Register 6	0	1	$09E_H$	136
P01_IOCR7	Port P01 Input/Output Control Register 7	0	1	$09F_H$	136
P01_OMCR	Port P01 Output Modification Clear Register	0	0	$09B_H$	145
P01_OMSR	Port P01 Output Modification Set Register	0	0	$09A_H$	144
P01_OMTR	Port P01 Output Modification Toggle Register	0	0	$09C_H$	146
P01_OUT	Port P01 Output Register	0	0	098_H	143
P01_PDISC	Port P01 Pin Function Decision Control Register	0	2	$09D_H$	142
P01_PDR0	Port P01 Pad Driver Mode 0 Register	0	2	098_H	139
P01_PDR2	Port P01 Pad Driver Mode 2 Register	0	2	099_H	139
P01_PDR4	Port P01 Pad Driver Mode 4 Register	0	2	$09A_H$	140
P01_PDR6	Port P01 Pad Driver Mode 6 Register	0	2	$09B_H$	140

49.10.5 Additional Port Pins Available as Input

This paragraph describes pins, which are not part of the SCR, but are configured via MODPISELn settings.

Table 647 Port Input Functions outside SCR

Port Pin	I/O	Pin Functionality
P14.1	I	UART_RXDF
		WCANRXDF
		EXINT15
		EXINT2D
		T2EXG
SCR_ESR1	I	EXINT1D
		T2EXF

49.10.6 General Port Control

There are a number of control registers for input pin selection of SSC, UART, External interrupts [15:0], T0, T1, T2CCU and WCAN modules. These registers are implemented within the SFR block. **Table 648** provides an overview of these registers.

Table 648 Register Overview - MODPISELx (sorted by Name)

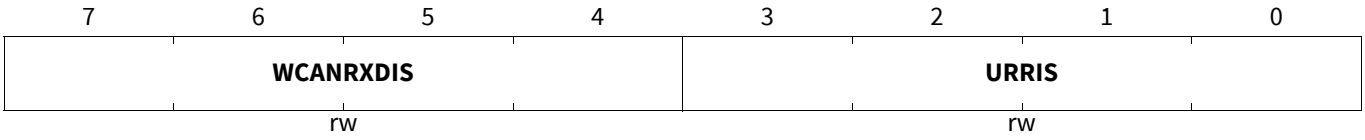
Short Name	Description	RMAP	PAGE	Offset Address	Page Number
MODPISEL0	Peripheral Input Select Register 0	0	2	$0F2_H$	**162**
MODPISEL1	Peripheral Input Select Register 1	0	2	$0F3_H$	**162**
MODPISEL2	Peripheral Input Select Register 2	0	2	$0F4_H$	**163**
MODPISEL3	Peripheral Input Select Register 3	0	2	$0F5_H$	**164**
MODPISEL4	Peripheral Input Select Register 4	0	2	$0F6_H$	**165**
MODPISEL5	Peripheral Input Select Register 5	0	2	$0F7_H$	**166**

49.10.6.1 Input Pin Function Selection

MODPISEL0 register are used for input pin selection of UART and WCAN modules. The register is on SCU_PAGE.

Peripheral Input Select Register 0

MODPISEL0
Peripheral Input Select Register 0 **($0F2_H$)** **Reset Value: Table 649**
RMAP: 0, PAGE: SCU_PAGE=2



Field	Bits	Type	Description
URRIS	3:0	rw	**UART Receiver Input Select** 0_H **UART_RXDA**, UART Receiver Input RXDA is selected 1_H **UART_RXDB**, UART Receiver Input RXDB is selected 2_H **UART_RXDC**, UART Receiver Input RXDC is selected 3_H **UART_RXDD**, UART Receiver Input RXDD is selected 4_H **UART_RXDE**, UART Receiver Input RXDE is selected 5_H **UART_RXDF**, UART Receiver Input RXDF is selected 6_H **UART_RXDG**, UART Receiver Input RXDG is selected **others**, Reserved
WCANRXDIS	7:4	rw	**WCAN Receiver Input Select** 0_H **WCANRXDA**, WCAN Receiver Input WCANRXDA is selected 1_H **WCANRXDB**, WCAN Receiver Input WCANRXDB is selected 2_H **WCANRXDC**, WCAN Receiver Input WCANRXDC is selected 3_H **WCANRXDD**, WCAN Receiver Input WCANRXDD is selected 4_H **WCANRXDE**, WCAN Receiver Input WCANRXDE is selected 5_H **WCANRXDF**, WCAN Receiver Input WCANRXDF is selected 6_H **WCANRXDG**, WCAN Receiver Input WCANRXDG is selected **others**, Reserved

Table 649 Reset Values of MODPISEL0

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Peripheral Input Select Register 1

The MODPISEL1 register is used for input pin selection of the External Interrupt Inputs 0, 1 and 2. The register is on SCU_PAGE.

MODPISEL1
Peripheral Input Select Register 1 **($0F3_H$)** **Reset Value: Table 650**
RMAP: 0, PAGE: SCU_PAGE=2

7	6	5	4	3	2	1	0
EXINT2IS		**EXINT1IS**		**EXINT0IS**		**0**	
rw		rw		rw		r	

Field	Bits	Type	Description
EXINT0IS	3:2	rw	**External Interrupt Input 0 Input Select** 00_B **EXINT0A**, External Interrupt Input EXINT0A is selected 01_B **EXINT0B**, External Interrupt Input EXINT0B is selected 10_B **EXINT0C**, External Interrupt Input EXINT0C is selected 11_B **EXINT0D**, External Interrupt Input EXINT0D is selected
EXINT1IS	5:4	rw	**External Interrupt Input 1 Input Select** 00_B **EXINT1A**, External Interrupt Input EXINT1A is selected 01_B **EXINT1B**, External Interrupt Input EXINT1B is selected 10_B **EXINT1C**, External Interrupt Input EXINT1C is selected 11_B **EXINT1D**, External Interrupt Input EXINT1D is selected
EXINT2IS	7:6	rw	**External Interrupt Input 2 Input Select** 00_B **EXINT2A**, External Interrupt Input EXINT2A is selected 01_B **EXINT2B**, External Interrupt Input EXINT2B is selected 10_B **EXINT2C**, External Interrupt Input EXINT2C is selected 11_B **EXINT2D**, External Interrupt Input EXINT2D is selected
0	1:0	r	**Reserved**

Table 650 Reset Values of MODPISEL1

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

MODPISEL2

MODPISEL2 register are used for input pin selection of T2CCU modules. The register is on SCU_PAGE.

Peripheral Input Select Register 2

The MODPISEL2 register is used for input pin selection of the T2CCU module. The register is on SCU_PAGE.

MODPISEL2
Peripheral Input Select Register 2 **($0F4_H$)** **Reset Value: Table 651**
RMAP: 0, PAGE: SCU_PAGE=2

7	6	5	4	3	2	1	0
T2CC3EXINT6IS		**T2CC2EXINT5IS**		**T2CC1EXINT4IS**		**T2CC0EXINT3IS**	
rw		rw		rw		rw	

Field	Bits	Type	Description
T2CC0EXINT3IS	1:0	rw	**External Interrupt 3/T2CCU Capture Channel 0 Input Select** 00_B **T2CC0A**, External Interrupt 3 Input EXINT0A/T2CCU Capture Channel 0 Input T2CC0A is selected 01_B **T2CC0B**, External Interrupt 3 Input EXINT0B/T2CCU Capture Channel 0 Input T2CC0B is selected 10_B **T2CC0C**, External Interrupt 3 Input EXINT0C/T2CCU Capture Channel 0 Input T2CC0C is selected 11_B **T2CC0D**, Reserved
T2CC1EXINT4IS	3:2	rw	**External Interrupt 4/T2CCU Capture Channel 1 Input Select** 00_B **T2CC1A**, External Interrupt 4 Input EXINT4A/T2CCU Capture Channel 1 Input T2CC1A is selected 01_B **T2CC1B**, External Interrupt 4 Input EXINT4B/T2CCU Capture Channel 1 Input T2CC1B is selected 10_B **T2CC1C**, External Interrupt 4 Input EXINT4C/T2CCU Capture Channel 1 Input T2CC1C is selected 11_B **T2CC1D**, Reserved
T2CC2EXINT5IS	5:4	rw	**External Interrupt 5/T2CCU Capture Channel 2 Input Select** 00_B **T2CC2A**, External Interrupt 5 Input EXINT5A/T2CCU Capture Channel 2 Input T2CC2A is selected 01_B **T2CC2B**, External Interrupt 5 Input EXINT5B/T2CCU Capture Channel 2 Input T2CC2B is selected 10_B **T2CC2C**, External Interrupt 5 Input EXINT5C/T2CCU Capture Channel 2 Input T2CC2C is selected 11_B **T2CC2D**, Reserved
T2CC3EXINT6IS	7:6	rw	**External Interrupt 6/T2CCU Capture Channel 3 Input Select** 00_B **T2CC3A**, External Interrupt 6 Input EXINT6A/T2CCU Capture Channel 3 Input T2CC3A is selected 01_B **T2CC3B**, External Interrupt 6 Input EXINT6B/T2CCU Capture Channel 3 Input T2CC3B is selected 10_B **T2CC3C**, External Interrupt 6 Input EXINT6C/T2CCU Capture Channel 3 Input T2CC3C is selected 11_B **T2CC3D**, Reserved

Table 651 Reset Values of MODPISEL2

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Peripheral Input Select Register 3

The MODPISEL3 register is used for input pin selection of the T2CCU and SSC modules. The register is on SCU_PAGE. The connections of Timer 2 T2EXF and T2EXG are available via PMS.

MODPISEL3
Peripheral Input Select Register 3 **($0F5_H$)** **Reset Value: Table 652**
RMAP: 0, PAGE: SCU_PAGE=2

7	6	5	4	3	2	1	0
CIS			T2IS		T2EXIS		
rw			rw		rw		

Field	Bits	Type	Description
T2EXIS	2:0	rw	**Timer 2 External Input Select** 000_B **T2EXA**, Timer 2 Input T2EXA is selected 001_B **T2EXB**, Timer 2 Input T2EXB is selected 010_B **T2EXC**, Timer 2 Input T2EXC is selected 011_B **T2EXD**, Timer 2 Input T2EXD is selected 100_B **T2EXE**, Timer 2 Input T2EXE is selected 101_B **T2EXF**, Timer 2 Input T2EXF is selected 110_B **T2EXG**, Timer 2 Input T2EXG is selected 111_B **T2EXH**, Timer 2 Input T2EXH is selected
T2IS	4:3	rw	**Timer 2 Input Select** 00_B **T2A**, Timer 2 Input T2A is selected 01_B **T2B**, Timer 2 Input T2B is selected 10_B **T2C**, Timer 2 Input T2C is selected 11_B Reserved
CIS	7:5	rw	**Slave Mode Clock Input Select** 000_B **SCLKA**, SSC Slave Clock Input SCLKA is selected 001_B **SCLKB**, SSC Slave Clock Input SCLKB is selected 010_B **SCLKC**, SSC Slave Clock Input SCLKC is selected **others**, Reserved

Table 652 Reset Values of MODPISEL3

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Peripheral Input Select Register 4

The MODPISEL4 register is used for input pin selection of the SSC module. The register is on SCU_PAGE.

MODPISEL4
Peripheral Input Select Register 4 **($0F6_H$)** **Reset Value: Table 653**
RMAP: 0, PAGE: SCU_PAGE=2

7	6	5	4	3	2	1	0
0		**SIS**			**MIS**		
r		rw			rw		

Field	Bits	Type	Description
MIS	2:0	rw	**Master Mode Receive Input Select** 000_B **MRSTA**, SSC Master Receiver Input MRSTA is selected 001_B **MRSTB**, SSC Master Receiver Input MRSTB is selected 010_B **MRSTC**, SSC Master Receiver Input MRSTC is selected **others**, Reserved
SIS	5:3	rw	**Slave Mode Receive Input Select** 000_B **MTSRA**, SSC Slave Receiver Input MTSRA is selected 001_B **MTSRB**, SSC Slave Receiver Input MTSRB is selected 010_B **MTSRC**, SSC Slave Receiver Input MTSRC is selected **others**, Reserved
0	7:6	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 653 Reset Values of MODPISEL4

Reset Type	Reset Value	Note
LVD Reset	$XX00\ 0000_B$	
Generated Reset	$--00\ 0000_B$	

Peripheral Input Select Register 5

The MODPISEL5 register is used for input pin selection of the T0 and T1 modules. The register is on SCU_PAGE.

MODPISEL5
Peripheral Input Select Register 5 **($0F7_H$)** **Reset Value: Table 654**
RMAP: 0, PAGE: SCU_PAGE=2

7	6	5	4	3	2	1	0
0				**T0IS**		**T1IS**	
r				rw		rw	

Field	Bits	Type	Description
T1IS	1:0	rw	**Timer 1 Input Select** 00_B **T1A**, Timer 1 Input T1A is selected 01_B **T1B**, Timer 1 Input T1B is selected 10_B **T1C**, Timer 1 Input T1C is selected 11_B Reserved
T0IS	3:2	rw	**Timer 0 Input Select** 00_B **T1A**, Timer 0 Input T0A is selected 01_B **T1B**, Timer 0 Input T0B is selected 10_B **T1C**, Timer 0 Input T0C is selected 11_B Reserved
0	7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 654 Reset Values of MODPISEL5

Reset Type	Reset Value	Note
LVD Reset	$X0_H$	
Generated Reset	-0_H	

49.10.7 Revision History

Table 655 Revision History

Reference	Change to Previous Version	Change Request
v3.5		
	First Official Release of completely reworked SCR chapter	
v3.6		
Page 155 **Page 158**	Added input for external 32 kHz / 32.768 kHz oscillator (SCR_P01.2/P33.10)	
Page 130	Added note on available pins	

49.11 Real-Time Clock (RTC)

49.11.1 Overview

One of the SCR's peripherals is the Real-Time Clock (RTC) that, once started, can work independently of the state of the rest of the microcontroller.

Features

- Periodic Wake-up Mode using either the 70 kHz clock or the 100 MHz/DIV clock
- Wake-up source during standby mode, e.g. operating system timer

49.11.2 System Information

This section provides system information relevant to the Real-Time Clock.

49.11.2.1 Interrupt Events and Assignment

Table 656 lists the interrupt event sources from the RTC, and the corresponding event interrupt enable bit and flag bit.

Table 656 RTC Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
RTC compare/wake-up	RTC_CON.ECRTC	RTC_CON.CFRTC

Table 657 shows the interrupt node assignment for each RTC interrupt source.

Table 657 RTC Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
RTC compare/wake-up	IEN1.ECCIP3	–	$6B_H$

49.11.2.2 Module Suspend Control

When the On-Chip Debug Support (OCDS) is in Monitor Mode (MMCR2.MMODE = 1) and the Debug-Suspend signal is active (MMCR2.DSUSP = 1), the timer/counter in RTC module in the SCR can be suspended based on the settings of their corresponding module suspend bits in register MODSUSP.

When suspended, only the timer stops counting as the counter input clock is gated off. The module is still clocked so that module registers are accessible.

49.11.3 Clock Source

RTC clock, f_{RTC}, is selectable between the 70 kHz clock or the 100 MHz/DIV clock via bit RTCCLKSEL in RTC_CON register, or an external 32 kHz / 32.768 kHz oscillator signal supplied via pin SCR_P01.2 (P33.10) can be used, selected via bit RTC32SEL.

Note: The external input option via pin SCR_P01.2 (P33.10) is NOT available in the TC39x, TC38x and TC35x devices. (EXT devices have this clock input.)

Once powered up, these clocks can operate irrespective of the state of the microcontroller. That is, it keeps running even when the SCR has entered idle or the device has entered standby mode.

In case of clock failure, the RTC stops counting.

49.11.4 Real Time Clock Operation

The real-time clock consists of a 41-bit timer (including 9-bit prescaler) which count up. It contains a set of 4 count registers, collectively know at RTC_CNT register, that shows the current count value or the current time of the real-time clock. Before starting the real-time clock, RTC_CNT register can be written with any value. The value written is used as an initial value for the real-time clock when it is started. Another set of registers, RTC_CR register, that consists of 4 registers can be used for interrupt generation. It can also be used to wake-up device from standby mode. The RTC_CR register is also used to store the capture value when a capture event is triggered. The real-time clock is started by setting bit RTCC in the RTC_CON register to 1. This enables the input clock into the real-time clock timer.

The RTC starts counting after 1 PCLK + 4 RTC Clock cycles after the RTCC is set.

Note: Before starting the RTC counter, the compare value in registers RTC_CRn should be set to a value <> 0 (e.g. 0xFFFF). Otherwise, the RTC will not start if it is initially 0, or stop after an overflow to 0.

49.11.4.1 Periodic Wake-up Mode

Figure 843 shows the real-time clock. It consists of a 41-bit general purposes timer. The PCLK is the input clock to the timer.

Before the real-time clock starts to run, RTC_CNT register of the real-time clock can be written with any values. The value written is used as an initial value for the real-time clock timer, when it is started by setting bit RTCC in the RTC_CON register to 1. This bit also enables the input clock into the real-time clock timer. RTC_CNT register is protected by the bit protection scheme as described in the SCU chapter. The initial count value can only be updated when the protection scheme is being disabled in the stop mode (RTCC = 0). The real-time clock's lower 9 bits, which serve as a prescaler into the 32-bit counters, RTC_CNT, are set to an initial value of 000000000_B when the RTC starts to run. The prescaler can be bypassed by setting bit RTPBYP in the RTC_CON register to 1.

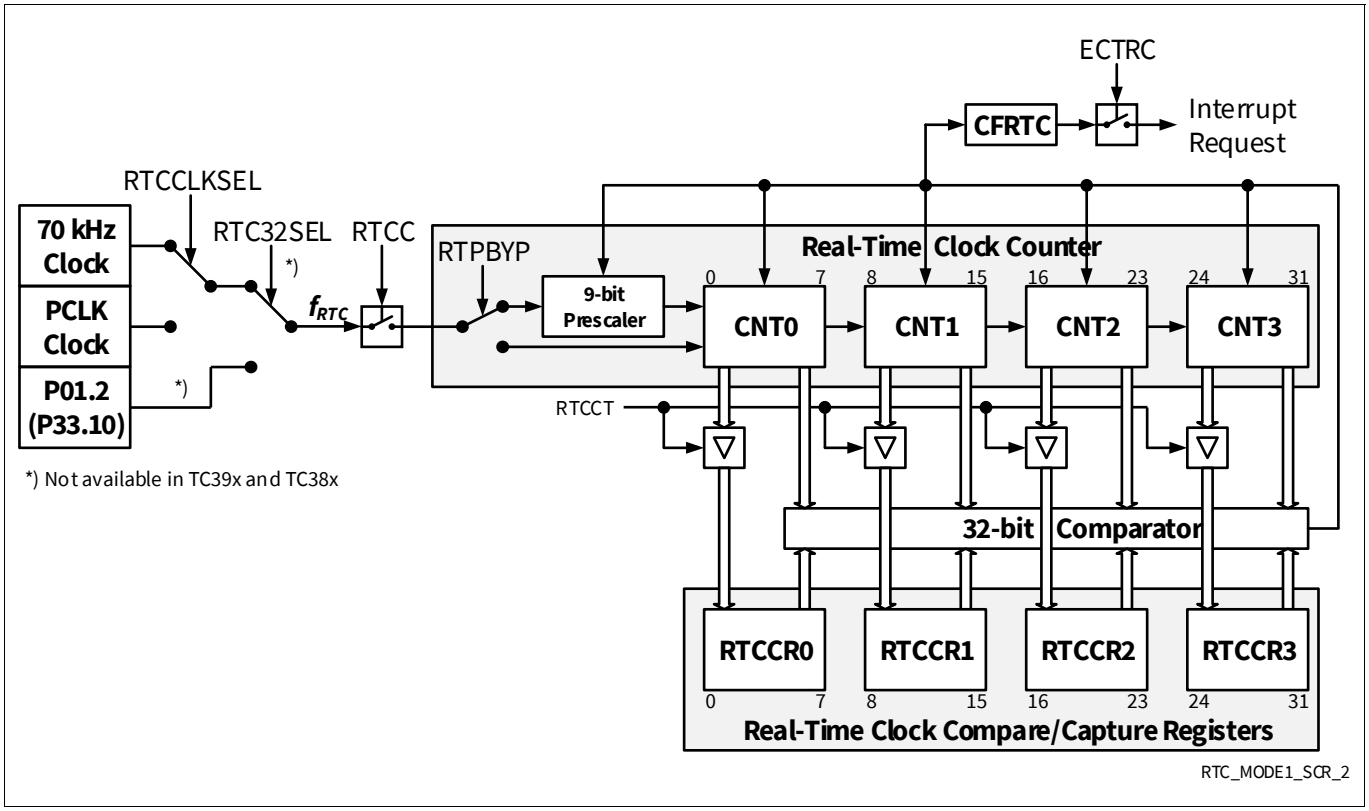


Figure 843 Real-Time Clock Periodic Wake-up Mode with 9-bit Prescaler

While the real-time clock is in operation, the contents of RTC_CR register will be compared to the real-time clock counter (RTC_CNT). An interrupt will be generated when the contents are equal if ECRTC is set to 1; bit CFRTC in register RTC_CON will be set. The CFRTC flag can be monitored for the real-time clock wake-up request, but the flag has to be cleared to 0 by user software. In such situation, the real-time clock counter is reset and starts counting from zero again.

Note: Before starting the RTC counter, the compare value in registers RTC_CRn should be set to a value <> 0 (e.g. 0xFFFF). Otherwise, the RTC will not start if it is initially 0, or stop after an overflow to 0.

With RTC still running in standby mode, it is able to wake-up at a fixed time by programming the number of count value that is equivalent to the length of time to wake-up in the RTC_CR register. The real-time clock can generate a wake-up request to the main controller if bit STDBYWKP.RTCWKSEL is set to 1.

A capture event could be triggered by setting RTCCT bit in RTC_CON register to 1. The previous content in the RTC_CR register will be overwritten with the captured RTC_CNT values after a maximum of 9 f_{RTC} cycles. An update of the actual compared value is necessary once a captured event is triggered. In SCR, it is recommended to trigger a capture event to read the value of the RTC counter (RTC_CNT). There is the potential of reading a wrong 32-bit real-time value while the RTC is in running mode as only 8 bit of data could be fetched at one time.

The real-time clock stops counting and RTC_CNT register holds the last value when the RTCC bit is set to 0. Setting this bit subsequently will start a new counting sequence which begin with the stop count.

Note: For the capture event to complete, up to 9 f_{RTC} cycles are required. During this time window, the compare event is temporarily disabled and no compare match happens. Once the capture is complete and the RTC_CR is updated, the compare event is enabled again. In both cases- i.e.- prescaler bypassed or not bypassed - it is ensured that the next compare match happens only after the next overflow.

49.11.4.2 RTC Access Delays and Restrictions

When accessing the RTC registers, the following delays and restrictions need to be obeyed:

- When the SCR is running and the clock is stable, RTC_CON.RTCC is allowed to be set/cleared.
- When the RTC clock is changed between available clocks, the waiting time of four old/current clock cycles and four clocks of the new/selected clock cycles need to be inserted before setting RTC_CON.RTCC.
  Moreover, upon each clock selection change, it is required to insert up to 9 RTC Clock cycles to the waiting time mentioned in the previous sentence.
- When an SCR reset is triggered, the RTC clock is set to 70 kHz per default. After triggering the reset, software has to wait 2 PCLK + 3 RTC clock cycles, before bit PMCON1.RTC_DIS may be cleared.
- When bit PMCON1.RTC_DIS is cleared, a waiting time of four RTC clock cycles needs to be inserted before any write access to RTC registers may be performed.

When the 9-bit prescaler is bypassed (RTPBYP = 1), then when consecutively reading the RTC count registers, the folowing delays, due to synchronization, need to be taken into account:

- When the RTC is running with PCLK, a delay of 6 PCLK cycles is present between the read accesses; thus, the RTC count value between two read operations may differ by 6 counts.
- When the RTC is running with 70 kHz, a delay of 5 RTC clock cycles is present between the read accesses; thus, the RTC count value between two read operations will differ by 5 counts.

49.11.5 Power Saving Mode Option

Once started, the real-time clock continues counting until the bit RTC_CON.RTCC is cleared. The real-time clock is not affected by the idle mode of the SCR, and continues counting in standby mode provided the PCLK clock is available. In addition, the real-time clock will not automatically disable the operation when there is a clock failure.

49.11.6 RTC Interrupt Request

The following figure illustrates the RTC interrupt request. Note that the flag must be cleared by software. See also the section on “Interrupt Structure 2” in chapter “Interrupt System”.

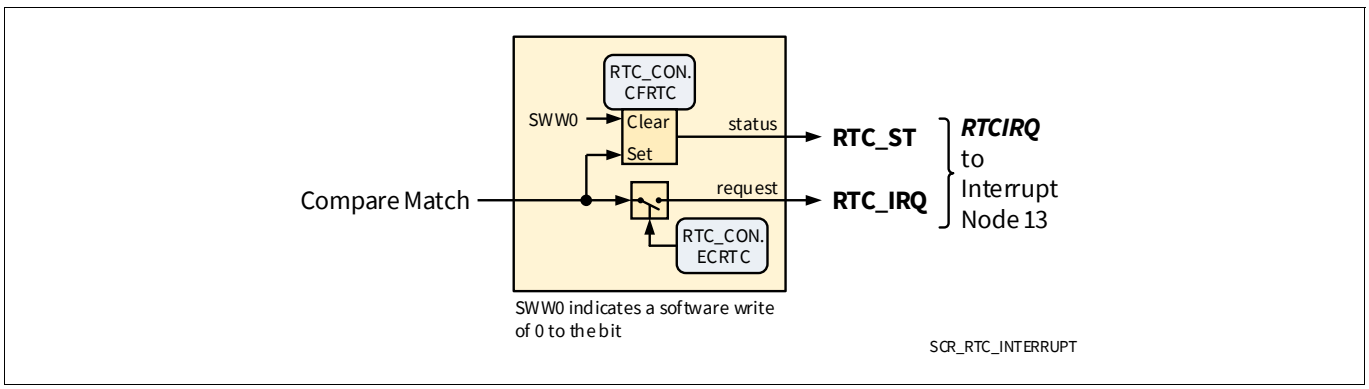


Figure 844 RTC Interrupt Request

49.11.7 Real-Time Clock Register

49.11.7.1 Register Mapping

9 SFRs are used to control the operation of real-time clock. They can be accessed from the standard (non-mapped) SFR area. The registers are described in the following.

Table 658 lists the addresses of these SFRs.

Table 658 Register Overview - RTC (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
RTC_CNTn	Count Clock Register n	0	X	$0E2_H$+n	176
RTC_CON	Real-Time Clock Control Register	0	X	$0E1_H$	174
RTC_CRn	Real-Time Clock Compare/Capture Register n	0	X	$0E6_H$+n	176

49.11.7.2 Register Description

Real-Time Clock Control Register

RTC_CON
Real-Time Clock Control Register **($0E1_H$)** **Reset Value: Table 660**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
INIT32	**CFRTC**	**RTC32SEL**	**ECRTC**	**RTCCT**	**RTPBYP**	**RTCCLKSEL**	**RTCC**
rw	rwh	rw	rw	rwh	rw	rw	rw

Field	Bits	Type	Description
RTCC	0	rw	**Real-Time Clock Start/Stop Control** This bit enables or disables the RTC. After software writes to this bit, it takes 6 RTC Clock cycles for the RTC to be actually enabled or disabled. 0_B Stop Real-Time clock operation 1_B Start Real-Time clock operation
RTCCLKSEL	1	rw	**Real-Time Clock Input Clock Selection** *Note: Real-Time clock must be in stop operation before the clock setting can be changed.* *Note: RTCCLKSEL must be cleared to 0 when the SCR is in standby mode and PCLK is 70 kHz, otherwise, the RTC counting will be disabled.* 0_B 70 kHz clock is selected 1_B PCLK clock is selected
RTPBYP	2	rw	**Real-Time Clock 9-bit Prescaler Bypass** *Note: Real-Time clock must be in stop operation before the clock setting can be changed.* 0_B 9-bit prescaler is not bypassed 1_B 9-bit prescaler is bypassed
RTCCT	3	rwh	**Real-Time Clock Capture Event Trigger** When set, this bit is held until the capture occurs, and then cleared by hardware. Software has to poll this bit until it is cleared to ensure that the capture has occured. 0_B No action 1_B Capture event is triggered
ECRTC	4	rw	**Real-Time Clock Compare Interrupt Enable** 0_B Disable Real-Time clock compare interrupt 1_B Enable Real-Time clock compare interrupt

Field	Bits	Type	Description
RTC32SEL	5	rw	**External Oscillator Selection (32 kHz / 32.768 kHz)** *Note: Real-Time clock must be in stop operation before the clock setting can be changed.* RTC32SEL selects the external clock input. This clock input is on pin SCR_P01.2 (P33.10), and can be 32.768 kHz or 32 kHz . This bit is write protected by bit INIT32 in this register. Before writing to RTC32SEL, INIT32 has to be set. INIT32 has to be cleared after setting RTC32SEL. 0_B External clock oscillator input is not selected; internal clock as configured by RTCCLKSEL is selected 1_B External clock oscillator input is selected for the RTC
CFRTC	6	rwh	**Real-Time Clock Compare Flag** This bit is set by hardware when there is a compare match, and can only be cleared by software. A wake-up request is generated only if the SCR is in standby mode. Writing 0 clears CFRTC. Writing 1 has no effect.
INIT32	7	rw	**RTC32SEL Protection Bit** INIT32 is the protection bit for bit RTC_CON.RTC32SEL. To write RTC32SEL, INIT32 needs to be set. INIT32 has to be cleared after setting RTC32SEL. 0_B RTC32SEL not writable 1_B RTC32SEL is writable

Table 659 Access Mode Restrictions of RTC_CON sorted by descending priority

Mode Name	Access Mode		Description
otherwise	r	RTC32SEL	
RTC_CON.INIT32 = 1 (default)	rw	RTC32SEL	Access protection to RTC32SEL

Table 660 Reset Values of RTC_CON

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Note: The external input option via pin SCR_P01.2 (P33.10) is NOT available in the TC39x. TC38x and TC35x devices. Thus, bits RTC32SEL and INIT32 in register RTC_CON are also not available in these devices, and the respective bit positions are read-only. EXT devices will have the clock input.

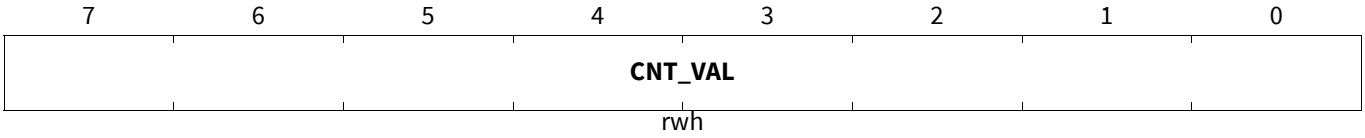
Note: If the application changes back from external RTC oscillator to PCLK AND at the same time, the SCR is waking up (from 70kHz to PCLK), then the RTC clock is for 3 cycles at 70kHz, before changing to PCLK.

Count Clock Register n

4 count registers, RTC_CNT0 - RTC_CNT3, are used to represent the most significant 32 bits of the 41-bit real-time clock timer.

Note: When RTC_CNTn is written, a minimum delay of 8 RTC clock cycles is required, until the next write access to the register is allowed. Otherwise, the second write access will be lost.

RTC_CNTn (n=0-3)
Count Clock Register n **($0E2_H$+n)** **Reset Value: Table 662**
RMAP: 0, PAGE: X



Field	Bits	Type	Description
CNT_VAL	7:0	rwh	**Real-Time Clock Count Value** These bits represent the current 32-bit real-time clock value. The assignment is as follows: • RTC_CR3: Bits [31:24] • RTC_CR2: Bits [23:16] • RTC_CR1: Bits [15:8] • RTC_CR0: Bits [7:0]

Table 661 Access Mode Restrictions of RTC_CNTn (n=0-3) sorted by descending priority

Mode Name	Access Mode		Description
otherwise	rh	CNT_VAL	
PASSWD.PROTECT_S = 0 (default)	rwh	CNT_VAL	

Table 662 Reset Values of RTC_CNTn (n=0-3)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Real-Time Clock Compare/Capture Register n

4 compare and capture registers, RTC_CR0 - RTC_CR3, are used to represent 32 bits of compare value,which will generate a compare event when it matches the counter value as specified in the RTC_CNT register.

When a capture event is triggered by setting RTCCT bit to 1, the content in the RTC_CRn register will be overwritten with the captured RTC_CNT value after a maximum of 8 f_{RTC} clock cycles. In order to take effect of the

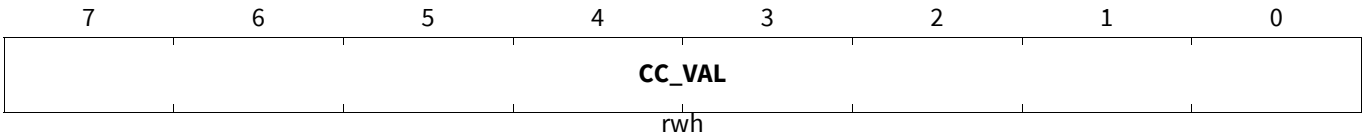
new capture value inside RTC_CRn, another delay of 9 RTC clock cycles is required. During this time, the compare interrupt is disabled.

If the prescaler is bypassed, then the captured value inside the register is 2 or 3 clock cycles higher in relation to the point in time of the trigger set in RTCCT. Otherwise, the value is exact or 1 clock cycle higher than the point in time of the trigger.

Note: When RTC_CRn is written, a minimum delay of 8 RTC clock cycles is required, until the next write access to the register is allowed. Otherwise, the second write access is not guaranteed.

Note: Before starting the RTC counter, the compare value in registers RTC_CRn should be set to a value different to 0 (e.g. 0xFFFF). Otherwise, the RTC will not start if it is initially 0, or stop after an overflow to 0.

RTC_CRn (n=0-3)
Real-Time Clock Compare/Capture Register n (0E6$_H$+n) **Reset Value: Table 663**
RMAP: 0, PAGE: X



Field	Bits	Type	Description
CC_VAL	7:0	rwh	**Compare/Capture Value** These bits represent the 32-bit compare/capture value. The assignment is as follows: • RTC_CR3: Bits [31:24] • RTC_CR2: Bits [23:16] • RTC_CR1: Bits [15:8] • RTC_CR0: Bits [7:0]

Table 663 Reset Values of RTC_CRn (n=0-3)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.11.8 Revision History

Table 664 Revision History

Reference	Change to Previous Version	
v4.4		
	First Official Release of completely reworked SCR chapter	
v4.5		
Page 169 **Page 171** **Page 174** **Page 175**	Added option for external 32 kHz / 32.768 kHz oscillator input	
v4.6		
Page 172	Updated sentence on RTC clock selection change	
v4.7		
Page 169	Updated note concerning devices. TC35x has no external clock input. EXT devices will have clock input.	
Page 174	Note concerning switching behavior. Only valid if switching from a precise clock to PCLK	

49.12 Timer 0 and Timer 1

49.12.1 Overview

Timer 0 and Timer 1 can function as both, timers or counters. When functioning as a timer, Timer 0 and Timer 1 are incremented at $f_{PCLK}/2$ (i.e. every 2 PCLKs). When functioning as a counter, Timer 0 and Timer 1 are incremented in response to a 1-to-0 transition (falling edge) at their respective external input pins, T0 or T1.

They are useful in many timing applications such as measuring the time interval between events, counting events and generating signals at regular intervals. In particular, Timer 1 can be used as the baudrate generator for the on-chip serial port.

Features:

- Four operational modes:
  - Mode 0: 13-bit timer/counter
  - Mode 1: 16-bit timer/counter
  - Mode 2: 8-bit timer/counter with auto-reload
  - Mode 3: Two 8-bit timers/counters

49.12.2 System Information

This section provides system information relevant to the Timer 0 and 1.

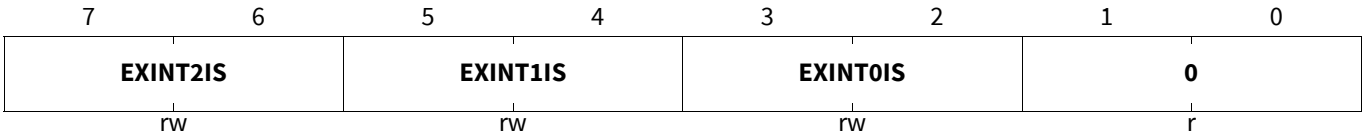
49.12.2.1 Pinning

This section describes the GPIO connections of timers 0 and 1. Please see also the GPIO chapter.

Peripheral Input Select Register 1

The MODPISEL1 register is used for input pin selection of the External Interrupt Inputs 0, 1 and 2. The register is on SCU_PAGE.

MODPISEL1
Peripheral Input Select Register 1 **($0F3_H$)** **Reset Value: Table 665**
RMAP: 0, PAGE: SCU_PAGE=2



Field	Bits	Type	Description
EXINT0IS	3:2	rw	**External Interrupt Input 0 Input Select** 00_B **EXINT0A**, External Interrupt Input EXINT0A is selected 01_B **EXINT0B**, External Interrupt Input EXINT0B is selected 10_B **EXINT0C**, External Interrupt Input EXINT0C is selected 11_B **EXINT0D**, External Interrupt Input EXINT0D is selected

Field	Bits	Type	Description
EXINT1IS	5:4	rw	**External Interrupt Input 1 Input Select** 00_B **EXINT1A**, External Interrupt Input EXINT1A is selected 01_B **EXINT1B**, External Interrupt Input EXINT1B is selected 10_B **EXINT1C**, External Interrupt Input EXINT1C is selected 11_B **EXINT1D**, External Interrupt Input EXINT1D is selected
EXINT2IS	7:6	rw	**External Interrupt Input 2 Input Select** 00_B **EXINT2A**, External Interrupt Input EXINT2A is selected 01_B **EXINT2B**, External Interrupt Input EXINT2B is selected 10_B **EXINT2C**, External Interrupt Input EXINT2C is selected 11_B **EXINT2D**, External Interrupt Input EXINT2D is selected
0	1:0	r	**Reserved**

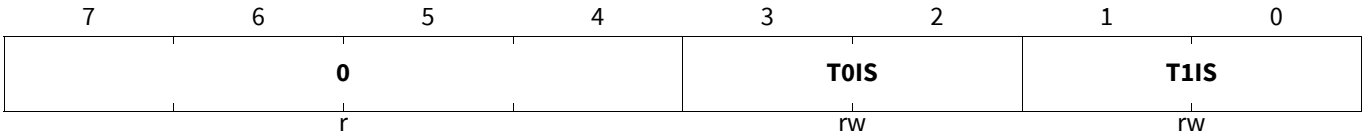
Table 665 Reset Values of MODPISEL1

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Peripheral Input Select Register 5

The MODPISEL5 register is used for input pin selection of the T0 and T1 modules. The register is on SCU_PAGE.

MODPISEL5
Peripheral Input Select Register 5 ($0F7_H$) Reset Value: Table 666
RMAP: 0, PAGE: SCU_PAGE=2



Field	Bits	Type	Description
T1IS	1:0	rw	**Timer 1 Input Select** 00_B **T1A**, Timer 1 Input T1A is selected 01_B **T1B**, Timer 1 Input T1B is selected 10_B **T1C**, Timer 1 Input T1C is selected 11_B Reserved
T0IS	3:2	rw	**Timer 0 Input Select** 00_B **T1A**, Timer 0 Input T0A is selected 01_B **T1B**, Timer 0 Input T0B is selected 10_B **T1C**, Timer 0 Input T0C is selected 11_B Reserved
0	7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 666 Reset Values of MODPISEL5

Reset Type	Reset Value	Note
LVD Reset	$X0_H$	
Generated Reset	-0_H	

Timer 0/1 Inputs Overview

The following figures shows the connections and selections for the inputs to timers 0 and 1.

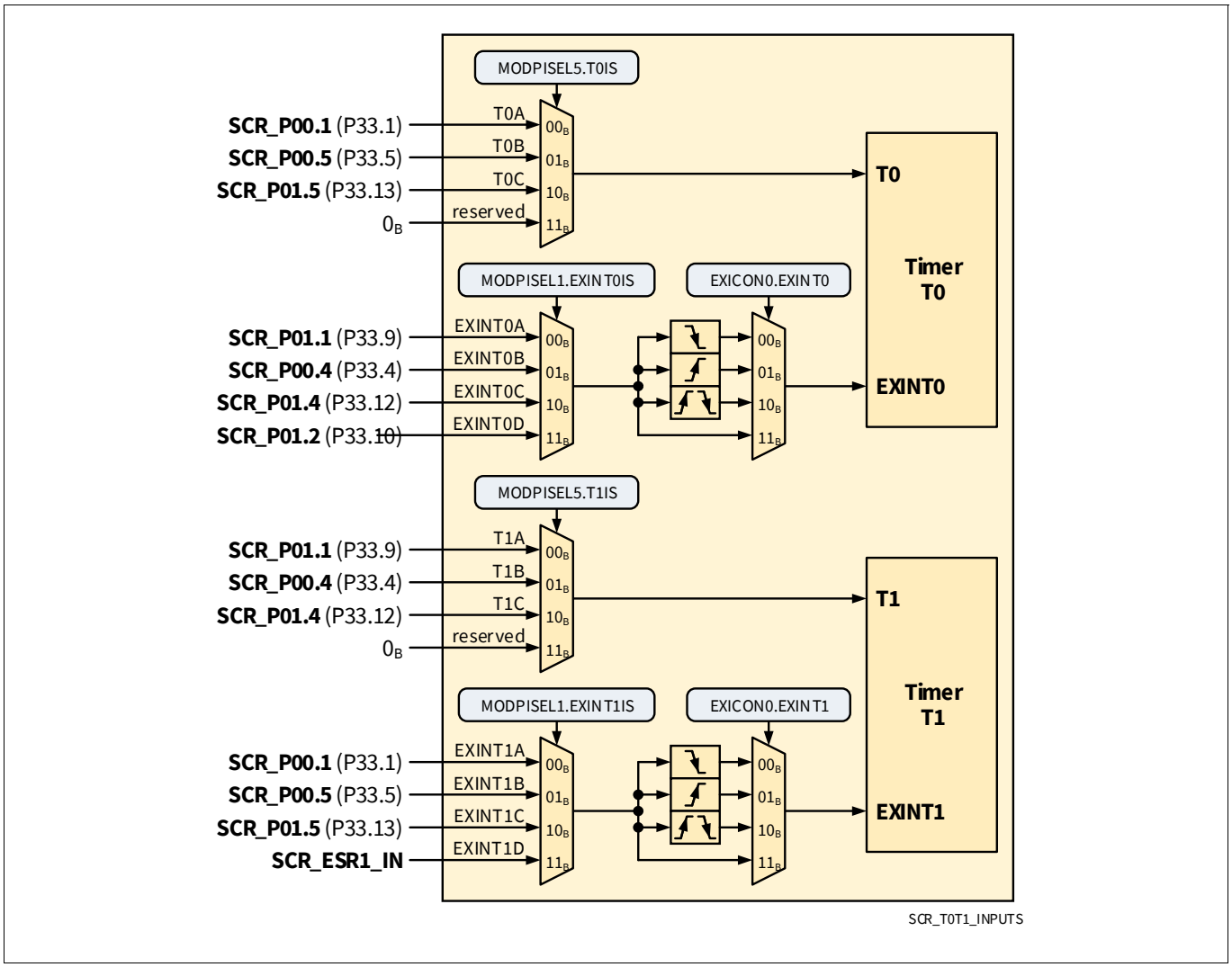


Figure 845 Timer 0 and 1 Modules Input Signals

49.12.2.2 Clocking Configuration

The Timer 0 and 1 run on the PCLK.

49.12.2.3 Interrupt Events and Assignment

Table 667 lists the interrupt event sources from Timer 0 and 1, and the interrupt node assignment for each Timer 0 and 1 interrupt source.

Table 667 Timer 0 and 1 Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
External Interrupt 0	IEN0.EX0	T01_TCON.IE0	03_H
Timer 0 Overflow	IEN0.ET0	T01_TCON.TF0	$0B_H$
External Interrupt 1	IEN0.EX1	T01_TCON.IE1	13_H
Timer 1 Overflow	IEN0.ET1	T01_TCON.TF1	$1B_H$

49.12.3 Basic Timer Operations

The operations of the two timers are controlled using the Special Function Registers (SFRs) T01_TCON and T01_TMOD. To enable a timer, i.e., allow the timer to run, its control bit T01_TCON.TRx is set. To select the timer input to be either from internal system clock or external pin, the input selector bit T01_TMOD.TxS is used.

Note: The "x" (e.g., T01_TCON.TRx) in this chapter denotes either 0 or 1.

Each timer consists of two 8-bit registers - T01_TLx (low byte) and T01_THx (high byte) which defaults to 00_H on reset. Setting or clearing T01_TCON.TRx does not affect the timer registers.

The timers 0 and 1 can be used as a counter, where the timer register is incremented in response to a 1-to-0 transition (falling edge) at the corresponding external input pin (T0 or T1). When the samples show a high for two PCLKs and a low in the next two PCLKs, the count is incremented. Since it takes four PCLKs to recognize a 1-to-0 transition, the maximum count rate is 1/4 of the PCLK frequency. There are no restrictions on the duty cycle of the external input signal, but it must be held at least two full PCLKs to ensure that a given level is sampled at least once before it changes.

Timer Overflow

When a timer overflow occurs, the timer overflow flag, T01_TCON.TFx, is set, and an interrupt may be raised if the interrupt enable control bit, IEN0.ETx, is set. The overflow flag is automatically cleared when the interrupt service routine is entered. See also **Section 49.12.5**.

When Timer 0 operates in mode 3, the Timer 1 control bits, TR1, TF1 and ET1 are reserved for TH0, see **Section 49.12.4.4**.

External Control

In addition to pure software control, the timers can also be enabled or disabled through external port control. When external port control is used, SFR EXICON0 must first be configured to bypass the edge detection circuitry for EXINTx to allow direct feed-through. When the timer is enabled (T01_TCON.TRx = 1) and T01_TMOD.GATEx is set, the respective timer will only run if the core external interrupt EXINTx = 1. This facilitates pulse width measurements. However, this is not applicable for Timer 1 in mode 3.

If T01_TMOD.GATEx is cleared, the timer reverts to pure software control.

49.12.4 Timer Modes

Timers 0 and 1 are fully compatible and can be configured in four different operating modes, as shown in **Table 668**. The bit field TxM in register T01_TMOD selects the operating mode to be used for each timer.

In modes 0, 1 and 2, the two timers operate independently, but in mode 3, their functions are specialized.

Table 668 Timer 0 and Timer 1 Modes

Mode	Operation
0	**13-bit timer/counter** The timer is essentially an 8-bit counter with a divide-by-32 prescaler. This mode is included solely for compatibility with Intel 8048 devices.
1	**16-bit timer/counter** The timer registers, T01_TLx and T01_THx, are concatenated to form a 16-bit timer/ counter.
2	**8-bit timer/counter with auto-reload** The timer register T01_TLx is reloaded with a user-defined 8-bit value in T01_THx upon overflow.
3	**Timer 0 operates as two 8-bit timers/counters** The timer registers, T01_TL0 and T01_TH0, operate as two separate 8-bit counters. Timer 1 is halted and retains its count even if enabled.

49.12.4.1 Mode 0

Putting either Timer 0 or Timer 1 into mode 0 configures it as an 8-bit timer/counter with a divide-by-32 prescaler. **Figure 846** shows the mode 0 operation.

In this mode, the timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the timer overflow flag TFx (see exception below). The overflow flag TFx can then be used to request an interrupt (see **Section 49.12.5**). The count input is enabled for the timer when TRx = 1 and either GATEx = 0 or EXINTx = 1 (setting GATEx = 1 allows the timer to be controlled by external input EXINTx to facilitate pulse width measurements). TRx is a control bit in the register TCON; bit GATEx is in register T01_TMOD.

The 13-bit register consists of all the 8 bits of T01_THx and the lower 5 bits of T01_TLx. The upper 3 bits of T01_TLx are indeterminate and should be ignored. Setting the run flag (TRx) does not clear the registers.

Mode 0 operation is the same for Timer 0 and Timer 1 (exception see below)

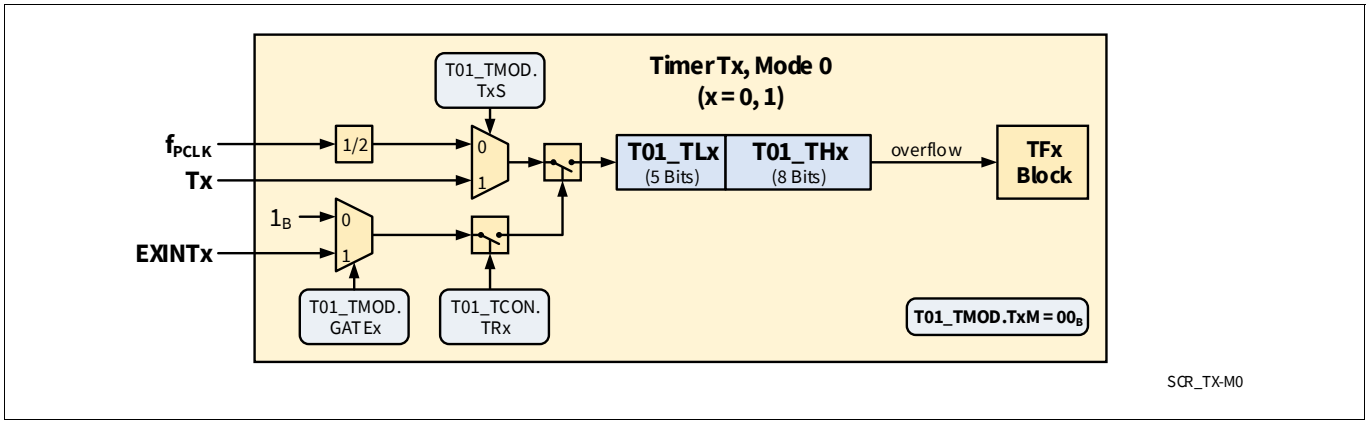


Figure 846 Timer x, Mode 0: 13-Bit Timer/Counter

The configuration shown in **Figure 846** above applies to Timer 1 as long as Timer 0 is NOT operating in mode 3. When Timer 0 is operating in mode 3, it uses resources from Timer 1; thus, the mode 0 configuration of Timer 1 in this case is shown in **Figure 847**.

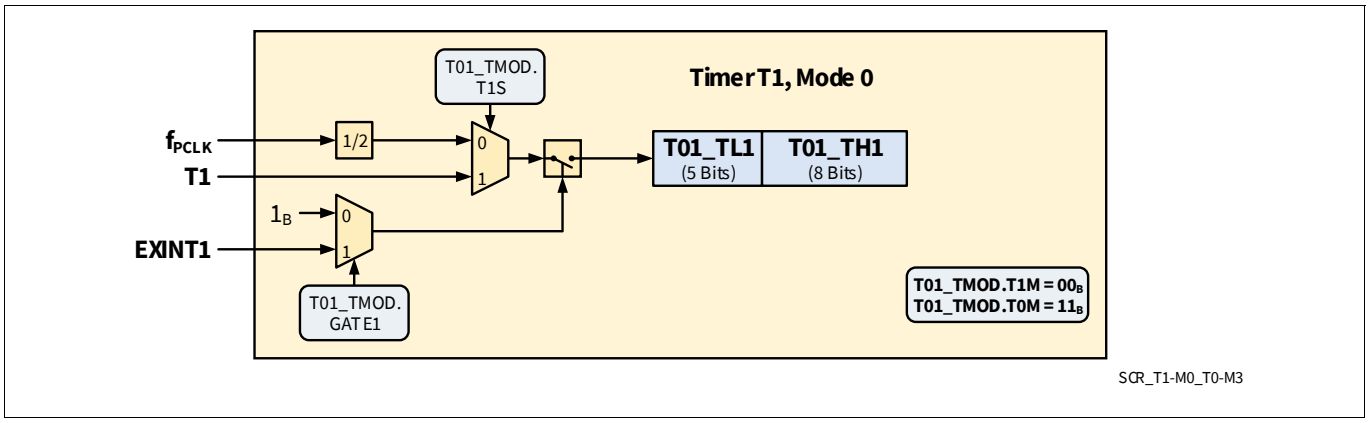


Figure 847 Timer 1 in Mode 0 while Timer 0 is in Mode 3: 13-bit Timer/Counter

49.12.4.2 Mode 1

Mode 1 operation is similar to that of mode 0, except that the timer register runs with all 16 bits. Mode 1 operation for Timer x is shown in **Figure 848**.

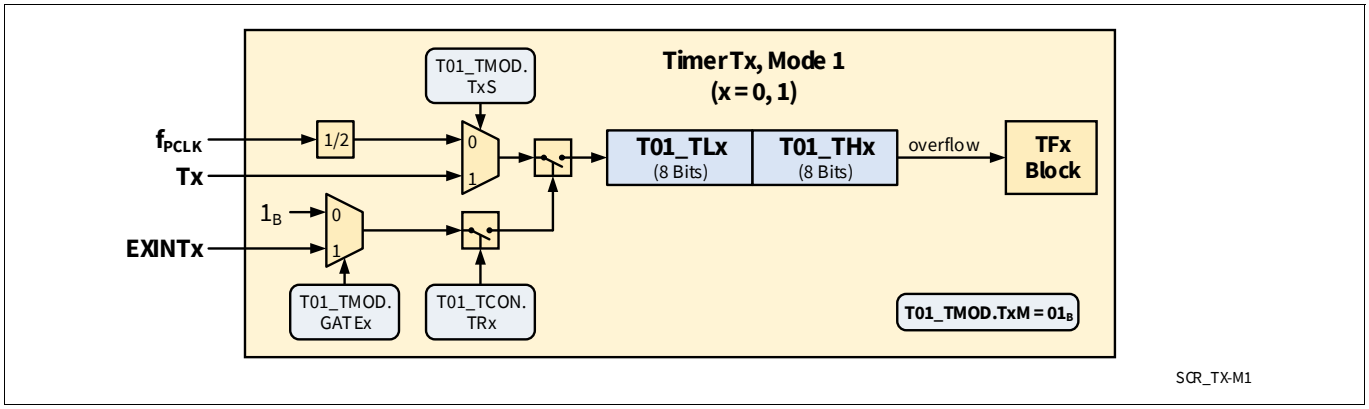


Figure 848 Timer x, Mode 1: 16-Bit Timer/Counter

The configuration shown in **Figure 848** above applies to Timer 1 as long as Timer 0 is NOT operating in mode 3. When Timer 0 is operating in mode 3, it uses resources from Timer 1; thus, the mode 1 configuration of Timer 1 in this case is shown in **Figure 849**.

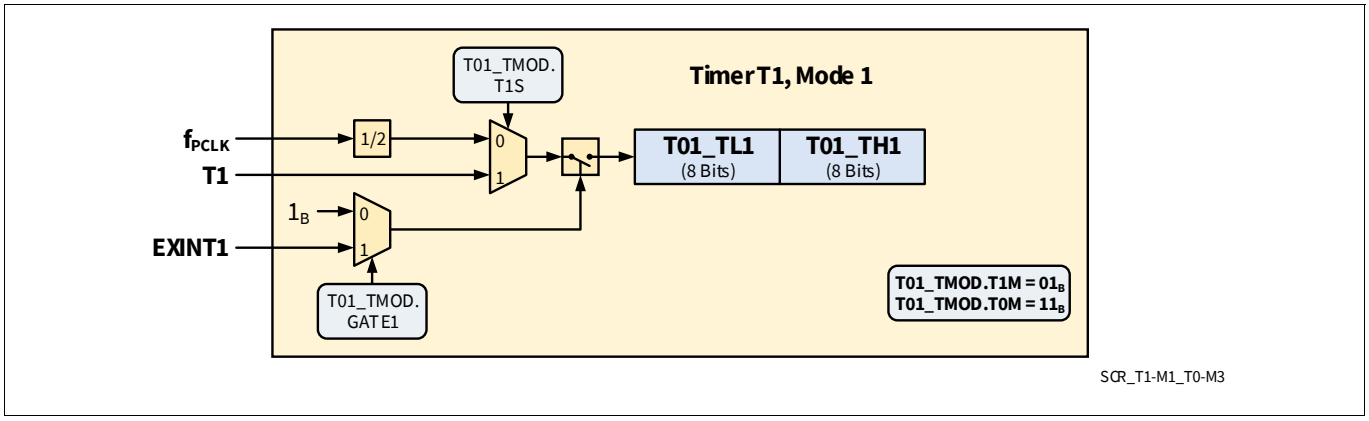


Figure 849 Timer 1 in Mode 1 while Timer 0 is in Mode 3: 16-bit Timer/Counter

49.12.4.3 Mode 2

In Mode 2 operation, the timer is configured as an 8-bit counter (TLx) with automatic reload, as shown in **Figure 850**.

An overflow from TLx not only sets TFx, but also reloads TLx with the contents of THx that has been preset by software. The reload leaves THx unchanged.

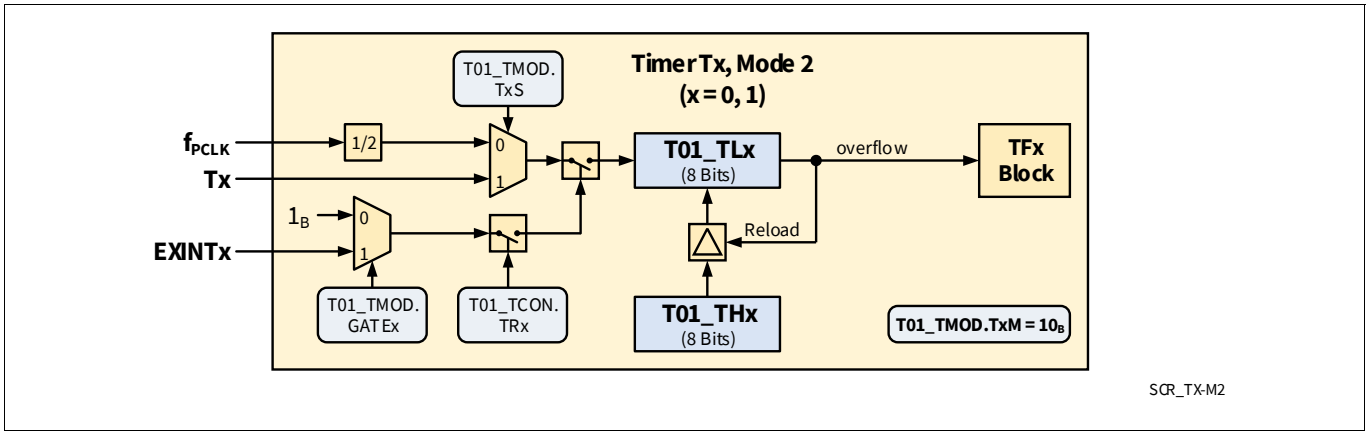


Figure 850 Timer x, Mode 2: 8-Bit Timer/Counter with Auto-Reload

The configuration shown in **Figure 850** above applies to Timer 1 as long as Timer 0 is NOT operating in mode 3. When Timer 0 is operating in mode 3, it uses resources from Timer 1; thus, the mode 2 configuration of Timer 1 in this case is shown in **Figure 851**.

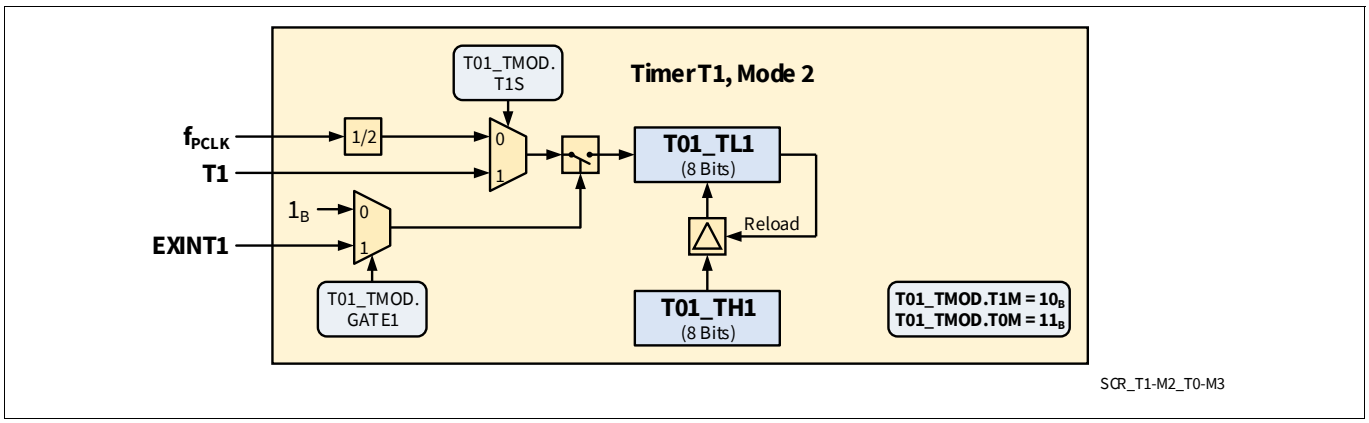


Figure 851 Timer 1 in Mode 2 while Timer 0 is in Mode 3: 8-bit Timer/Counter

49.12.4.4 Mode 3

In mode 3, Timer 0 and Timer 1 behave differently. Timer 0 in mode 3 establishes TL0 and TH0 as two separate counters. Timer 1 in mode 3 simply holds its count. The effect is the same as setting TR1 = 0.

The logic for mode 3 operation for Timer 0 is shown in **Figure 852**. TL0 uses the Timer 0 control bits GATE0, TR0 and TF0, while TH0 is locked into a timer function (counting $f_{PCLK}/2$) and takes over the use of TR1 and TF1 from Timer 1. Thus, TH0 now sets TF1 upon overflow and generates an interrupt if ET1 is set (see also **Section 49.12.5**).

Mode 3 is provided for applications requiring an extra 8-bit timer. When Timer 0 is in mode 3 and TR1 is set, Timer 1 can be turned on by switching it to any of the other modes and turned off by switching it into mode 3.

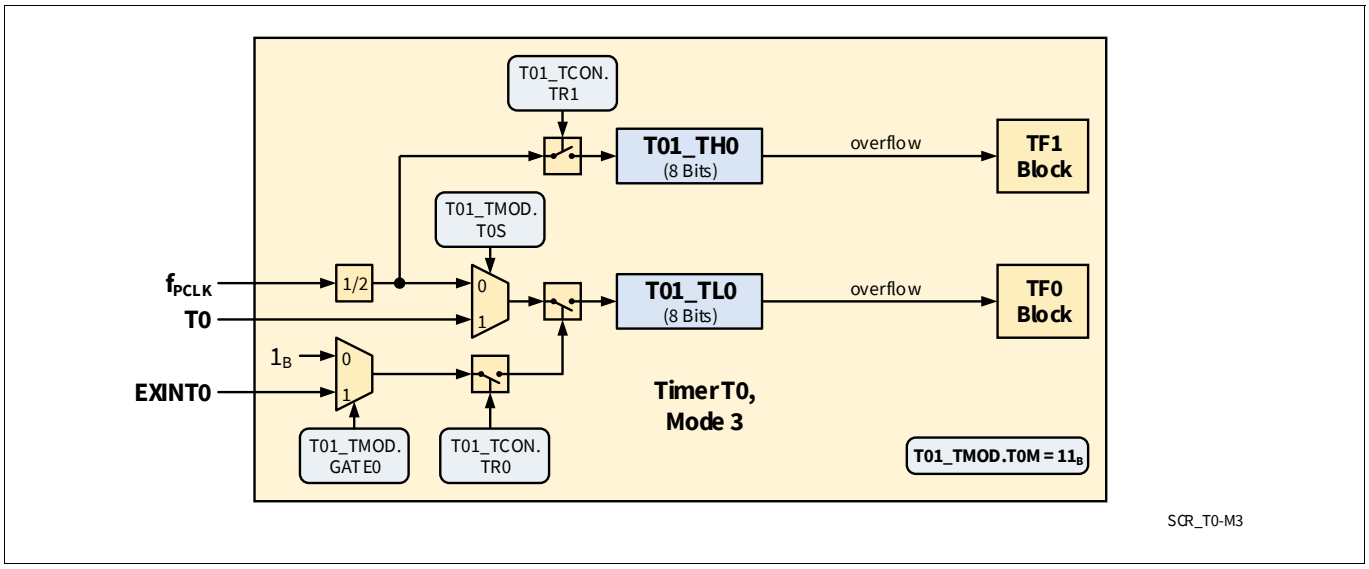


Figure 852 Timer 0, Mode 3: Two 8-Bit Timers/Counters

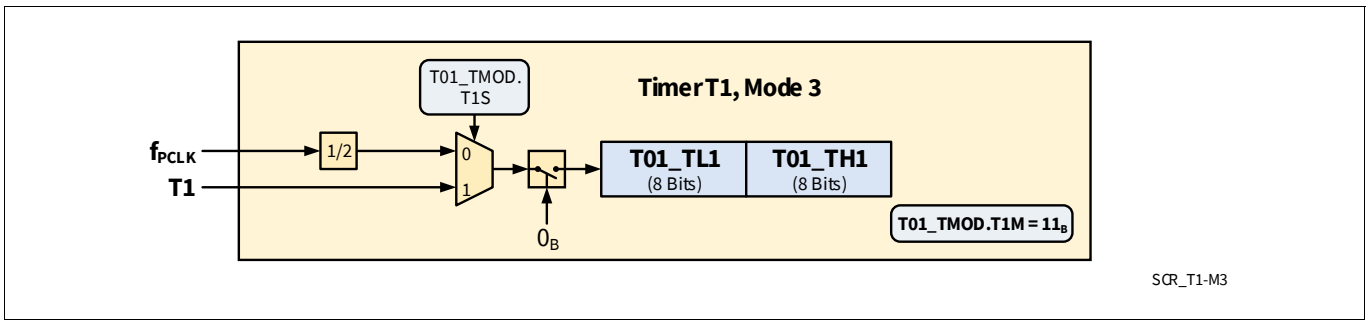


Figure 853 Timer 1 in Mode 3: Timer stops

49.12.5 T0/T1 Interrupt Flags and Requests

The following figure illustrates the T0/T1 interrupt requests. Besides the overflow interrupts, the T0/T1 module also handles the control for the two external interrupts EXINT0 and EXINT1.

Note that when timer T0 operates in Mode 3, flag TF0 is set by an overflow of the 8-bit timer part TL0, while flag TF1 is set by an overflow of the 8-bit timer part TH0. Besides being set by hardware, the flags can also be set by software (triggering an interrupt request).

The interrupt flags are automatically cleared by the CPU on entry into the respective service routine. They can also be cleared by software. See also the section on “Interrupt Structure 1” in chapter “Interrupt System”.

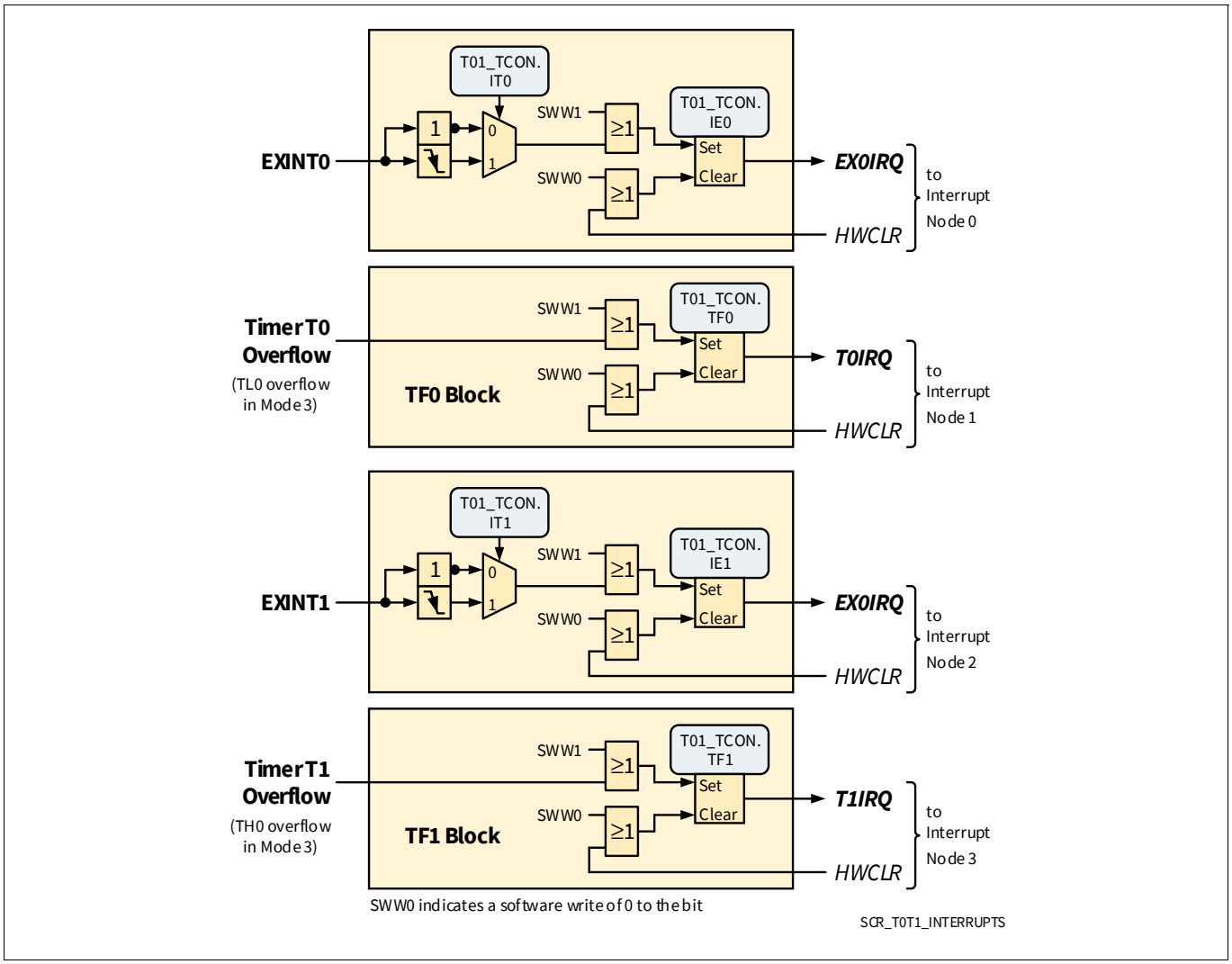


Figure 854 T0/T1 Interrupt Requests

49.12.6 Register Map

Seven SFRs control the operations of Timer 0 and Timer 1. They can be accessed from both the standard (non-mapped) and mapped SFR area.

Table 669 lists the addresses of these SFRs.

Table 669 Register Overview - T0 / T1 (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
IEN0	Interrupt Enable Register 0	X	X	$0D8_H$	195
T01_TCON	Timer 0/1 Control Register	X	X	$0C9_H$	192
T01_TH0	Timer 0 High Byte	X	X	$0CD_H$	191
T01_TH1	Timer 1 High Byte	X	X	$0CE_H$	192
T01_TL0	Timer 0 Low Byte	X	X	$0CB_H$	190
T01_TL1	Timer 1 Low Byte	X	X	$0CC_H$	190
T01_TMOD	Timer Mode Register	X	X	$0CA_H$	193

49.12.7 Register Description

The low bytes (TL0, TL1) and high bytes (TH0, TH1) of both Timer 0 and Timer 1 can be combined to a one-timer configuration depending on the mode used. Register T01_TCON controls the operations of Timer 0 and Timer 1. The operating modes of both timers are selected using register T01_TMOD. Register IEN0 contains bits that enable interrupt operations in Timer 0 and Timer 1.

Timer 0 Low Byte

T01_TL0
Timer 0 Low Byte **($0CB_H$)** **Reset Value: Table 670**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
VAL							
rwh							

Field	Bits	Type	Description
VAL	7:0	rwh	**Timer 0 Low Byte Value** *Note: If TMOD.T0M is equal to:* *00_B: TL0 holds the 5-bit prescaler value.* *01_B: TL0 holds the lower 8-bit part of the 16-bit timer value.* *10_B: TL0 holds the 8-bit timer value.* *11_B: TL0 holds the 8-bit timer value.*

Table 670 Reset Values of T01_TL0

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

Timer 1 Low Byte

T01_TL1
Timer 1 Low Byte **($0CC_H$)** **Reset Value: Table 671**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
VAL							
rwh							

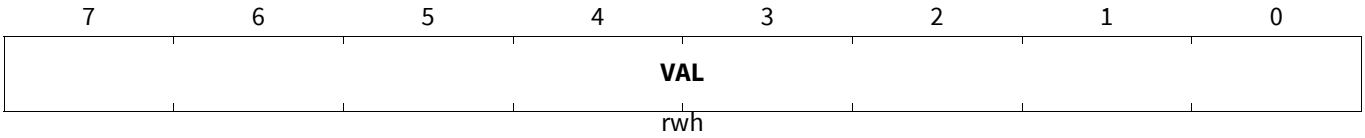
Field	Bits	Type	Description
VAL	7:0	rwh	**Timer 1 Low Byte Value** *Note: If TMOD.T1M is equal to:* *00_B: TL1 holds the 5-bit prescaler value.* *01_B: TL1 holds the lower 8-bit part of the 16-bit timer value.* *10_B: TL1 holds the 8-bit timer value.* *11_B: TL1 is not used.*

Table 671 Reset Values of T01_TL1

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

Timer 0 High Byte

T01_TH0
Timer 0 High Byte **($0CD_H$)** **Reset Value: Table 672**
RMAP: X, PAGE: X



Field	Bits	Type	Description
VAL	7:0	rwh	**Timer 0 High Byte Value** *Note: If TMOD.T0M is equal to:* *00_B: TH0 holds the 8-bit prescaler value.* *01_B: TH0 holds the higher 8-bit part of the 16-bit timer value.* *10_B: TH0 holds the 8-bit reload value.* *11_B: TH0 holds the 8-bit timer value.*

Table 672 Reset Values of T01_TH0

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

Timer 1 High Byte

T01_TH1
Timer 1 High Byte **($0CE_H$)** **Reset Value: Table 673**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
VAL							
rwh							

Field	Bits	Type	Description
VAL	7:0	rwh	**Timer 1 High Byte Value** *Note: If TMOD.T1M is equal to:* *00_B: TH1 holds the 8-bit prescaler value.* *01_B: TH1 holds the higher 8-bit part of the 16-bit timer value.* *10_B: TH1 holds the 8-bit reload value.* *11_B: TH1 is not used.*

Table 673 Reset Values of T01_TH1

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

Timer 0/1 Control Register

T01_TCON
Timer 0/1 Control Register **($0C9_H$)** **Reset Value: Table 674**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**
rwh	rw	rwh	rw	rwh	rw	rwh	rw

Field	Bits	Type	Description
IT0	0	rw	**External Interrupt 0 Level/Edge Trigger Control** 0_B Low level triggered external interrupt 0 is selected 1_B Falling edge triggered external interrupt 0 is selected
IE0	1	rwh	**External Interrupt 0 Flag** Set by hardware when external interrupt 0 event is detected. Cleared by hardware when the processor vectors to interrupt routine. Can also be set/cleared by software.

Field	Bits	Type	Description
IT1	2	rw	**External Interrupt 1 Level/Edge Trigger Control** 0_B Low level triggered external interrupt 1 is selected 1_B Falling edge triggered external interrupt 1 is selected
IE1	3	rwh	**External Interrupt 1 Flag** Set by hardware when external interrupt 1 event is detected. Cleared by hardware when the processor vectors to interrupt routine. Can also be set/cleared by software.
TR0	4	rw	**Timer 0 Run Control** 0_B Timer is halted 1_B Timer runs
TF0	5	rwh	**Timer 0 Overflow Flag** Set by hardware on Timer/Counter 0 overflow. Cleared by hardware when processor vectors to interrupt routine. Can also be set/cleared by software.
TR1	6	rw	**Timer 1 Run Control** *Note: Timer 1 Run Control also affects TH0, if Timer 0 operates in Mode 3.* 0_B Timer is halted 1_B Timer runs
TF1	7	rwh	**Timer 1 Overflow Flag** Set by hardware on Timer/Counter 1 overflow. Cleared by hardware when processor vectors to interrupt routine. Can also be set/cleared by software. *Note: TF1 is set by TH0 instead, if Timer 0 operates in Mode 3.*

Table 674 Reset Values of T01_TCON

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

Timer Mode Register

T01_TMOD
Timer Mode Register **($0CA_H$)** **Reset Value: Table 675**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
GATE1	**T1S**	**T1M**		**GATE0**	**T0S**	**T0M**	
rw	rw	rw		rw	rw	rw	

Field	Bits	Type	Description
T0M	1:0	rw	**Timer 0 Mode Select Bits** 00_B **TIMER_13B**, 13-bit timer (M8048 compatible mode) 01_B **TIMER_16B**, 16-bit timer 10_B **TIMER_8B_AUTO_REL**, 8-bit auto-reload timer 11_B **TIMER0_SPLIT**, Timer 0 is split into two halves. TL0 is an 8-bit timer controlled by the standard Timer 0 control bits, and TH0 is the other 8-bit timer controlled by the standard Timer 1 control bits. TH1 and TL1 of Timer 1 are held (Timer 1 is stopped).
T0S	2	rw	**Timer 0 Input Selector** 0_B Input is from internal system clock 1_B Input is from T0 pin
GATE0	3	rw	**Timer 0 Gate Control** 0_B Timer 0 will only run if TCON.TR0 = 1 (software control) 1_B Timer 0 will only run if EXINT0 pin = 1 (hardware control) and TCON.TR0 is set
T1M	5:4	rw	**Timer 1 Mode Select Bits** 00_B **TIMER_13B**, 13-bit timer (M8048 compatible mode) 01_B **TIMER_16B**, 16-bit timer 10_B **TIMER_8B_AUTO_REL**, 8-bit auto-reload timer 11_B **TIMER0_SPLIT**, Timer 0 is split into two halves. TL0 is an 8-bit timer controlled by the standard Timer 0 control bits, and TH0 is the other 8-bit timer controlled by the standard Timer 1 control bits. TH1 and TL1 of Timer 1 are held (Timer 1 is stopped).
T1S	6	rw	**Timer 1 Input Selector** 0_B Input is from internal system clock 1_B Input is from T1 pin
GATE1	7	rw	**Timer 1 Gate Control** 0_B Timer 1 will only run if TCON.TR1 = 1 (software control) 1_B Timer 1 will only run if EXINT1 pin = 1 (hardware control) and TCON.TR1 is set

Table 675 Reset Values of T01_TMOD

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

Interrupt Enable Register 0

IEN0
Interrupt Enable Register 0 **(0D8$_H$)** **Reset Value: Table 676**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
EA	**0**	**ET2**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**
rw	r	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
EX0	0	rw	**Interrupt Node XINTR0 Enable** 0_B XINTR0 is disabled 1_B XINTR0 is enabled
ET0	1	rw	**Interrupt Node XINTR1 Enable** 0_B XINTR1 is disabled 1_B XINTR1 is enabled
EX1	2	rw	**Interrupt Node XINTR2 Enable** 0_B XINTR2 is disabled 1_B XINTR2 is enabled
ET1	3	rw	**Interrupt Node XINTR3 Enable** 0_B XINTR3 is disabled 1_B XINTR3 is enabled
ES	4	rw	**Interrupt Node XINTR4 Enable** 0_B XINTR4 is disabled 1_B XINTR4 is enabled
ET2	5	rw	**Interrupt Node XINTR5 Enable** 0_B XINTR5 is disabled 1_B XINTR5 is enabled
EA	7	rw	**Global Interrupt Mask** 0_B All pending interrupt requests (except NMI) are blocked from the core 1_B Pending interrupt requests are not blocked from the core
0	6	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 676 Reset Values of IEN0

Reset Type	Reset Value	Note
Generated Reset	0X00 0000_B	
LVD Reset	0–00 0000_B	

49.12.8 Revision History

Table 677 Revision History

Reference	Change to Previous Version	Change Request
v2.5		
	First Official Release of completely reworked SCR chapter	
	No change	

49.13 T2CCU Module

The T2CCU module is a flexible timing unit with two 16-bit timers and a 6-channel Capture/Compare Unit (CCU).

49.13.1 Overview

The block diagram of the T2CCU module is shown in **Figure 855**. It consists of the Timer 2, the Capture/Compare Timer (CCT), and a Capture/Compare Unit (CCU). Control is available in the CCU, for each of its 16-bit capture/compare channels, to individually select either Timer 2 or the Capture/Compare Timer (CCT) as the time base. Both timers have a resolution of 16 bits. The clock frequency of T2CCU, f_{T2CCU}, is set at PCLK frequency.

The T2CCU can be used for various digital signal generation and event capturing like pulse generation, pulse width modulation, pulse width measuring etc. Target applications include various automotive control as well as industrial applications (frequency generation, digital-to-analog conversion, process control etc.).

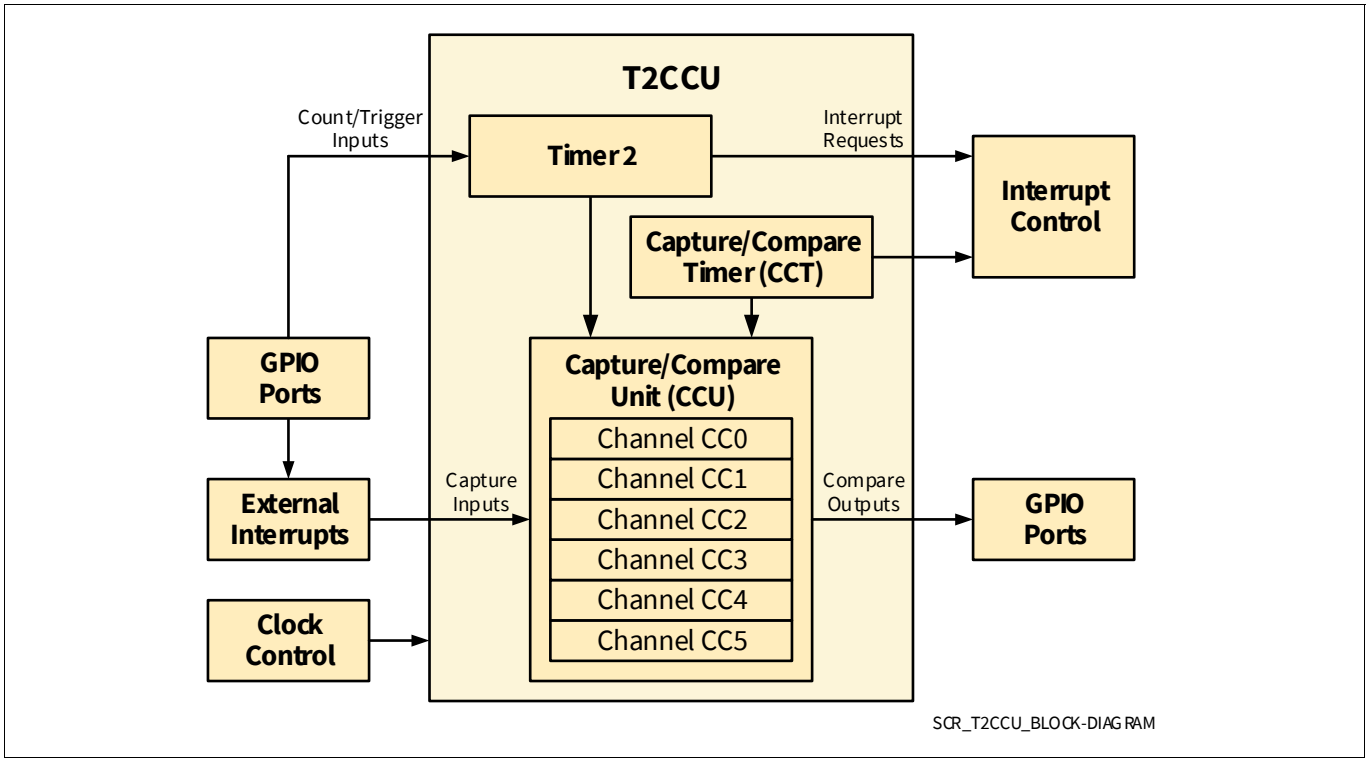


Figure 855 T2CCU Module Block Diagram

The T2CCU has the following features:

Timer 2

- 16-bit resolution
- Timer or counter operation
- 16-bit auto-reload mode
- Selectable up or down counting
- One channel, 16-bit capture mode
- Baudrate generator for U(S)ART
- Usable as base timer for the CCU

Capture/Compare Timer (CCT)

- 16-bit resolution
- Timer operation
- 16-bit reload mode
- Extremely flexible Capture/Compare Timer count rate by cascading with Timer 2
- Capture/Compare Timer may be ‘reset’ immediately by triggering overflow event

Capture/Compare Unit (CCU)

- 16-bit resolution
- Option to individually select either Timer 2 or the CCT as base timer for a channel
- Six compare channels in total
  - Shadow register for each compare register: Transfer via software control or on timer overflow
  - Compare Mode 0: Compare output signal changes from the inactive level to active level on compare match. Returns to inactive level on timer overflow.
    Compare Mode 0 can be configured with or without dead-time generation.
    Active level can be defined by register bit for channel groups A and B.
    Support of 0% to 100% duty cycle in compare mode 0.
  - Compare Mode 1: Full control of the software on the compare output signal level, for the next compare match.
  - Concurrent Compare Mode with channel 0
- Four capture channels, shared with the compare channels
  - Capture Mode 0: Capture on any external event (rising/falling/both edge) at the 4 pins T2CC0 to T2CC3.
  - Capture Mode 1: Capture upon writing to the low byte of the corresponding channel capture register.
  - Capture mode 0 or 1 can be established independently on the 4 capture channels.

49.13.2 Timer 2

The Timer 2 of the T2CCU has the following features:

- 16-bit auto-reload mode
- selectable up or down counting
- One channel, 16-bit capture mode
- Timer 2 can be used as a stand-alone timer, aside of the CCU, or can be used as a base timer for the CCU

49.13.2.1 Basic Timer 2 Operations

The following sections describe the basic functions of Timer 2.

49.13.2.1.1 Timer 2 Start/Stop and Count Control

The count input to Timer 2 is controlled by the run bit TR2, located in register T2_CON. Timer 2 can be started by setting run bit TR2 either through hardware or software. The timer can only be stopped via software by clearing bit TR2.

The run bit TR2 can be set by various events:

- By software: Write ’1’ to bit T2CON.TR2.
- By hardware:
  - A selectable transition at input T2EX. This input is enabled/disabled via bit T2_MOD.T2RHEN. The active transition of the input, either a rising or a falling edge, is selected via bit T2_MOD.REGS.
    Please note that input T2EX can also be used for the capture/reload control of Timer 2 (see following sections).
  - Via the synchronous start signal from the capture/compare timer CCT. This event is enabled/disabled via bit TIMSYN in register T2CCU_CCTCON. This option can be used to trigger a synchronous start of both timers, timer 2 and timer CCT. See **Synchronized Start of Timer 2 and CCT Timer** for details on this option.

All of these actions set the run bit TR2 to 1. Its output controls the count input to the Timer 2. The state of the run bit can be read by software. To stop the timer, clear the run bit TR2 by writing a ‘0’ to it.

Figure 856 illustrates the start/stop and count control of timer 2. The count input selection is described in the next section. In the further diagrams of Timer 2, the start/stop control is shown as a block, together with the count input control. For the details, please refer to **Figure 856**.

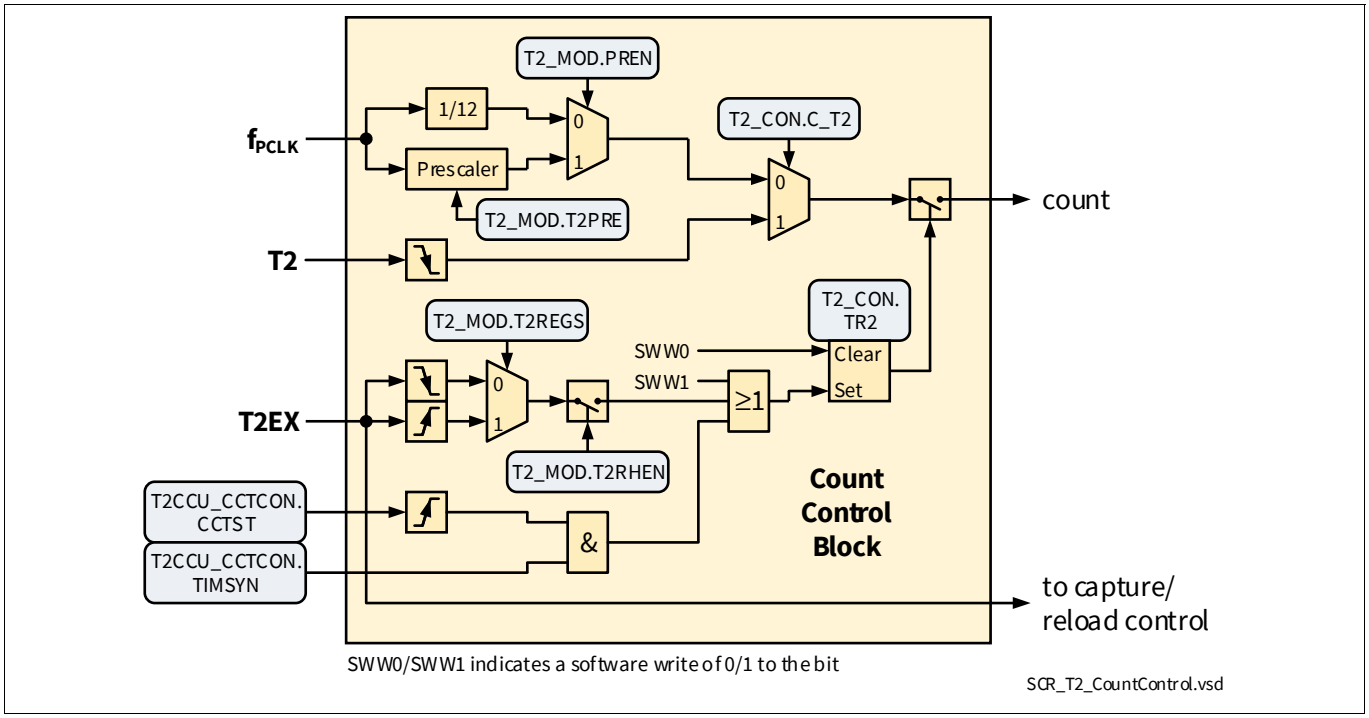


Figure 856 Timer 2 Start/Stop and Count Control

49.13.2.1.2 Count Clock Options

Timer 2 can either operate as a timer or as a counter. The selection is made via bit C_T2 in register T2_CON.

- Timer mode (C_T2 = 0):
  Either $f_{PCLK}/12$ or f_{PCLK} divided by a programmable prescaler can be selected as input clock to Timer 2. The selection is made via bit PREN in register T2_MOD.
  The divide factor for the programmable prescaler is chosen via bitfield T2_MOD.T2PRE.
- Counter mode ((C_T2 = 1):
  Timer 2 counts in response to a 1-to-0 transition at input T2. Input T2 is sampled over 2 PCLK clock cycles. If a 1 was detected during the first clock and a 0 was detected in the following clock, then a count clock for Timer 2 is generated. Therefore, the input levels should be stable for at least 1 PCLK clock cycle.

49.13.2.2 Timer 2 Operating Modes

The following sections describe the various operating modes of Timer 2.

Note: In order to simplify the diagrams, the connection of the Timer 2 contents to the Capture/Compare Unit (CCU) is not shown in the following figures.

49.13.2.2.1 Reload Mode of Timer 2

In reload mode of Timer 2 (bit CP_RL2 in register T2_CON is 0), the RC2 register (register pair T2_RC2H | T2_RC2L) is used to hold a reload/trigger value for Timer 2:

- When down-counting is disabled, the value in RC2 is loaded into Timer 2 either on overflow or in response to an external signal transition.
- When down-counting is enabled, the value in RC2 is used as a trigger value to load $FFFF_H$ into Timer 2.

In the following sections, these operations are described in detail.

*Note: For a description of the interrupt request and flag logic "TF2 Block" and "EXF2 Block", please see **Section 49.13.2.3, Timer 2 Interrupt Requests**.*

Reload Operation in Up-Counting Mode

When bit DCEN in register T2_MOD is 0, Timer 2 can only count up, and the RC2 register is used to hold a reload value. **Figure 857** shows an operational block diagram of Timer 2 in this configuration.

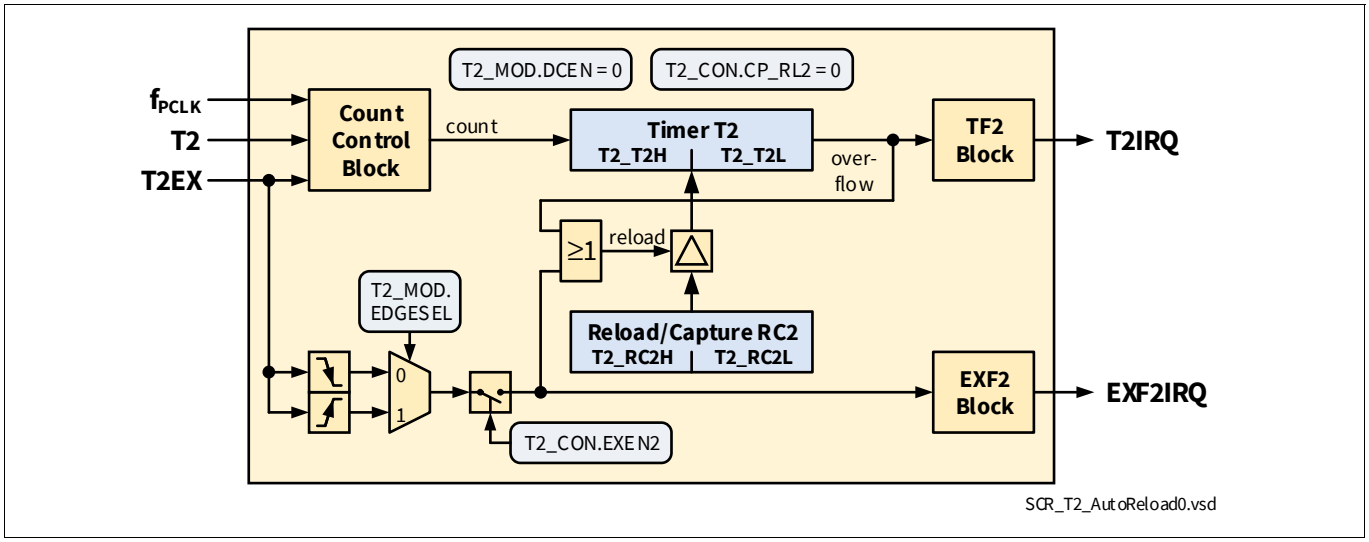


Figure 857 Reload Operation in Up-Counting Mode (T2_MOD.DCEN = 0)

Depending on bit EXEN2 in register T2_CON, two options exist:

- EXEN2 = 0: Reload on overflow only (auto-reload):
  - Once the timer is started, it counts up to its maximum value $FFFF_H$.
  - The next count clock pulse leads to an overflow condition: Timer 2 is loaded with the contents of register RC2, and flag TF2 in register T2_CON is set to 1 (can be used to generate an interrupt request).
  - With the next count clock pulse(s), the timer continues to count up from the reloaded value.
- EXEN2 = 1: Reload on overflow or on external signal edge:
  - Once the timer is started, it counts up.
  - A reload of Timer 2 is performed either on an overflow condition (auto-reload, see above) or by a selected signal edge on input T2EX (external reload). With the next count clock pulse(s), the timer continues to count up from the reloaded value.
  - The signal edge can either be a positive (0-to-1), or a negative (1-to-0) transition at input T2EX. The selection is done via bit EDGESEL in register T2_MOD.
  - The selected signal transition also sets flag EXF2 in register T2_CON to 1 (can be used to generate an interrupt request).

Note that the input signal T2EX can be used for both, the timer count control and the external reload function.

When bit T2RHEN is set, Timer 2 is started by first falling edge/rising edge at pin T2EX, which is defined by bit T2REGS. If bit EXEN2 is set, bit EXF2 is also set at the same point when Timer 2 is started with the same falling edge/rising edge at pin T2EX, which is defined by bit EDGESEL. The reload will happen with the following negative/positive transitions at pin T2EX, which is defined by bit EDGESEL.

Note: In counter mode, if the reload via T2EX and the count clock T2 are detected simultaneously, the reload takes precedence over the count. The counter increments its value with the following T2 count clock.

Reload Operation in Up/Down-Counting Mode

When bit DCEN in register T2_MOD is set to 1, the up/down count selection is enabled for Timer 2. The count direction is determined by the level at input pin T2EX. **Figure 858** shows an operational block diagram of Timer 2 in this configuration.

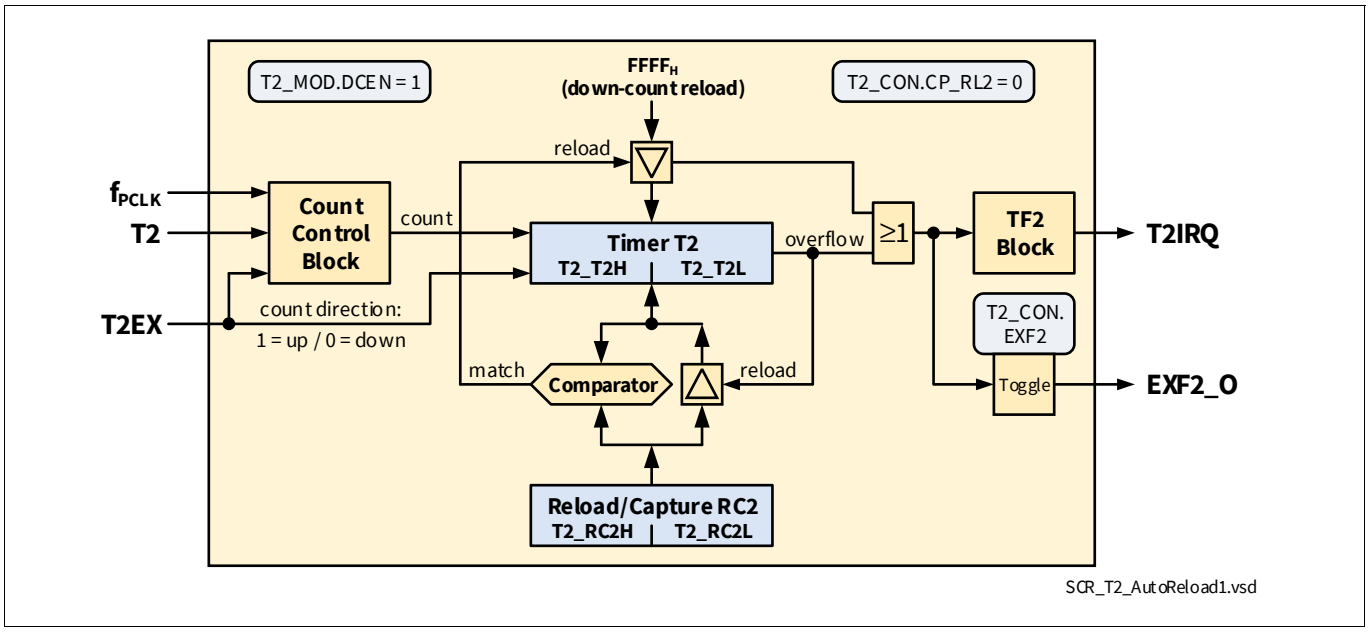


Figure 858 Reload Operation in Up/Down-Counting Mode (T2_MOD.DCEN = 1)

The operation in this mode is as follows:

- A logic 1 at pin T2EX sets the Timer 2 to up-counting mode:
  - Once the timer is started, it counts up to its maximum value FFFF$_H$.
  - The next count clock pulse leads to an overflow condition: Timer 2 is loaded with the contents of register RC2, and flag TF2 in register T2_CON is set to 1 (can be used to generate an interrupt request).
  - With the next count clock pulse(s), the timer continues to count up from the reloaded value.
- A logic 0 at pin T2EX sets the Timer 2 to down-counting mode:
  - Once the timer is started, it counts down.
  - When the timer contents match the value stored in register RC2, an “underflow” condition is generated. Timer 2 is loaded with the value FFFF$_H$, and flag TF2 in register T2_CON is set to 1 (can be used to generate an interrupt request).
  - With the next count clock pulse(s), the timer continues to count down from FFFF$_H$.

If bit T2RHEN is set, Timer 2 can be started either by rising edge (T2REGS = 1) at pin T2EX and then proceed with the up counting, or be started by falling edge (T2REGS = 0) at pin T2EX and then proceed with the down counting.

In this mode, bit EXF2 toggles whenever an overflow or an underflow condition is detected. This flag, however, does not generate an interrupt request.

49.13.2.2.2 Capture Mode of Timer 2

In capture mode of Timer 2 (bit T2_CON.CP_RL2 is 1), the RC2 register (register pair T2_RC2H | T2_RC2L) is used to latch the current contents of the timer in response to an external event.

Figure 859 shows an operational block diagram of Timer 2 in this configuration.

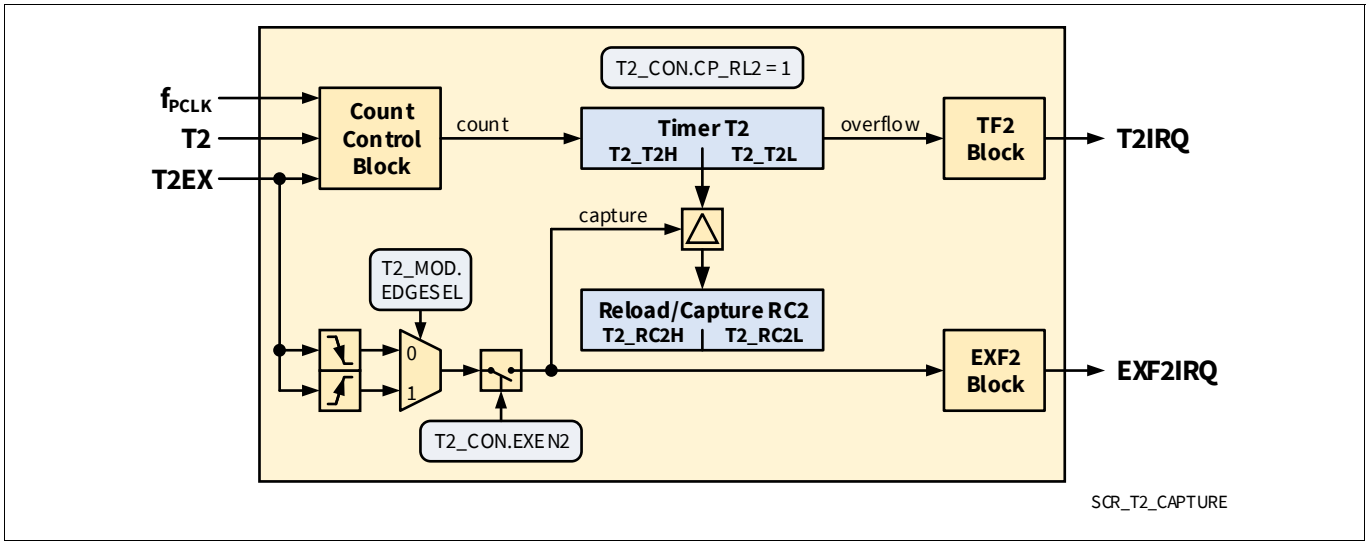


Figure 859 Capture Mode of Timer 2

To operate the timer in capture mode, the following settings must be made:

- Enable capture mode: Set bit CP_RL2 in register T2_CON to 1.
- Disable the down-count function: Clear bit DCEN in register T2_MOD to 0.
- Enable the external input T2EX: Set bit EXEN2 in register T2_CON to 1.

The timer functions as a 16-bit timer and always counts up to $FFFF_H$. With the next clock pulse, an overflow condition occurs; bit TF2 is set (which can generate an interrupt request), and the timer registers are loaded with 0000_H. The counting continues with the next clock pulse(s).

With a falling or rising edge (chosen by T2_MOD.EDGESEL) at input T2EX, the contents of the timer register are captured into the RC2 register. At the same time, bit EXF2 in register T2_CON is set to 1.
The external input is sampled in every PCLK clock cycle. When a sample shows a low (high) level in one PCLK clock cycle, and a high (low) in the next PCLK clock cycle, a transition is recognized. If the capture signal is detected while the timer is being incremented, the timer is first incremented before the capture operation is performed. This ensures that always the latest value of the timer register is captured.

Note that the input signal T2EX can be used for both, the timer count control and the external capture function.

If bit T2RHEN is set, Timer 2 is started by first falling edge/rising edge at pin T2EX, which is defined by bit T2REGS. If bit EXEN2 is set, bit EXF2 is also set at the same point when Timer 2 is started with the same falling edge/rising edge at pin T2EX, which is defined by bit EDGESEL. The capture will happen with the following negative/positive transitions at pin T2EX, which is defined by bit EDGESEL.

49.13.2.3 Timer 2 Interrupt Requests

Timer 2 can generate two different interrupt requests to the CPU:

- An interrupt request on timer overflow, and
- an interrupt request on an external transition at input T2EX.

These interrupt requests are detailed in the following sections.

49.13.2.3.1 Timer Overflow Interrupt Request

Figure 860 shows a functional diagram of the overflow interrupt request generation. The flag TF2 in register T2_CON is set on an overflow of the timer (in certain modes), or it can be set by software. When enabled via bit TF2EN in register T2_CON1, the signal which sets the TF2 flag will generate an interrupt request to the CPU.

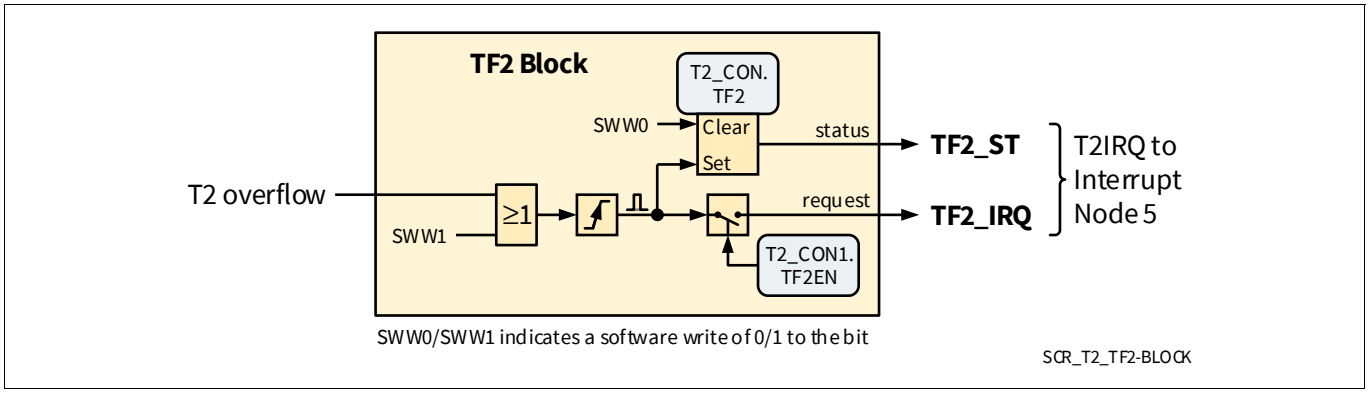


Figure 860 TF2 Flag Block

The overflow flag and the corresponding overflow interrupt request (if enabled) will be set/generated in the following cases:

- By software:
  By writing a 1 to flag T2_CON.TF2, the flag will be set and an interrupt request (if enabled) will be generated.
- Reload Operation in Up-Counting Mode (T2_MOD.DCEN = 0, T2_CON.CP_RL2 = 0):
  When an overflow of Timer 2 occurs.
- Reload Operation in Up/Down-Counting Mode (T2_MOD.DCEN = 1, T2_CON.CP_RL2 = 0):
  When an overflow of Timer 2 occurs, or when an “underflow” (timer contents match the value stored in register RC2) occurs.
- Capture Mode of Timer 2 (T2_MOD.DCEN = 0, T2_CON.CP_RL2 = 1):
  When an overflow of Timer 2 occurs.

Flag TF2 must be cleared by software.

Note: The overflow interrupt request logic of Timer 2 is basically of type “interrupt structure 2” (see section “Interrupt Structure” in chapter “Interrupt System”), however, the status signal does not have an individual enable control. Thus, in order to clear the pending request flag in the interrupt system, it is necessary to clear all status flags assigned to that interrupt node, even if the respective interrupt request is not enabled.

Note: Although two signals, TF2_IRQ and TF2_ST are connected to the respective interrupt node, the general signal name T2IRQ is used to represent both signals (interrupt structure 2).

49.13.2.3.2 Timer External Interrupt Request

Figure 861 shows a functional diagram of the external interrupt request generation. The flag EXF2 in register T2_CON is set on an external transition at input T2EX (in certain modes), or it can be set by software. When enabled via bit EXF2EN in register T2_CON1, the signal which sets the EXF2 flag will generate an interrupt request to the CPU.

The output of flag EXF2 is also connected to certain GPIO port lines.

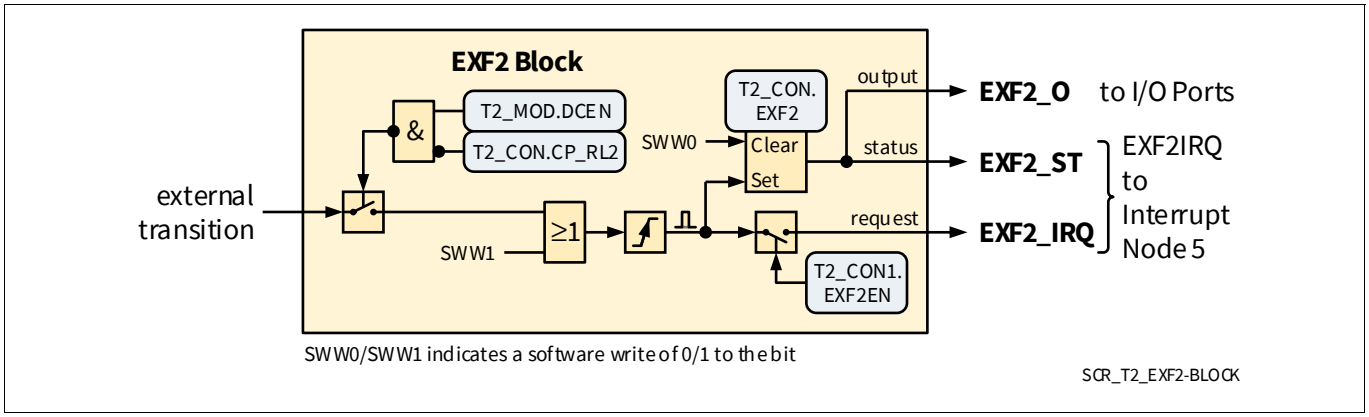


Figure 861 EXF2 Flag Block

The external transition flag EXF2 and the corresponding external interrupt request (if enabled) will be set/generated in the following cases:

- By software:
  By writing a 1 to flag T2_CON.EXF2, the flag will be set and an interrupt request (if enabled) will be generated.
- Reload Operation in Up-Counting Mode (T2_MOD.DCEN = 0, T2_CON.CP_RL2 = 0):
  When bit T2_CON.EXEN2 is set and the selected transition at input T2EX occurs.
- Reload Operation in Up/Down-Counting Mode (T2_MOD.DCEN = 1, T2_CON.CP_RL2 = 0):
  In this mode, flag EXF2 will toggle on each overflow or “underflow” (timer contents match the value stored in register RC2) of Timer 2. No interrupt request will be generated (this is not shown in **Figure 861**).
- Capture Mode of Timer 2 (T2_MOD.DCEN = 0, T2_CON.CP_RL2 = 1):
  When bit T2_CON.EXEN2 is set and the selected transition at input T2EX occurs.

Flag EXF2 must be cleared by software.

Note that while the timer/counter function is disabled (TR2 = 0), it is still possible to generate a Timer 2 interrupt to the core via an external event at input T2EX, as long as Timer 2 remains enabled (PMCON1.T2CCU_DIS = 0). To achieve this, bit EXEN2 in register T2_CON must be set. As a result, any transition on T2EX will cause either a dummy reload or a dummy capture, depending on the CP_RL2 bit selection.

Note: The external interrupt request logic of Timer 2 is basically of type “interrupt structure 2” (see section “Interrupt Structure” in chapter “Interrupt System”), however, the status signal does not have an individual enable control. Thus, in order to clear the pending request flag in the interrupt system, it is necessary to clear all status flags assigned to that interrupt node, even if the respective interrupt request is not enabled.

Note: Although two signals, EXF2_IRQ and EXF2_ST are connected to the respective interrupt node, the general signal name EXF2IRQ is used to represent both signals (interrupt structure 2).

49.13.2.4 Input Selection for Timer 2

Timer 2 can be used as an event counter which counts 1-to-0 transitions at the external input T2. This input is selected from three different sources, T2A, T2B and T2C. The selection is performed by the SFR bits MODPISEL3.T2IS. In addition, there are eight sources for the input T2EX for Timer 2. They can be selected by MODPISEL3.T2EXIS.

The pin assignment for the T2CCU is shown in the GPIO chapter of the SCR.

The bitfield PAGE of SCU_PAGE register must be programmed before accessing the MODPISEL3 register.

Peripheral Input Select Register 3

The MODPISEL3 register is used for input pin selection of the T2CCU and SSC modules. The register is on SCU_PAGE. The connections of Timer 2 T2EXF and T2EXG are available via PMS.

MODPISEL3
Peripheral Input Select Register 3 **($0F5_H$)** **Reset Value: Table 678**
RMAP: 0, PAGE: SCU_PAGE=2

7	6	5	4	3	2	1	0
CIS			**T2IS**		**T2EXIS**		
rw			rw		rw		

Field	Bits	Type	Description
T2EXIS	2:0	rw	**Timer 2 External Input Select** 000_B **T2EXA**, Timer 2 Input T2EXA is selected 001_B **T2EXB**, Timer 2 Input T2EXB is selected 010_B **T2EXC**, Timer 2 Input T2EXC is selected 011_B **T2EXD**, Timer 2 Input T2EXD is selected 100_B **T2EXE**, Timer 2 Input T2EXE is selected 101_B **T2EXF**, Timer 2 Input T2EXF is selected 110_B **T2EXG**, Timer 2 Input T2EXG is selected 111_B **T2EXH**, Timer 2 Input T2EXH is selected
T2IS	4:3	rw	**Timer 2 Input Select** 00_B **T2A**, Timer 2 Input T2A is selected 01_B **T2B**, Timer 2 Input T2B is selected 10_B **T2C**, Timer 2 Input T2C is selected 11_B Reserved
CIS	7:5	rw	**Slave Mode Clock Input Select** 000_B **SCLKA**, SSC Slave Clock Input SCLKA is selected 001_B **SCLKB**, SSC Slave Clock Input SCLKB is selected 010_B **SCLKC**, SSC Slave Clock Input SCLKC is selected **others**, Reserved

Table 678 Reset Values of MODPISEL3

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.13.2.5 Timer 2 Register Map

All Timer 2 register names described in the following sections are referenced in other chapters of this document with the module name prefix “T2_”, e.g., T2_CON.

The Timer 2 SFRs are located in the standard (non-mapped) SFR area. **Table 679** lists these addresses.

Table 679 Register Overview - Timer 2 (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
T2_CON	Timer 2 Control Register	0	0	$0C0_H$	208
T2_CON1	Timer 2 Control Register 1	0	0	$0C6_H$	209
T2_MOD	Timer 2 Mode Register	0	0	$0C1_H$	207
T2_RC2H	Timer 2 Reload/Capture Register, High Byte	0	0	$0C3_H$	211
T2_RC2L	Timer 2 Reload/Capture Register, Low Byte	0	0	$0C2_H$	210
T2_T2H	Timer 2, High Byte	0	0	$0C5_H$	212
T2_T2L	Timer 2, Low Byte	0	0	$0C4_H$	211

49.13.2.6 Register Description

49.13.2.6.1 Mode Register

The T2_MOD (T2CCU_PAGE) is used to configure Timer 2 for various modes of operation.

Timer 2 Mode Register

T2_MOD
Timer 2 Mode Register **($0C1_H$)** **Reset Value: Table 680**
RMAP: 0, PAGE: T2CCU_PAGE=0

7	6	5	4	3	2	1	0
T2REGS	**T2RHEN**	**EDGESEL**	**PREN**		**T2PRE**		**DCEN**
rw	rw	rw	rw		rw		rw

Field	Bits	Type	Description
DCEN	0	rw	**Up/Down Counter Enable** 0_B Up/Down Counter function is disabled 1_B Up/Down Counter function is enabled and controlled by pin T2EX (Up = 1, Down = 0)

Field	Bits	Type	Description
T2PRE	3:1	rw	**Timer 2 Prescaler Bit** Selects the input clock for Timer 2 which is derived from the peripheral clock. 000_B **PCLK**, fT2 = fPCLK 001_B **PCLK_DIV2**, fT2 = fPCLK / 2 010_B **PCLK_DIV4**, fT2 = fPCLK / 4 011_B **PCLK_DIV8**, fT2 = fPCLK / 8 100_B **PCLK_DIV16**, fT2 = fPCLK / 16 101_B **PCLK_DIV32**, fT2 = fPCLK / 32 110_B **PCLK_DIV64**, fT2 = fPCLK / 64 111_B **PCLK_DIV128**, fT2 = fPCLK / 128
PREN	4	rw	**Prescaler Enable** 0_B Prescaler is disabled, and the 2 or 12 divider takes effect 1_B Prescaler is enabled (see T2PRE bit), and the 2 or 12 divider is bypassed
EDGESEL	5	rw	**Edge Select in Capture Mode/Reload Mode** 0_B The falling edge at pin T2EX is selected 1_B The rising edge at pin T2EX is selected
T2RHEN	6	rw	**Timer 2 External Start Enable** 0_B Timer 2 External Start is disabled 1_B Timer 2 External Start is enabled
T2REGS	7	rw	**Edge Select for Timer 2 External Start** 0_B The falling edge at pin T2EX is selected 1_B The rising edge at Pin T2EX is selected

Table 680 Reset Values of T2_MOD

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.13.2.6.2 Control Register

Control register T2_CON (T2CCU_PAGE) is used to control the operating modes and interrupt of Timer 2. In addition, it contains the status flags for interrupt generation.

Timer 2 Control Register

T2_CON
Timer 2 Control Register **($0C0_H$)** **Reset Value: Table 681**
RMAP: 0, PAGE: T2CCU_PAGE=0

7	6	5	4	3	2	1	0
TF2	**EXF2**	**0**	**0**	**EXEN2**	**TR2**	**C_T2**	**CP_RL2**
rwh	rwh	rw	rw	rw	rwh	rw	rw

Field	Bits	Type	Description
CP_RL2	0	rw	**Capture/Reload Select** 0_B Reload upon overflow or upon negative/positive transition at pin T2EX (when EXEN2 = 1) 1_B Capture Timer 2 data register contents on the negative/positive transition at pin T2EX, provided EXEN2 = 1. The negative or positive transition at pin T2EX is selected by bit EDGESEL in register T2MOD.
C_T2	1	rw	**Timer or Counter Select** 0_B Timer function selected 1_B Count upon negative edge at pin T2
TR2	2	rwh	**Timer 2 Run Control** This bit can be set or cleared by software to start or stop the timer T2. It can also be set by hardware via input T2EX, if bit T2MOD.T2RHEN is set to 1, or via a synchronized start with the CCT timer of the T2CCU. *Note: Writing a one to this bit while the timer is running has no effect.* 0_B Timer T2 is halted 1_B Timer T2 is running
EXEN2	3	rw	**Timer 2 External Enable Control** 0_B External events are disabled 1_B External events are enabled in Capture/Reload Mode
EXF2	6	rwh	**Timer 2 External Flag** In Capture/Reload Mode, this bit is set by hardware when a negative/positive transition occurs at pin T2EX, if bit EXEN2 = 1. This bit must be cleared by software. *Note: When bit DCEN = 1 in auto-reload mode, no interrupt request to the core is generated.*
TF2	7	rwh	**Timer 2 Overflow/Underflow Flag** Set by a Timer 2 overflow/underflow. Must be cleared by software.
0	5:4	rw	**Reserved** Shall be written with 0.

Table 681 Reset Values of T2_CON

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Timer 2 Control Register 1

Register T2_CON1 (T2CCU_PAGE) is used to enable the external interrupt and the overflow interrupt.

T2_CON1
Timer 2 Control Register 1 **($0C6_H$)** **Reset Value: Table 682**
RMAP: 0, PAGE: T2CCU_PAGE=0

7	6	5	4	3	2	1	0
0						TF2EN	EXF2EN
r						rw	rw

Field	Bits	Type	Description
EXF2EN	0	rw	**External Interrupt Enable** 0_B External interrupt is disabled 1_B External interrupt is enabled
TF2EN	1	rw	**Overflow/Underflow Interrupt Enable** 0_B Overflow/underflow interrupt is disabled 1_B Overflow/underflow interrupt is enabled
0	7:2	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 682 Reset Values of T2_CON1

Reset Type	Reset Value	Note
LVD Reset	XXXX $XX11_B$	
Generated Reset	---- $--11_B$	

49.13.2.6.3 Timer 2 Reload/Capture Register

The T2_RC2 (T2CCU_PAGE) register is used for a 16-bit reload of the timer count upon overflow or a capture of current timer count depending on the mode selected. This register is divided into two 8-bit registers.

Timer 2 Reload/Capture Register, Low Byte

T2_RC2L
Timer 2 Reload/Capture Register, Low Byte **($0C2_H$)** **Reset Value: Table 683**
RMAP: 0, PAGE: T2CCU_PAGE=0

7	6	5	4	3	2	1	0
RC2							
rwh							

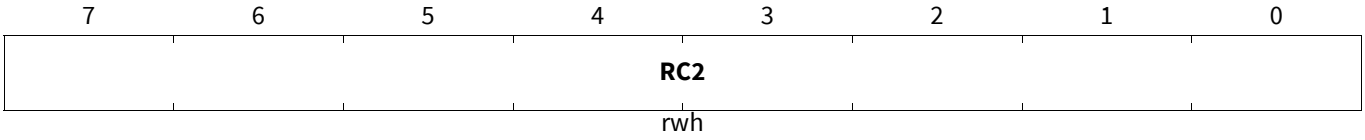
Field	Bits	Type	Description
RC2	7:0	rwh	**Reload/Capture Value [7:0]** If T2_CON.CP_RL2 = 0, these contents are loaded into the timer register upon an overflow condition. If T2_CON.CP_RL2 = 1, this register is loaded with the current timer count upon a negative/positive transition at pin T2EX when EXEN2 = 1.

Table 683 Reset Values of T2_RC2L

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Timer 2 Reload/Capture Register, High Byte

T2_RC2H
Timer 2 Reload/Capture Register, High Byte ($0C3_H$) Reset Value: Table 684
RMAP: 0, PAGE: T2CCU_PAGE=0



Field	Bits	Type	Description
RC2	7:0	rwh	**Reload/Capture Value [15:8]** If T2_CON.CP_RL2 = 0, these contents are loaded into the timer register upon an overflow condition. If T2_CON.CP_RL2 = 1, this register is loaded with the current timer count upon a negative/positive transition at pin T2EX when EXEN2 = 1.

Table 684 Reset Values of T2_RC2H

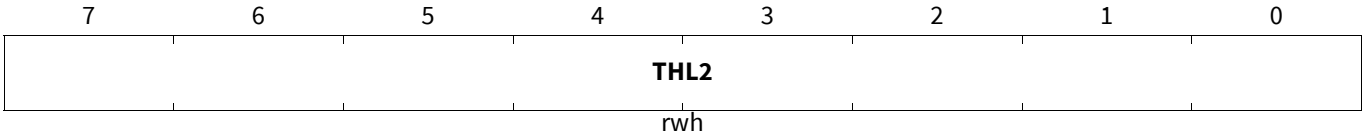
Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.13.2.6.4 Timer 2 Count Register

Registers T2_T2L and T2_T2H (T2CCU_PAGE) hold the current 16-bit value of the Timer 2 count.

Timer 2, Low Byte

T2_T2L
Timer 2, Low Byte ($0C4_H$) Reset Value: Table 685
RMAP: 0, PAGE: T2CCU_PAGE=0



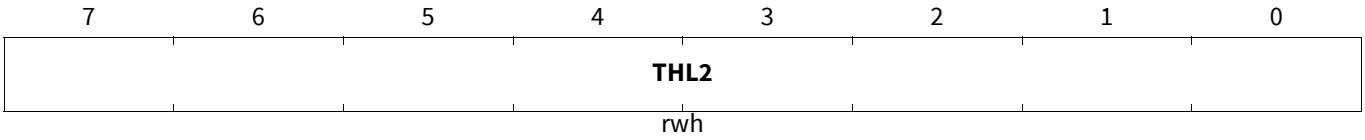
Field	Bits	Type	Description
THL2	7:0	rwh	**Timer 2 Value [7:0]** These bits indicate the lower 8 bits of the current 16-bit timer value.

Table 685 Reset Values of T2_T2L

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Timer 2, High Byte

T2_T2H
Timer 2, High Byte **($0C5_H$)** **Reset Value: Table 686**
RMAP: 0, PAGE: T2CCU_PAGE=0



Field	Bits	Type	Description
THL2	7:0	rwh	**Timer 2 Value [15:8]** These bits indicate the upper 8 bits of the current 16-bit timer value.

Table 686 Reset Values of T2_T2H

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.13.3 Capture/Compare Timer of T2CCU

The Capture/Compare Timer (CCT) is part of the T2CCU, and is a dedicated time base for the CCU capture/compare operations. **Figure 862** shows a block diagram of the CCT timer.

Note: In order to simplify the diagrams, the connection of the CCT contents to the Capture/Compare Unit (CCU) is not shown in the following figures.

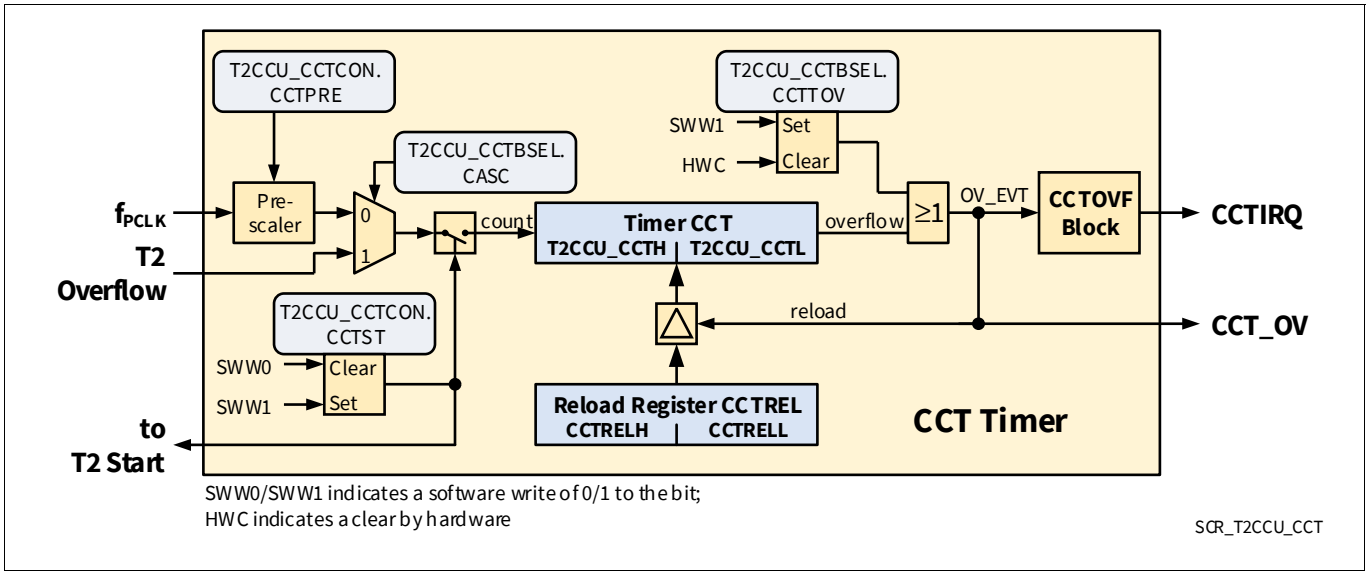


Figure 862 CCT Timer Block Diagram

49.13.3.1 CCT Timer Basic Operation

The features of the CCT are:

- 16-bit timer, comprised of register pair T2CCU_CCTH | T2CCU_CCTL (in the following simply called CCT).
- 16-bit reload register, comprised of register pair T2CCU_CCTRELH | T2CCU_CCTRELL (in the following simply called CCTREL).
- Start or stop controlled by software.
- Input clock prescaler with 12 selectable stages (input clock range from f_{PCLK} to $f_{PCLK}/2048$).
- Once started, the CCT counts up from the initial value stored in its count register to $FFFF_H$. With the next clock pulse, it overflows, and gets loaded with the value stored in register CCTREL. The counting up continues from this value with the following clock pulse(s).
- CCT can be concatenated with Timer 2.
- Optional synchronous start of CCT and Timer 2 possible.
- CCT overflow event is generated either on CCT timer overflow or triggered by software. CCT overflow event performs the following:
  - Reloads CCT with contents of register CCTREL.
  - Sets flag CCTOVF to 1 (can generate an interrupt request).
  - Pulses output line CCT_OV.

When writing to the CCT timer register pair T2CCU_CCTH | T2CCU_CCTL, the following rules have to be obeyed:

- When the CCT is not running (CCTCON.CCTST = 0), a write to the byte registers T2CCU_CCTH/T2CCU_CCTL updates the corresponding byte of the CCT timer.

- When the CCT is running (CCTCON.CCTST = 1), register T2CCU_CCTH has to be written first (this value is held in a shadow register). With the following write to register T2CCU_CCTL, both actual timer registers are updated, and the timer continues counting up from the written value.

The CCT may be stopped at any time by clearing bit CCTST to 0. The timer registers will hold the last count value.

49.13.3.2 Software-Triggered CCT Timer Overflow

A special feature of the CCT timer is the possibility to 'reset' the timer immediately by a Software-Triggered Timer Overflow event. By 'reset', the CCT timer is reloaded and starts counting from the value from the reload registers, while all actions associated with the operation mode of the timer channels on base timer overflow will be executed. When setting bit T2CCU_CCTBSEL.CCTTOV by software, the CCT timer overflow event is triggered asynchronously to the timer count operation. Bit CCTTOV is held at 1 for one PCLK cycle, then it is automatically cleared by hardware.

An example usage for a channel operating in compare mode 0 is illustrated in **Figure 863**. For more details on compare mode 0, refer to **Section 49.13.4.2**.

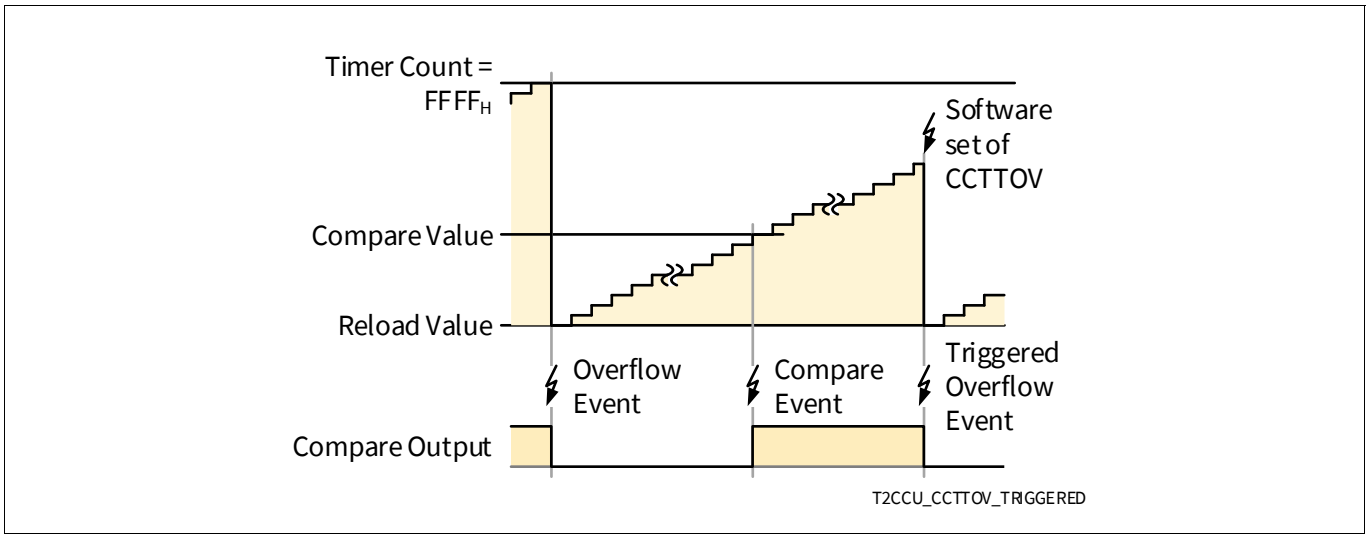


Figure 863 Example of Channel in Compare Mode 0 on 'Triggered Timer Overflow' Event

49.13.3.3 Synchronized Start of Timer 2 and CCT Timer

As described in the previous sections, the two timers of the T2CCU are usually started or stopped individually: Timer 2 via bit TR2, and the CCT via bit CCTST.
However, it is also possible to start both timers synchronously. The procedure required to perform this is listed below:

1. Ensure CCT timer is not already running;
2. Clear bit TR2 (stop Timer 2) and ensure timers are properly initialized;
3. Set bit TIMSYN in register CCTCON to enable synchronized timer starts;
4. Set bit CCTST; this will also set bit TR2.

Steps 3 and 4 can be combined in a single write operation to SFR T2CCU_CCTCON.

To stop the timers, the respective timer bits need to be cleared independently.

49.13.3.4 Cascading Timer 2 and CCT Timer for Flexible Count Rate

Especially for applications requiring the base timer to run at a flexible count rate, it is possible to cascade Timer 2 with the CCT timer. When bit T2CCU_CCTBSEL.CASC is set, the CCT timer will increment its count with every

overflow of Timer 2. In this mode, Timer 2 acts as a prescaler for timer CCT. See **Figure 864** for a block diagram of this option.

Note: Timer 2 may be configured in timer mode or as an external event counter via input T2.

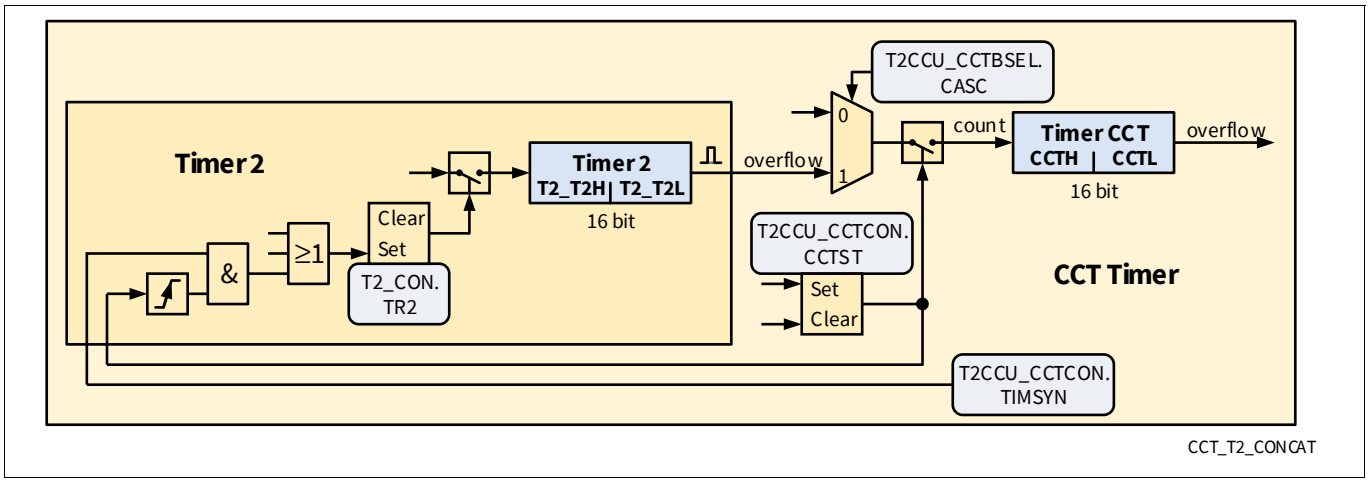


Figure 864 Cascading Timer 2 with the CCT Timer

49.13.3.5 CCT Overflow Flag and Interrupt Request

The overflow flag and the corresponding overflow interrupt request (if enabled) will be set/generated in the following cases:

- By software:
  - By writing a 1 to flag T2CCU_CCTCON.CCTOVF.
  - By writing a 1 to bit T2CCU_CCTBSEL.CCTTOV (Software-Triggered Timer Overflow, see **Section 49.13.3.2**).
- By hardware, when an overflow of the CCT timer occurs.

An interrupt request to the CPU is enabled when the enable bit T2CCU_CCTCON.CCTOVEN is set to 1.

Note that flag CCTOVF must be cleared by software.

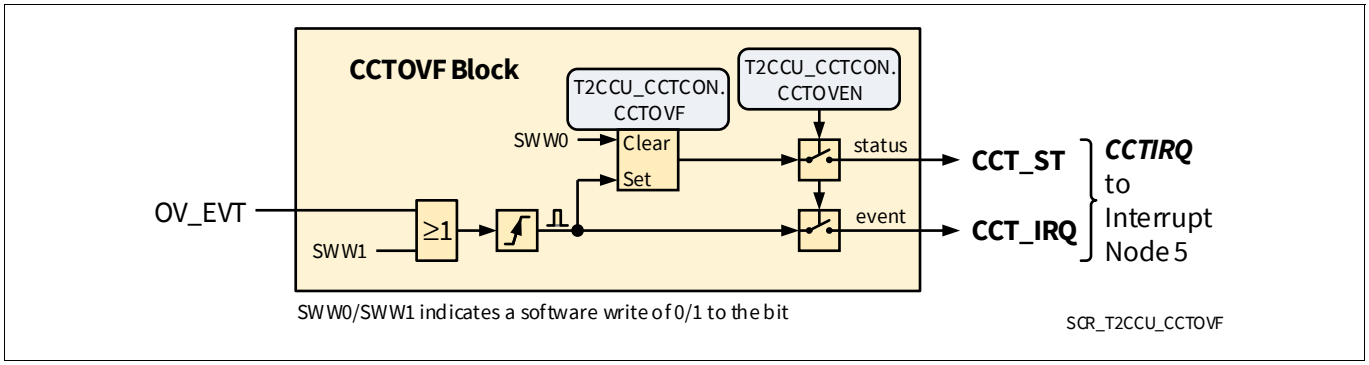


Figure 865 CCT Overflow Flag Block

Note: Although two signals, CCT_IRQ and CCT_ST are connected to the respective interrupt node, the general signal name CCTIRQ is used to represent both signals (interrupt structure 2).

49.13.4 Capture/Compare Unit (CCU) of the T2CCU

The Capture/Compare Unit (CCU) of the T2CCU consists of four 16-bit capture/compare channels (CC0 - CC3) and two 16-bit compare-only channels (CC4 - CC5), illustrated in **Figure 866**.

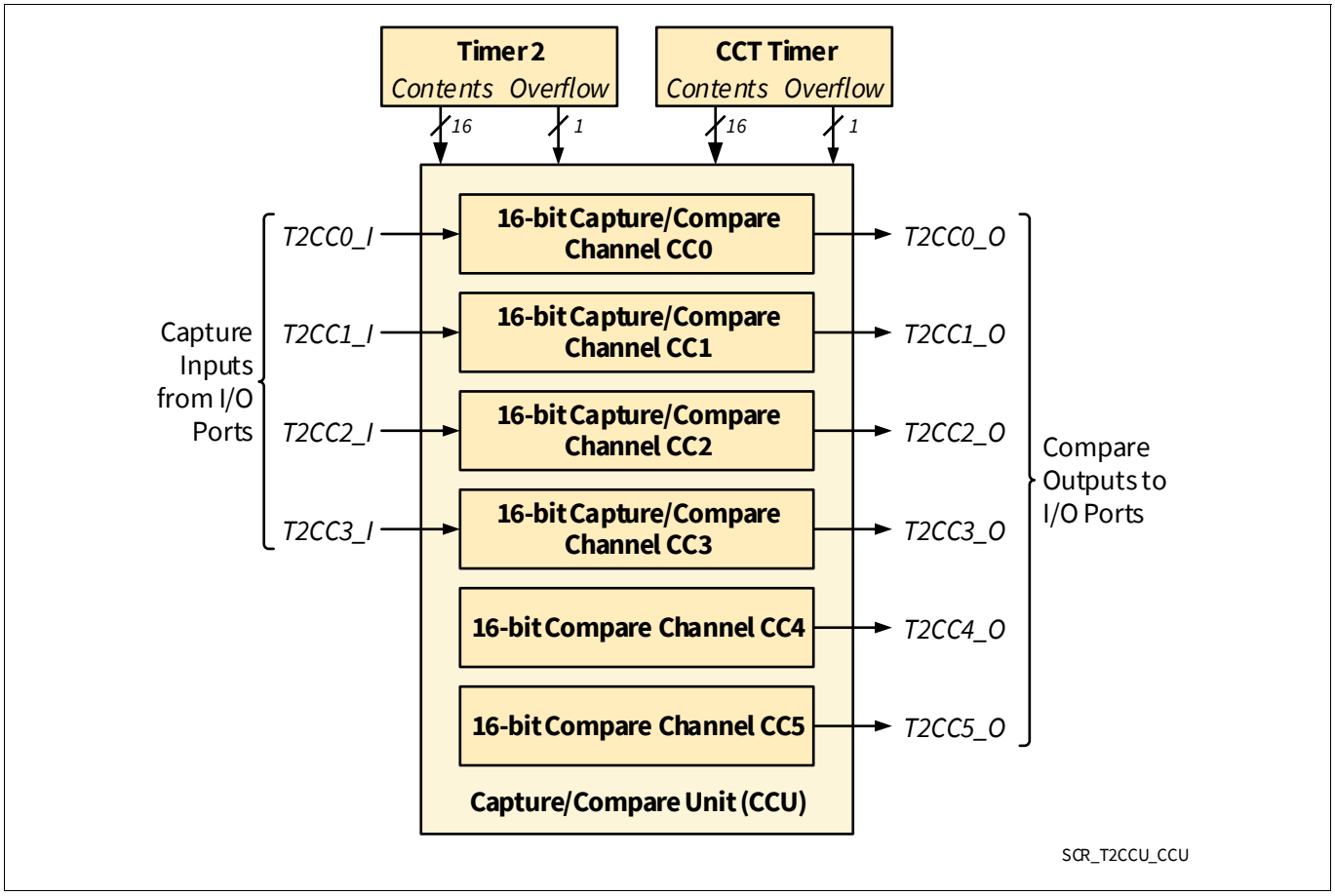


Figure 866 T2CCU Capture/Compare Unit (CCU) Overview

The main features of the CCU are:

- 4 channels (CC0 - CC3), which can be used for capture or compare operation.
- 2 channels (CC4 - CC5), which can be used for compare operation.
- Each capture channel (CC0 - CC3) has a capture input T2CCx_I, coming from the I/O pins.
- Each compare channel (CC0 - CC5) has a compare output signal T2CCx_O, leading to the I/O pins.
- Each channel can individually select either Timer 2 or the CCT as a base timer (using the contents and the overflow signal of the selected timer). The capture/compare resolution for a channel is determined by the operating frequency of the selected base timer.
- Compare Operation: Various compare modes are available:
  - Compare Mode 0 without dead-time generation.
  - Compare Mode 0 with dead-time generation.
  - Compare Mode 1.
  - Concurrent Compare Mode.
- Capture Operation: Two modes are available:
  - Hardware-triggered capture.
  - Software-triggered capture.

The input/output signals of the CCU are assigned as alternate functions of general-purpose I/O ports. The compare outputs share the same pins as the capture inputs, and the ports are (partly) shared with External Interrupt Inputs, see **Table 687**. The corresponding external interrupt request flags are set for the capture/compare function. See also **Section 49.13.4.4**.

Table 687 Input/Output Signals of the CCU

Output/Input	Function
T2CC0_O / T2CC0_I / EXINT3	Compare output/Capture input for channel 0
T2CC1_O / T2CC1_I / EXINT4	Compare output/Capture input for channel 1
T2CC2_O / T2CC2_I / EXINT5	Compare output/Capture input for channel 2
T2CC3_O / T2CC3_I / EXINT6	Compare output/Capture input for channel 3
T2CC4_O	Compare output for channel 4
T2CC5_O	Compare output for channel 5

49.13.4.1 Capture/Compare Operation

The following sections describe the compare and capture operations of the CCU in detail.

Note: For defined behavior, it is required of the user to initialize properly all the T2CCU related SFRs before enabling any T2CCU capture or compare mode.

49.13.4.2 Compare Operation

In compare operation, the contents of a compare register can be regarded as a “time stamp”, at which the corresponding output reacts in a predefined way (with either a positive or negative transition). Variation of this “time stamp” changes the waveform of a rectangular output signal at the pin. As a variation of the duty cycle of a periodic signal, this may be used for pulse width modulation as well as for a continually controlled generation of any kind of square waveforms. Various compare modes are available to cover a wide range of possible applications.

The value stored in a compare register is constantly compared to the current value of the selected base timer. When the timer value is equal to or higher than the compare value, a “compare match” is detected, and appropriate actions are taken.

Figure 867 illustrates the configuration of the CCU for compare operation. The following items are important for the compare operation:

- Each 16-bit compare registers CCx (x = 0..5) is represented by two 8-bit registers, named T2CCU_CCxH (high-byte register) and T2CCU_CCxL (low-byte register).
- The desired compare mode is selected by bit T2CCU_COCON.COMOD, and applies to all enabled compare channels.
- For output polarity control and the dead-time generation, the channels are organized into two groups. Group A comprises channels CC0, CC2, and CC4, while Group B comprises channels CC1, CC3, and CC5.
- In all compare modes, the new output value arrives at the port pin within the same PCLK cycle in which the internal compare signal is activated.
- The compare output is only actively driven when the compare function is enabled. When the compare function is enabled by writing to a channel’s CCMx bits, the respective compare output is latched (as initial value) based on the settings of the channel’s registers and bit.
- The rising edge of the compare match signal is the active edge for all compare match operations.

- Each compare register has a shadow transfer mechanism which allows glitch-free pulse width modulation; see detailed description below.

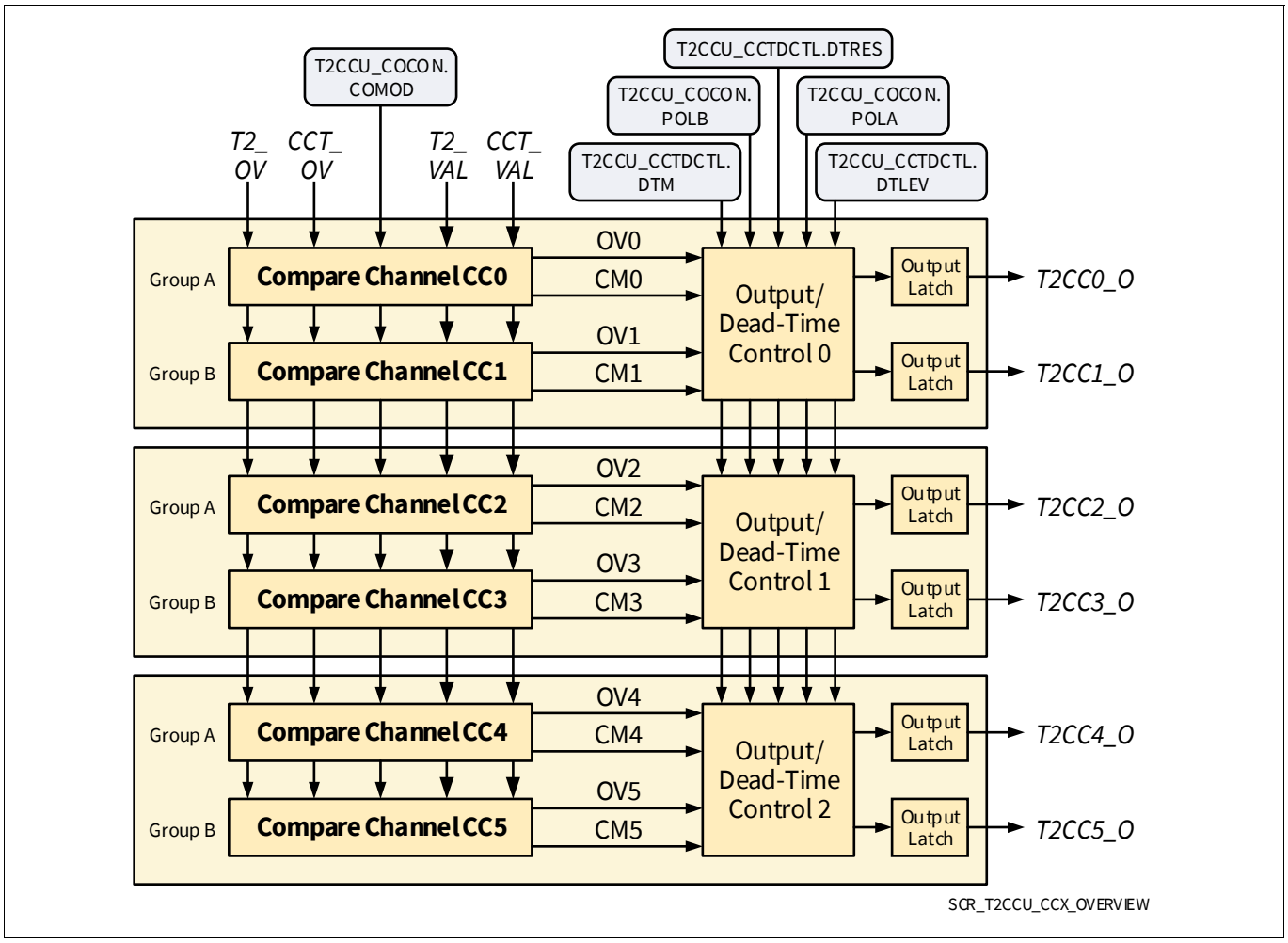


Figure 867 CCU Compare Channel Configuration Overview

Shadow Registers for Compare Registers

To update a 16-bit compare value, two write accesses are required because of the two 8-bit registers of a compare channel. To allow for a glitch free update of a compare value without an intermediate “wrong” value, shadow registers are implemented. Software write accesses do not target the actual compare register, but the respective shadow register. After the write access, the value is transferred from the shadow register into the actual compare register via one of two options:

1. Hardware Control: The transfer from the shadow register into the compare register is triggered by the overflow signal of the selected base timer. In this case, the update of the compare registers is controlled by hardware and is therefore independent from any service delay.
   This option is selected by setting bit T2CCU_COSHDW.TXOV to 1.
2. Software Control: The transfer from the shadow register into the compare register is triggered by setting the corresponding bit T2CCU_COSHDW.COOUTx via software. Bit COOUTx will be cleared automatically by hardware at the next PCLK, after being set.
   This option is selected by setting bit T2CCU_COSHDW.ENSHDW to 1.

In both cases, the rising edge of the event pulse signal is the active edge for the transfer from shadow register to compare register.

Note that it is not allowed to set both bits, TXOV and ENSHDW, at the same time. If bit TXOV is set to enable shadow transfer on timer overflow, bit ENSHDW will be cleared by internal hardware to disable shadow transfer by software.

Figure 868 illustrates the shadow transfer mechanism.

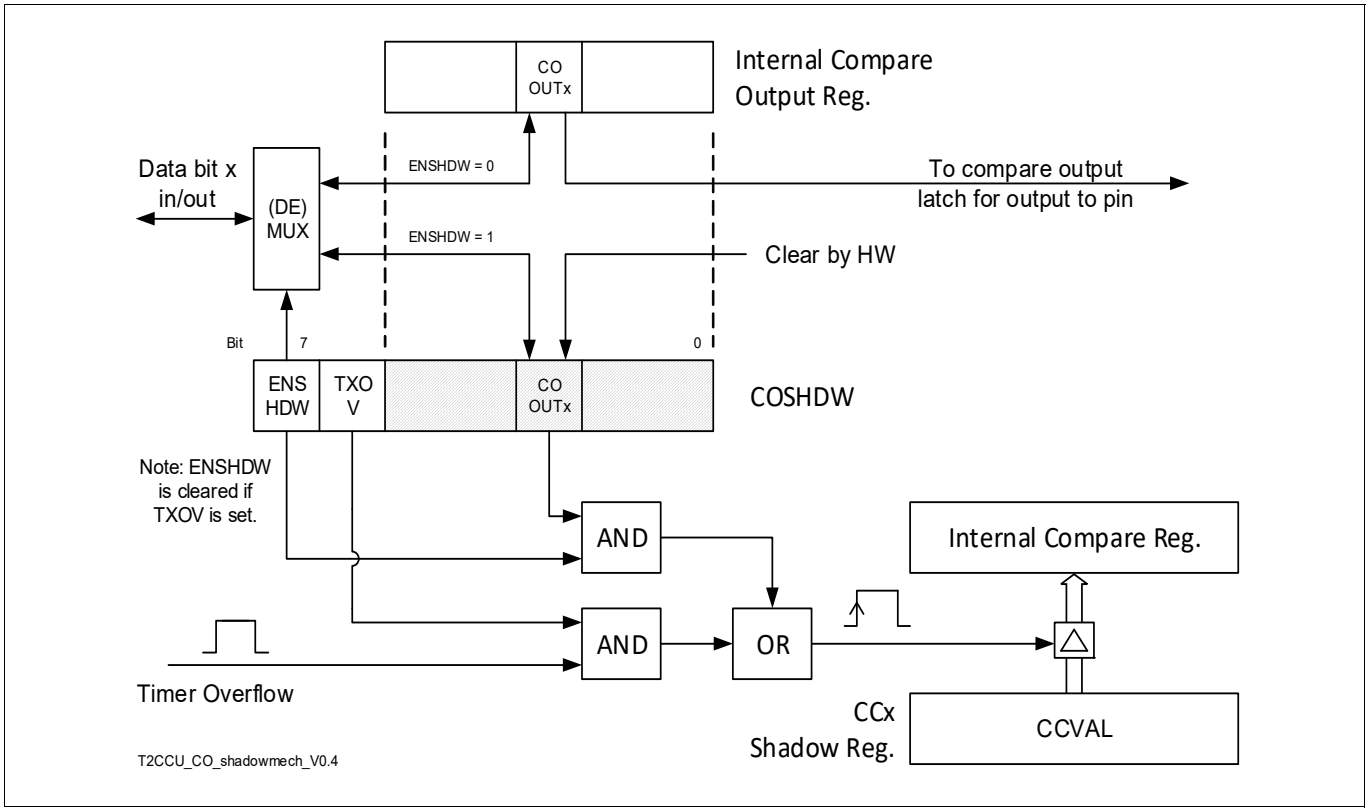


Figure 868 Compare Value Shadow Transfer Mechanism and the Dual Function of Bits COOUTx

Mapped Compare Output Bits

In Compare Mode 1 and Concurrent Compare Mode, bits COOUTx in register T2CCU_COSHDW are used to set the polarity of the respective compare output. To select between the two functions of the COOUTx bits (shadow transfer vs. output polarity), the state of bit ENSHDW in register T2CCU_COSHDW is used (see **Figure 868**):

- When ENSHDW = 1, bits COOUTx can be directly accessed to trigger the shadow transfer of the compare value.
- When ENSHDW = 0, writing or reading to the COOUTx bits actually accesses an internal compare output register instead of the COOUTx bits in register T2CCU_COSHDW.

The following compare modes are available, described in detail in the following sections:

- **Compare Mode 0 (without dead-time control)**.
- **Compare Mode 0 with Dead-Time Control**.
- **Compare Mode 1**.
- **Concurrent Compare Mode**.

49.13.4.2.1 Compare Mode 0 (without dead-time control)

In Compare Mode 0 without dead-time control, the output signal is affected by a compare match and by the overflow of the associated timer. A compare match sets the output to active level, while the timer overflow resets the output to the inactive level.

Compare Mode 0 is ideal for generating pulse width modulated output signals, which in turn can be used for digital-to-analog conversion via a filter network or by the controlled device (e.g. the inductance of a DC or AC motor). Compare Mode 0 may also be used for providing output clocks with initially defined period and duty cycle. This usage needs the least CPU time. Once set up, the output goes on oscillating without any CPU intervention.

Figure 869 shows a timing diagram example of Compare Mode 0, while **Figure 870** provides a functional diagram of this mode. Please see **Figure 871** for the output control.

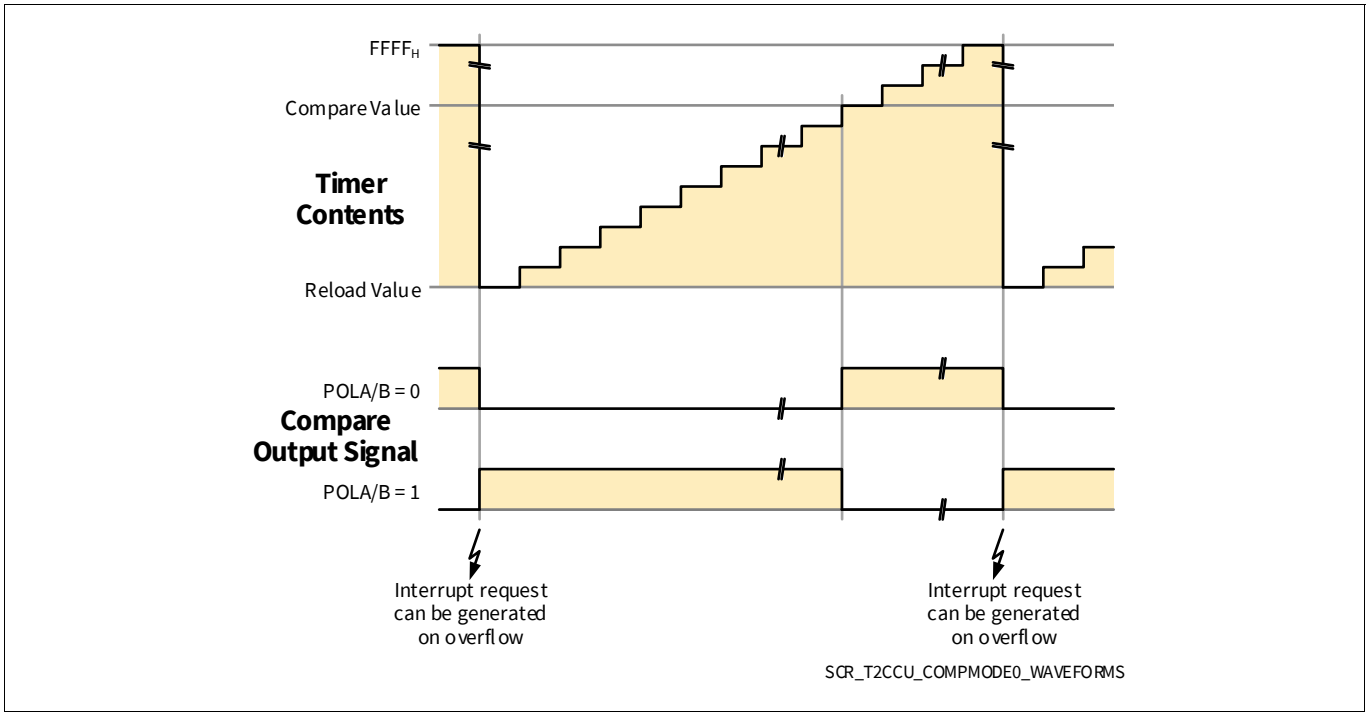


Figure 869 Channel Output Example in Compare Mode 0 (without dead-time control)

To configure Compare Mode 0, the following steps need to be done:

- Set bitfield COMOD in register T2CCU_COCON to 00_B to select Compare Mode 0.
- Set bit TXOV in register T2CCU_COSHDW to 1_B (the usual way to perform the compare shadow value transfer in Compare Mode 0 is via the timer overflow).
- Configure the desired output polarity of the compare outputs via bits POLA and POLB in register T2CCU_COCON. Bit POLA applies to channels 0, 2, and 4 (channel group A), while POLB applies to channels 1, 3, and 5 (channel group B).

Note: The two steps listed above can be done in one write access to register T2CCU_COCON. However, the enabling of compare channels 4 and 5 (see following step) must be done in a separate write access to register COCON, to ensure a glich-free compare output level.

- For compare channels 0 through 3, individually enable the compare mode for the desired compare registers by setting bitfield CCMx in register T2CCU_CCEN to 10_B.
  For compare channels 4 and 5, individually enable the compare mode for the desired compare registers by setting bit CCMx in register T2CCU_COCON to 1_B.

The compare output is immediately set to the appropriate level when the associated compare register is enabled via bit/bitfield CCMx:

- To the active level if the base timer contents are equal to or greater than the compare register contents.

- To the inactive level if the base timer contents are less than the compare register contents.

Either Timer 2 or the CCT timer can be used as base timer in Compare Mode 0. The shadow transfer of a new compare value is usually triggered by the overflow of the selected base timer (set bit TXOV in register T2CCU_COSHDW to 1_B).

Note that no interrupt is generated when a compare match occurs. See **Section 49.13.4.4** for further information.

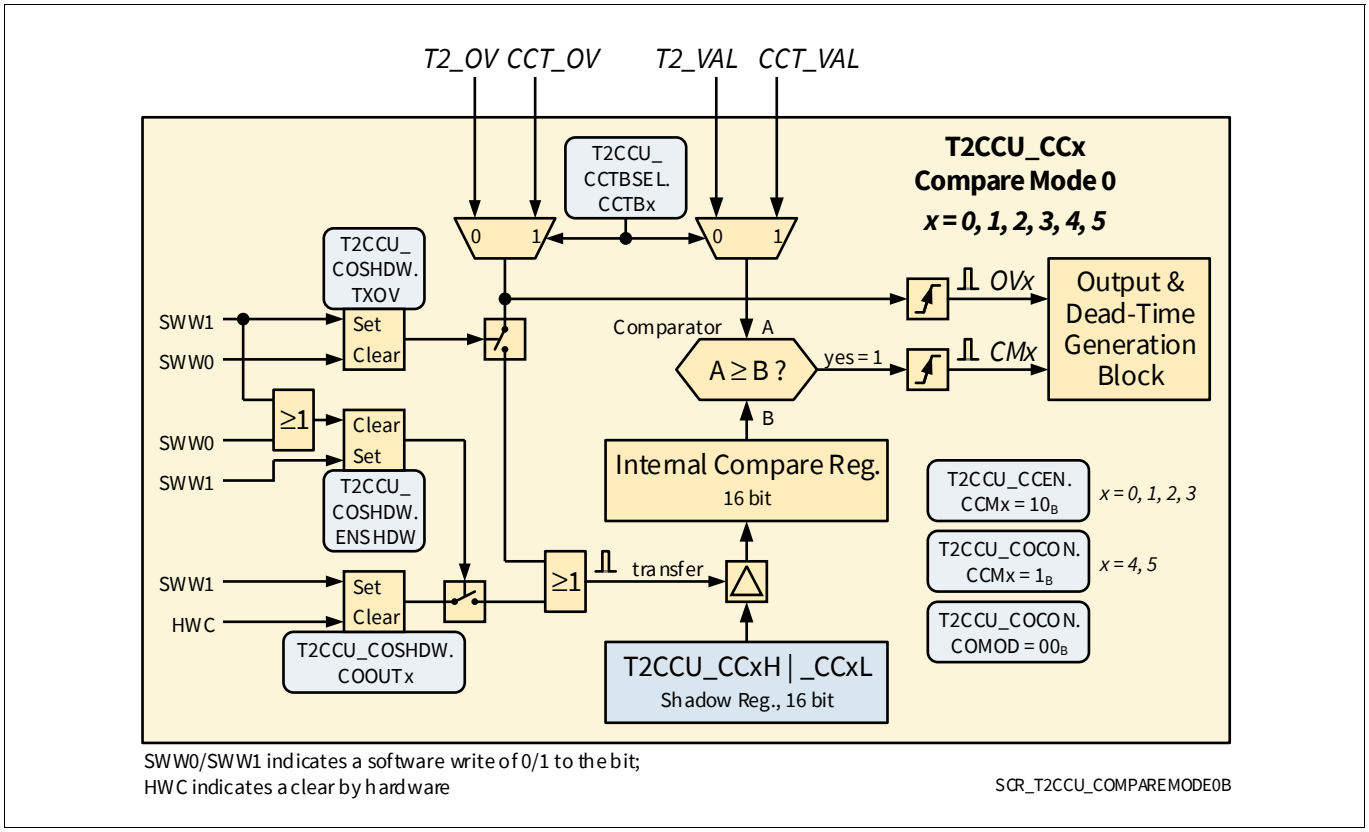


Figure 870 Functional Diagram of Compare Mode 0 (without dead-time control)

Operation in the first timer period after initialization

The following needs to be regarded concerning the compare output level for the first timer period after initialization:

- Although a compare value might have been written to the compare shadow register before enabling the compare operation, the actual compare register will contain 0000_H (if no transfer from the shadow register to the actual compare register has been forced via software, see below).
- In this case, the value of the timer will be greater than or equal to the value in the compare register (in any case - whether the timer contains the initial value of 0x0 or has been programmed to another value).
- Thus, the condition of a “compare match” is true (timer ≥ compare value), and the compare output level will be set to the active level.
- On the following timer overflow, the first compare value shadow transfer takes place (bit T2CCU_COSHDW.TXOV = 1_B), and the output will go to its inactive level if the programmed compare value is > 0000_H.
- From this second timer period on, the output will switch from inactive to active level on compare match and vice versa on timer overflow.

In order to get a proper compare operation already during the first timer period, a compare value shadow transfer must be forced by software. This can only be done after compare mode is enabled.

- After enabling the compare mode and before starting the timer, a compare value shadow transfer must be forced by software.
- Write a value greater than the timer value to the compare shadow register.
- Set bit T2CCU_COSHDW.ENSHDW to 1_B (clear bit T2CCU_COSHDW.TXOV to 0_B with the same write access), and set T2CCU_COSHDW.COOUTx to 1_B.
- After this, clear bit T2CCU_COSHDW.ENSHDW to 0_B and set bit T2CCU_COSHDW.TXOV to 1_B with the same write access (bit T2CCU_COSHDW.COOUTx is cleared automatically by hardware).
- Now the actual compare register contains the programmed value (> timer value), and the compare output goes to inactive level.
- The timer can now be started for its first period.

PWM Signal from 0% to 100% Duty Cycle in Compare Mode 0

The active level of a PWM signal refers to the compare output signal level from the time of compare match until the next timer overflow. To generate signals with duty cycles of 100% or 0%, the following methods can be used:

- To support a 100% duty cycle for a compare channel with T2CCU_COCON.POLA/B = 0 (active level = high), the compare value must be set equal to the timer reload value. In this case, the internal hardware will not cause the compare output signal to the inactive level on timer overflow, and the output stays at the active level until the next timer overflow, when the compare value is no more equal to the timer reload value.
- To support a 0% duty cycle for a compare channel with T2CCU_COCON.POLA/B = 0, the compare value must be smaller than the timer overflow reload value. In this case, the timer reload value must not be zero. The compare match event will not happen, and the compare output stays at the inactive level (low).

The mechanisms described above apply correspondingly for the case that the chosen active level = low (T2CCU_COCON.POLA/B = 1).

49.13.4.2.2 Compare Mode 0 with Dead-Time Control

Compare Mode 0 offers an additional feature for the generation of complementary signals with a so-called dead-time between the deactivation of one signal and the activation of the other signal.

The generation of (complementary) signals with dead-time for the high-side and the low-side switches of a power inverter phase is based on twin grouping of the consecutive compare channels – one from group A and the other from group B. The three compare channel pairs therefore are comprised of channel 0 with 1, channel 2 with 3, and channel 4 with 5.

The Compare Mode 0 with dead-time generation can be used under the following conditions:

- All involved compare channels must use the CCT timer.
- Both channels of a channel pair must have the same compare value (CC0=CC1; CC2=CC3; CC4=CC5).
- All used channels must have the dead-time generation enabled (bit DTEy in register T2CCU_CCTDCTH = 1).
- All high-side and low-side switches must have the same active polarity. E.g., if the high-side switch should be active while the CCT timer value is above the compare value, then the low-side switch should be active while the CCT timer value is below the compare value.
- The active signal polarity is controlled by the bits T2CCU_COCON.POLA (for compare channel group A) and T2CCU_COCON.POLB (for compare channel group B). POLA and POLB must be set to complementary values.

Figure 871 illustrates the dead-time generation. This logic is implemented three times, one for each compare channel pair. It mainly consists of an 8-bit down-counter (dead-time counter, clocked with the CCT count rate), and a logic to manipulate the output signals.

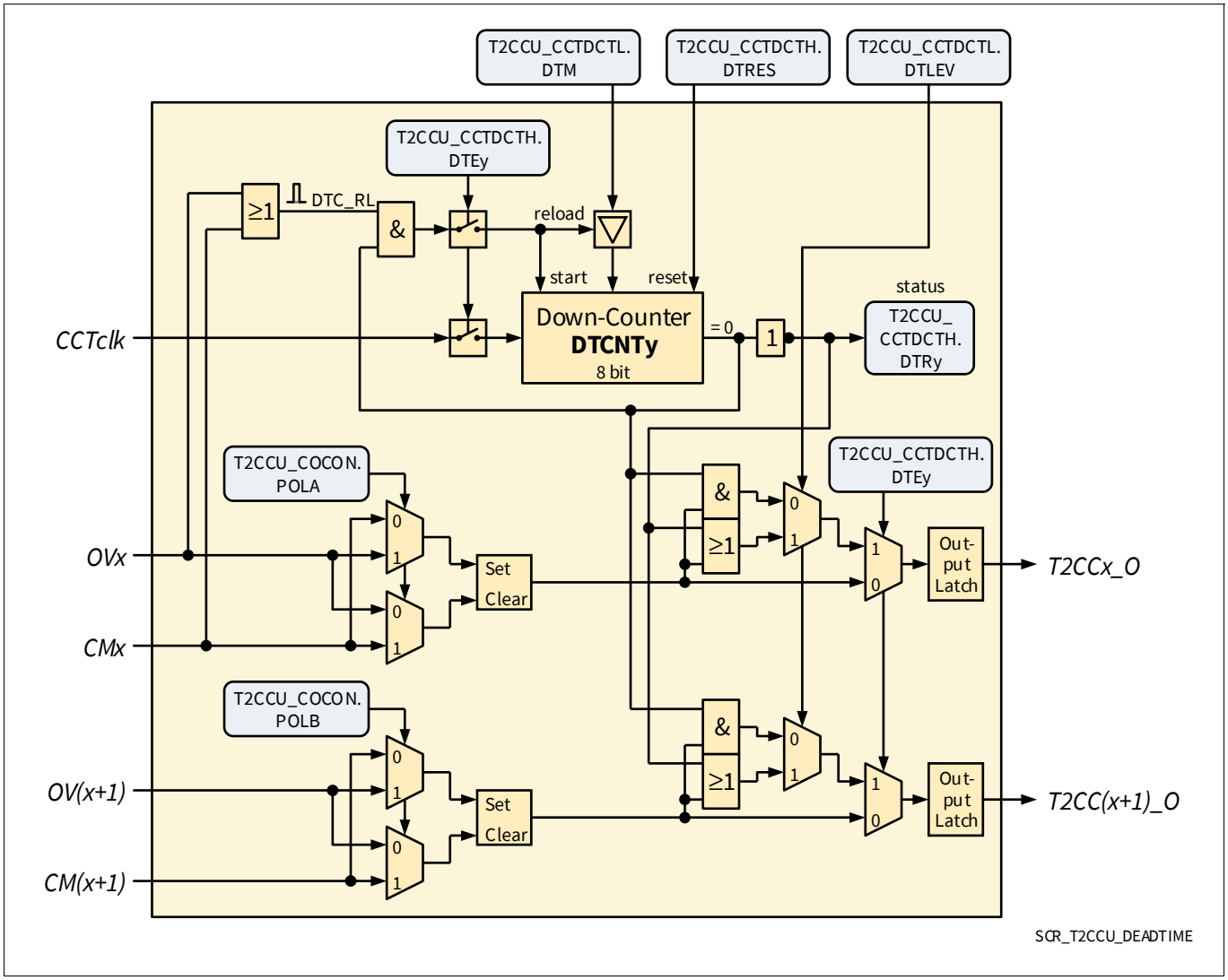


Figure 871 Output & Dead-Time Generation Block

The detailled operation is as follows:

- If dead-time is not enabled for a channel (bit DTEy in register T2CCU_CCTDCTH = 0), the output signal polarity is solely controlled by bits POLA/POLB.
- If dead-time is enabled for a channel (bit DTEy in register T2CCU_CCTDCTH = 1), the output signal polarity is controlled by bits POLA/POLB and the dead-time control logic:
  - As long as the dead-time counter is not running (signal '= 0' is active, status flag T2CCU_CCTDCTH.DTRy = 0), the output signals are determined by bits POLA/POLB.
  - Each instance of the CCT timer overflow or a compare match event (of channel from group A, rising edge) triggers the corresponding dead-time counter.
    The trigger pulse (DTC_RL) reloads the dead-time counter with the value programmed in bitfield T2CCU_CCTDTCL.DTM, and starts it. This can only take place if the dead-time feature is enabled by bit T2CCU_CCTDTCH.DTEx and while the counter is zero.
  - Once started, the dead-time counter counts down until it reaches 0. Then it stops counting.
  - As long as the dead-time counter is running, the output signals are forced to their programmed dead-time level. This level is determined by bit DTLEV in register T2CCU_CCTDCTL. DTLEV must be set such that the outputs a forced to the inactive level during the dead-time.

Please see also the section **Operation in the first timer period after initialization** on **Page 221**, which applies in a similar manner for the dead-time operation.

Figure 872 illustrates the resulting signal waveforms for the dead-time operation (example with POLA = 0 and POLB = 1).

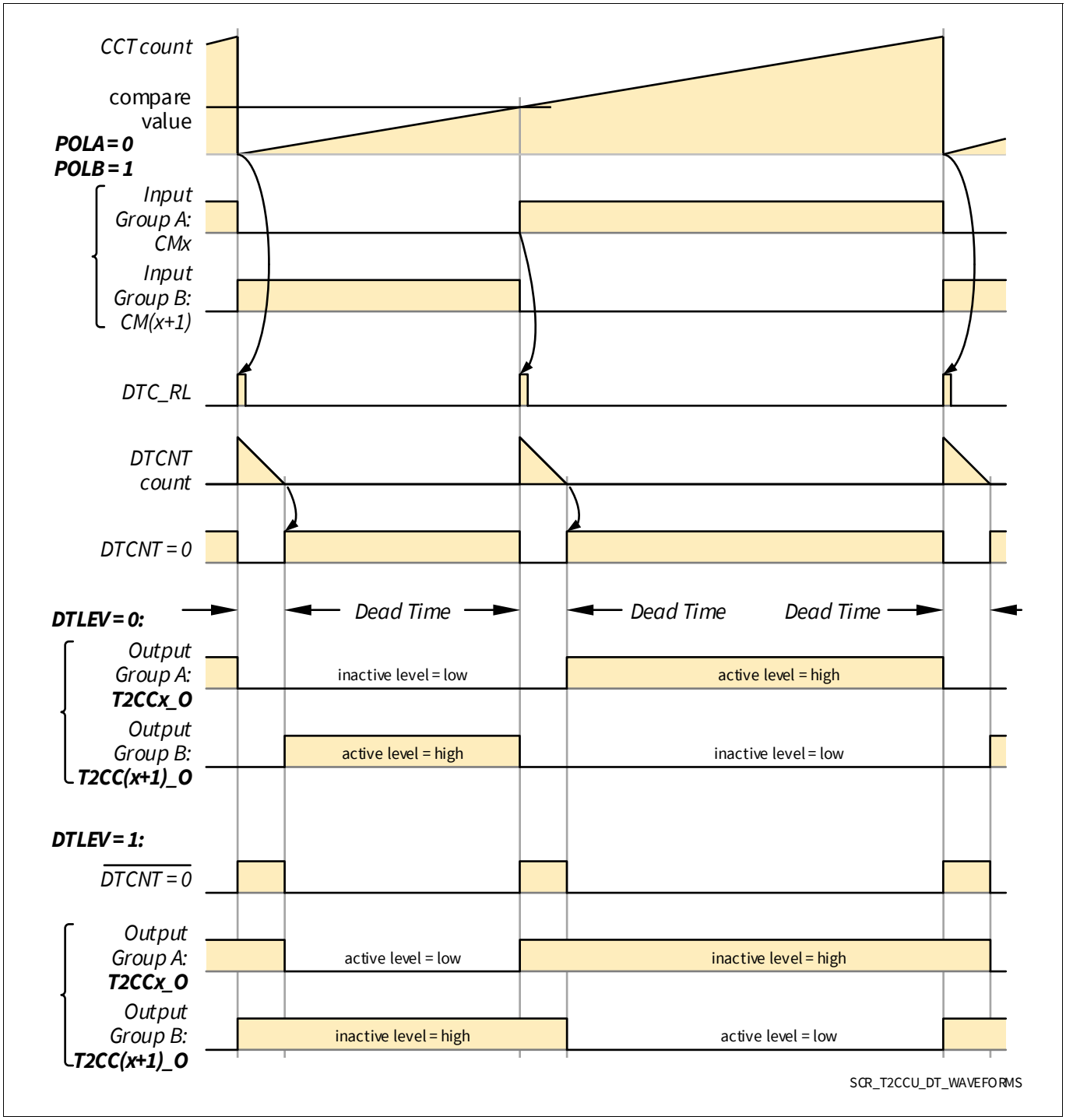


Figure 872 PWM-Signals with Dead-Time Generation (illustrated for one channel pair)

Note: While the channels pairs can be individually enabled or disabled for dead-time generation via their DTEy bits, the dead-time (DTM) and the dead-time level (DTLEV) apply to all channels. In addition, the reset control (DTRES) applies to all three dead-time counters.

Note: The dead-time logic is designed for the application case in which one and only one compare event is generated within one timer period. If more than one compare event happens within one timer period or if two or more timer overflows happen without a compare event in between, proper functionality of the dead-time logic is not assured.
In addition, the timer must not be disabled while the dead-time is in progress or between a timer overflow and a compare match.

Note: When the dead-time DTM in register T2CCU_CCTDTCL is set to 0_H, no dead-time will be generated. This is equivalent as if the dead-time function is not enabled (e.g. DTEy = 0_B).

49.13.4.2.3 Compare Mode 1

In Compare Mode 1, the output signal is only updated on a compare match, but not by the respective timer overflow (as in Compare Mode 0). In Compare Mode 1, the software adaptively determines the transition of the output signal. It is commonly used when output signals are not related to a constant signal period (as in a standard PWM Generation), but must be controlled very precisely with high resolution and without jitter. In Compare Mode 1, the compare output signal level is under the full control of the software.

Figure 873 shows a functional diagram of Compare Mode 1.

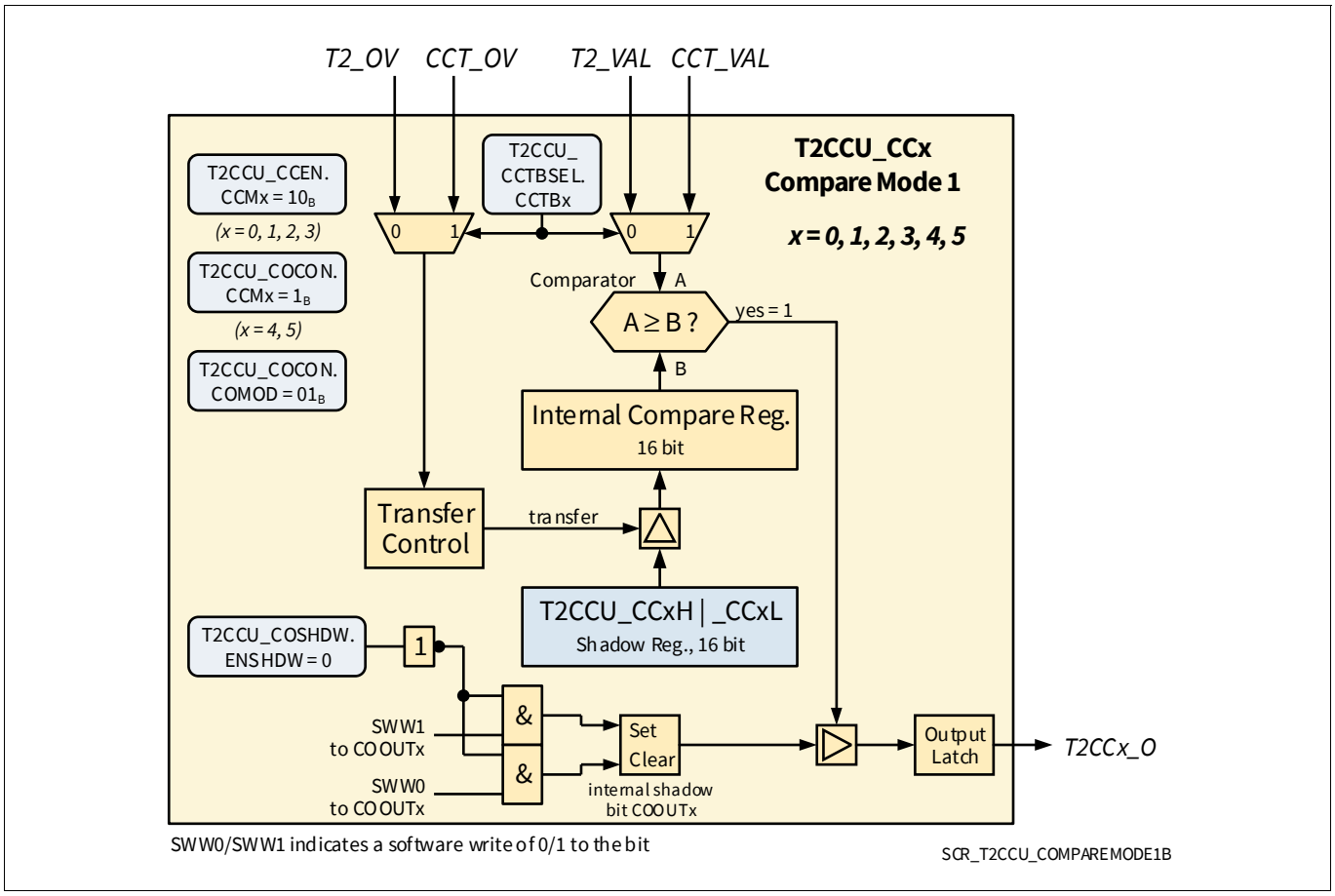


Figure 873 Functional Diagram of Compare Mode 1

To configure Compare Mode 1, the following steps need to be done:

- Set bitfield COMOD in register T2CCU_COCON to 01_B to select Compare Mode 1.
- For compare channels 0 through 3, individually enable the compare mode for the desired compare registers by setting bitfield CCMx in register T2CCU_CCEN to 10_B.

For compare channels 4 and 5, individually enable the compare mode for the desired compare registers by setting bit CCMx in register T2CCU_COCON to 1_B.

- The shadow transfer of a new compare value is triggered by the overflow of the selected base timer (T2CCU_COSHDW.TXOV = 1).
- The software writes the desired output level for the next compare match to the COOUTx bit(s) of the internal compare output register (in this case, bit T2CCU_COSHDW.ENSHDW must be cleared to 0). This new value will not appear at the output until the next compare match occurs.
  Thus, one can choose, independently for each compare channel, whether the output signal is to toggle (1-to-0 or 0-to-1) or to remain at the same level, when a compare match occurs.

The compare output is immediately set to the value initialized to bit COOUTx (of the internal compare output register), when the associated compare register is enabled via bit/bitfield CCMx.

Either Timer 2 or the CCT timer can be used as base timer in Compare Mode 0. The shadow transfer of a new compare value must be triggered by the overflow of the selected base timer.

Note that no interrupt is generated when a compare match occurs. See **Section 49.13.4.4** for further information.

49.13.4.2.4 Concurrent Compare Mode

Concurrent Compare Mode allows to manipulate more than one output with the same compare event. Up to all six compare outputs can be updated concurrently when a compare channel 0 event occurs.

Concurrent compare is an ideal and effective mode for applications where more than one synchronous output signal is to be generated. Such applications could include, e.g., a multiple-phase stepper motor control, or control of ignition coils of a car engine. These applications generally require predefined bit-patterns to be put to an output port at precisely predefined moments.

Figure 874 illustrates the function of Concurrent Compare Mode.

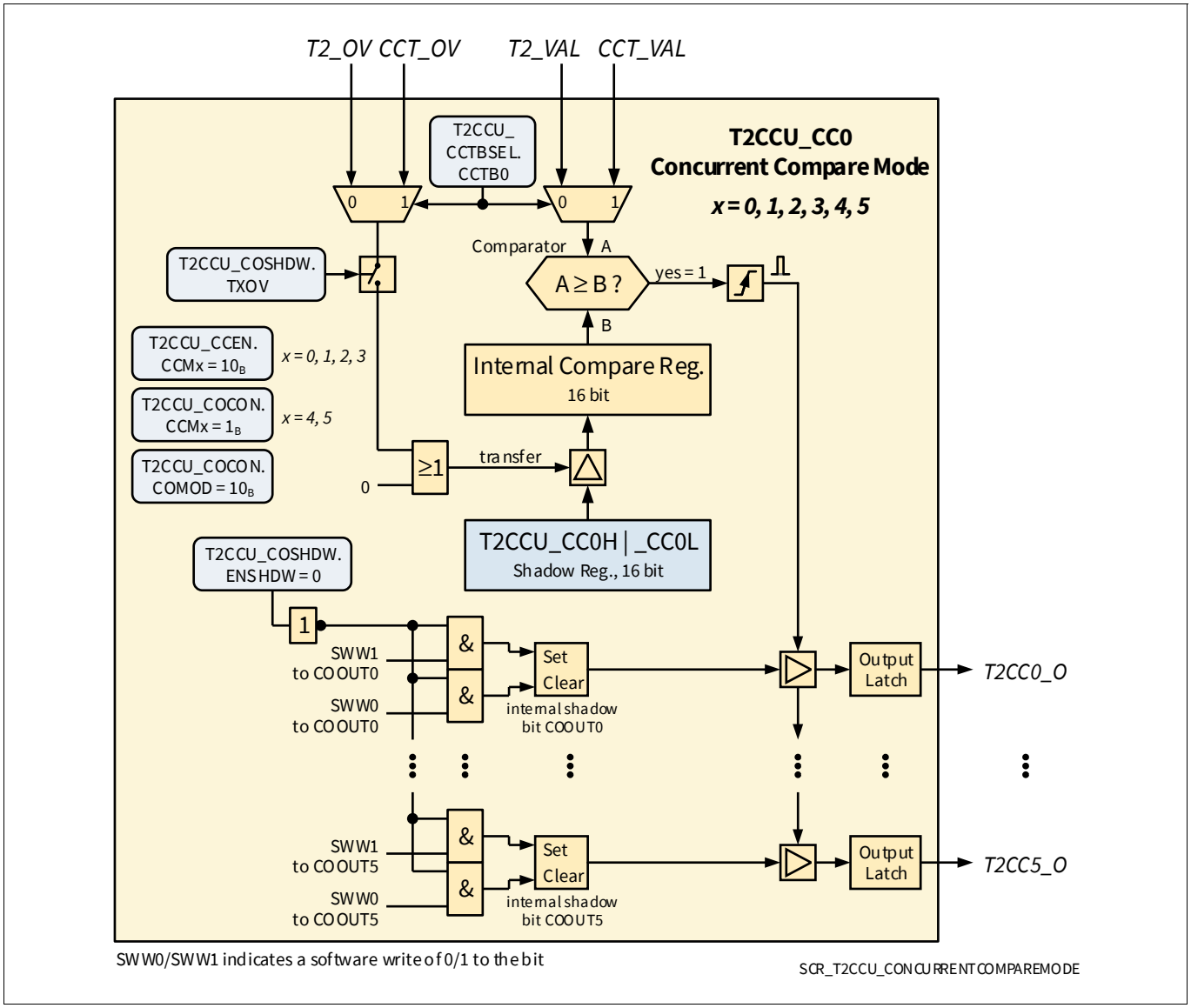


Figure 874 Functional Diagram of Concurrent Compare Mode

To operate the CCU in Concurrent Compare Mode, the following configurations must be made:

- Bit T2CCU_COSHDW.ENSHDW must be cleared to 0.
- The shadow transfer is triggered by the overflow of the selected base timer; set bit T2CCU_COSHDW.TXOV to 1_B.
- Set the compare mode to Concurrent Compare Mode, by setting bitfield T2CCU_COCON.COMOD to 10_B.
- Write the initial output values to the internal compare output bits T2CCU_COSHDW.COOUTx of the desired outputs.
- Only Compare Channel 0 is used for the outputs. Write the required compare value to T2CCU_CC0H and T2CCU_CC0L.
- Enable the desired output signals via their individual enable bits (for channels 0 to 3: Set bitfield T2CCU_CCEN.CCMx to 10_B; for channels 4 and 5: Set bit T2CCU_COCON.CCMx to 1_B).

Note: The compare outputs are latched with the corresponding values initialized to the COOUTx bits of the internal compare output register when setting the CCMx enable control for the desired outputs (make

sure to first select Concurrent Compare Mode via T2CCU_COCON.COMOD, and to program the desired values into the COOUTx bits).

- During steady operation, write the new compare value to register pair T2CCU_CC0H and T2CCU_CC0L, and the new output values to bits T2CCU_COSHDW.COOUTx.
- With the following timer overflow, the new compare value will be used during the new timer period, and the new output values will appear at the outputs when the compare match occurs.

Figure 875 illustrates two examples of how four rectangular waveforms can be generated at the port using a pattern table. The patterns are written to the internal compare output register (with T2CCU_COSHDW.ENSHDW = 0) before the corresponding timer count is reached. The (future) timer count at which the pattern shall be available at the port must be loaded to the compare shadow registers T2CCU_CC0H and T2CCU_CC0L, and transferred to the internal compare registers. To manage the waveform generation, one example uses a time schedule table with list of compare values, the other example makes use of a fixed compare value (such as $FFF0_H$) with shorter timer period. In these ways, each channel can be controlled at predefined time(s) by the user on whether to toggle or not.

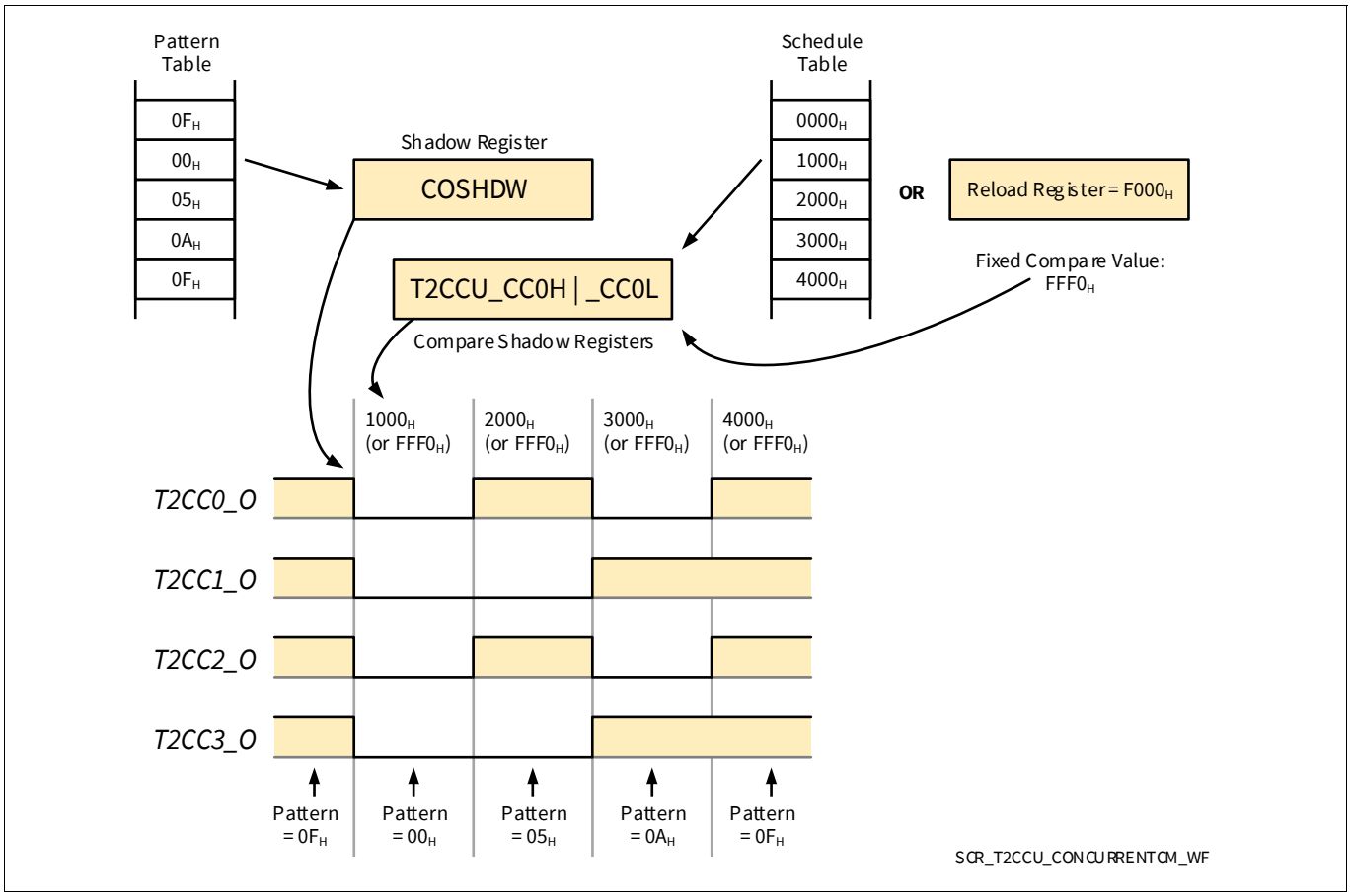


Figure 875 Two Examples of Waveform Generation in Concurrent Compare Mode (by Schedule Table of Compare Values or Fixed Compare Value)

49.13.4.2.5 Using Interrupts in Combination with the Compare Function

There are no interrupt requests generated within the CCU on a compare match. Thus, it is not possible to directly trigger an interrupt service routine on a compare match to handle the programming of new compare and/or output values.

However, there are two options to overcome this:

- In many cases, the overflow interrupt request of the associated base timer can be used.
- Since the compare outputs are connected to the same pins which serve as external interrupt inputs (and CCU capture inputs), these interrupts can be used.

These options offer the following possibilities:

- Compare Mode 0 (with or without dead-time):
  In this mode, a compare match will always show a defined signal edge at the compare output. This signal edge will also appear on the associated input line for the external interrupt input. By appropriately programming the external interrupt configuration, an interrupt request to the CPU can be generated. In the associated service routine, the next compare value can be written to the compare shadow register.
- Compare Mode 1 and Concurrent Compare Mode:
  In these modes, the shadow transfer of compare and output value is always triggered with the overflow of the selected base timer. Thus, the associated overflow interrupt of the timer is the right trigger for a respective service routine to set up new compare and output values.

See also **Section 49.13.4.3.3** for more information for configuring the external interrupt input.

49.13.4.3 Capture Function

The capture modes of the CCU allow the measurement of external signal pulse widths or the latching of time-stamps via the capture/compare registers CC0 through CC3. Each of the 16-bit registers can be used to latch the current 16-bit count value of the selected base timer. Two capture modes are provided for this function, a hardware- and a software-triggered capture:

- In Capture Mode 0, an external event at input T2CCUx_I latches the contents of the selected base timer into the associated capture register.
- In Capture Mode 1, a software write to the low byte of a capture register latches the contents of the selected base timer into the associated capture register.

The two capture modes can be established individually for each capture channel. That means, in contrast to the compare modes, it is possible to simultaneously select Mode 0 for one capture channel and Mode 1 for another channel. **Figure 876** shows the capture operation a capture/compare channel.

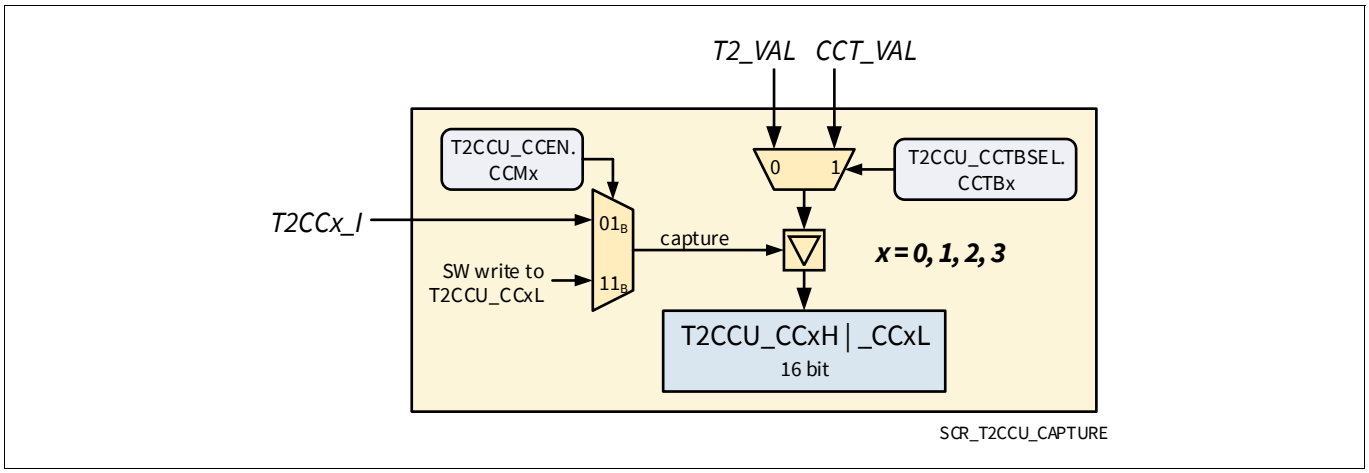


Figure 876 Functional Diagram of Capture Mode

49.13.4.3.1 Capture Mode 0

In Capture Mode 0, any selected external event (rising/falling/both edge) at the associated input (T2CC0_I through T2CC3_I) latches the channel timer contents to the corresponding channel capture register. The capture edge is defined by the corresponding external interrupt set up (each capture input shares the same pin with an external interrupt). The timer contents are latched into the corresponding capture register in the PCLK cycle following the one in which the external transition was recognized.

To configure Capture Mode 0, the following steps need to be done:

- Configure the external input signal (and interrupt request) associated with the desired capture channel (see **Section 49.13.4.3.3**).
- Select the base timer for the desired capture register via T2CCU_CCTBSEL.CCTBx.
- Set bitfield T2CCU_CCEN.CCMx to 01_B (capture mode 0).

49.13.4.3.2 Capture Mode 1

In Capture Mode 1, a capture will occur upon writing to the low byte of the corresponding channel capture register. Thus, the software is able to read the contents of the channel timer “on-the-fly”. The value written to the capture register is irrelevant for this function. The timer contents will be latched into the channel capture register following the write instruction. In this mode, no interrupt request will be generated.

To configure Capture Mode 1, the following steps need to be done:

- Select the base timer for the desired capture register via T2CCU_CCTBSEL.CCTBx.
- Set bitfield T2CCU_CCEN.CCMx to 11_B (capture mode 1).
- Write any value to the low-byte register T2CCU_CCxL to trigger the capture operation.

49.13.4.3.3 Configuring the External Capture Input

Each capture input shares the same pin with an external interrupt. In fact, the capture input T2CCUx_I is derived from the external interrupt input such that it is connected from the edge selection multiplexer for the external interrupt to the CCU of the T2CCU. This is exemplary illustrated in **Figure 877** for capture input T2CC0_I.

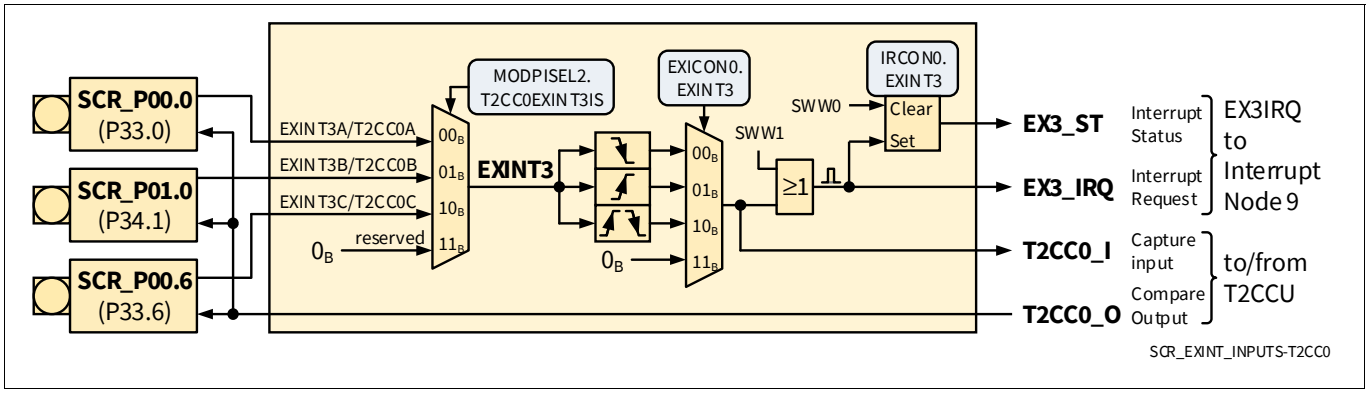


Figure 877 Capture Input Configuration

An external interrupt input is sampled in every PCLK cycle. When a sample shows a low (high) level in one PCLK cycle and a high (low) in the next PCLK cycle, a transition is recognized. The corresponding external interrupt request flag(s) (EXINT3 to EXINT6) is set. If the interrupt(s) is enabled, the CPU will vector to the appropriate interrupt service routine.

When a hardware capture mode is programmed for a CCU capture channel, the edge selected for the associated external interrupt input is also used as the relevant capture trigger. The timer contents are latched to the corresponding capture register in the PCLK cycle following the one in which the transition was identified.

When configuring an external capture input, the following points are relevant:

- The desired input pin is selected via bitfield T2CCxEXINTyIS in register MODPISEL2.
- The external interrupt inputs EXINT3 through EXINT6, which are shared with the capture inputs T2CC0 through T2CC3, all share one interrupt node, XINTR9.
- The common enable bit for this interrupt node is IEN1.EXM.
- The active edge is selected via bitfields EXINTx (x = 3 to 6) in registers EXICON0 resp. EXICON1.
- The interrupt status flags are EXINT3 through EXINT6 in register IRCON0.

Compare Match Interrupts

Since a compare match does not directly generate an interrupt request, the external interrupt inputs, which are shared with the compare outputs, can be used to generate an appropriate interrupt request, if required.

For this, the measures described in the section above for the capture inputs are taken. Since the channel is programmed for one of the compare modes, the capture input line has no effect.

The change of the compare output signal on a compare match is then used to trigger the associated interrupt request.

The main advantage in using compare match interrupts is to avoid unintentional overwriting of compare/compare output registers before a match has been reached. This could happen when the CPU writes to the compare registers without knowing about the actual timer count.

Note: As an alternative, the timer overflow interrupt may be used in place of the compare match interrupt.

49.13.4.4 CCU Interrupt and I/O Connections

This section describes the connections of the capture/compare channels to interrupts, as well as to the I/O ports (and other modules).

Usually, the interrupt request blocks are used which also provide the respective capture input T2CCx_I. However, since the capture/compare I/O lines are connected to more than on port pin, alternative interrupt requests can be used. In addition, the connection to other SCR modules is shown.

49.13.4.4.1 Capture/Compare Channel 0

Figure 878 illustrates the connections of capture/compare channel 0 to the ports, interrupt request blocks, and other SCR modules.

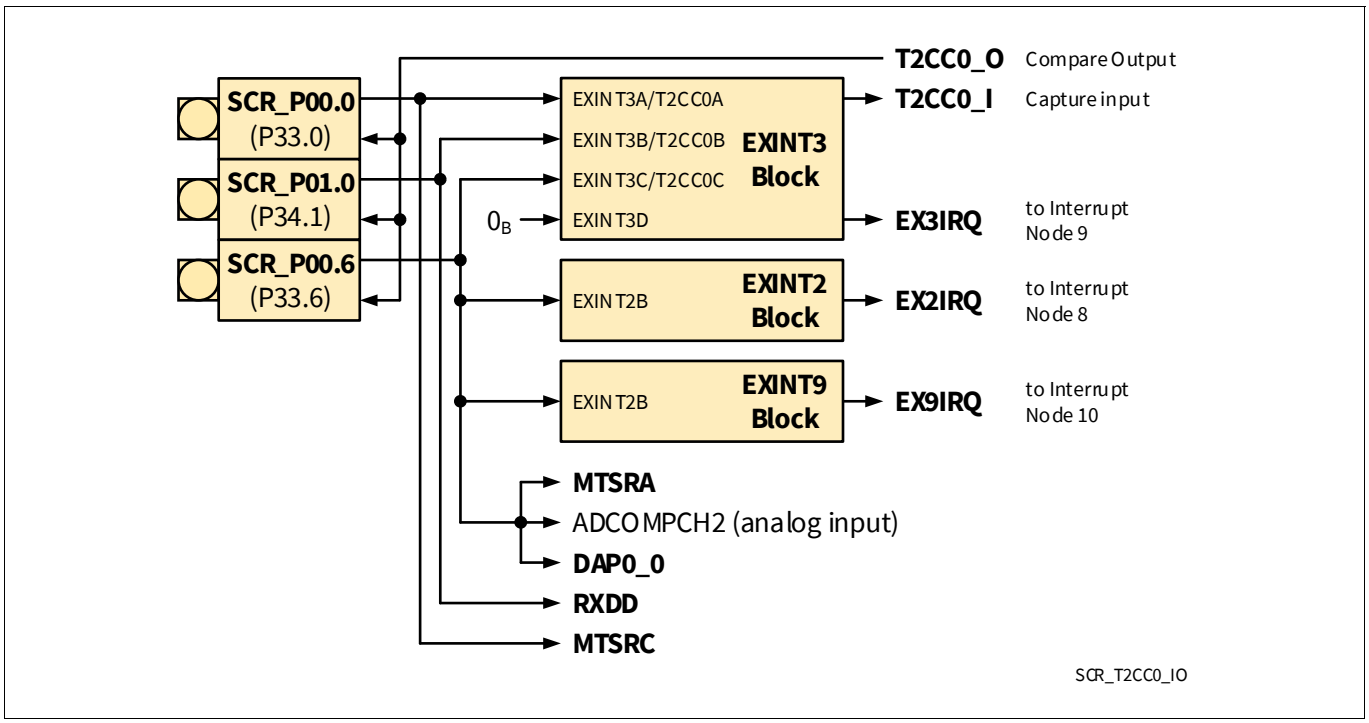


Figure 878 Interrupt and I/O Connections for Capture/Compare Channel 0

49.13.4.4.2 Capture/Compare Channel 1

Figure 879 illustrates the connections of capture/compare channel 1 to the ports, interrupt request blocks, and other SCR modules.

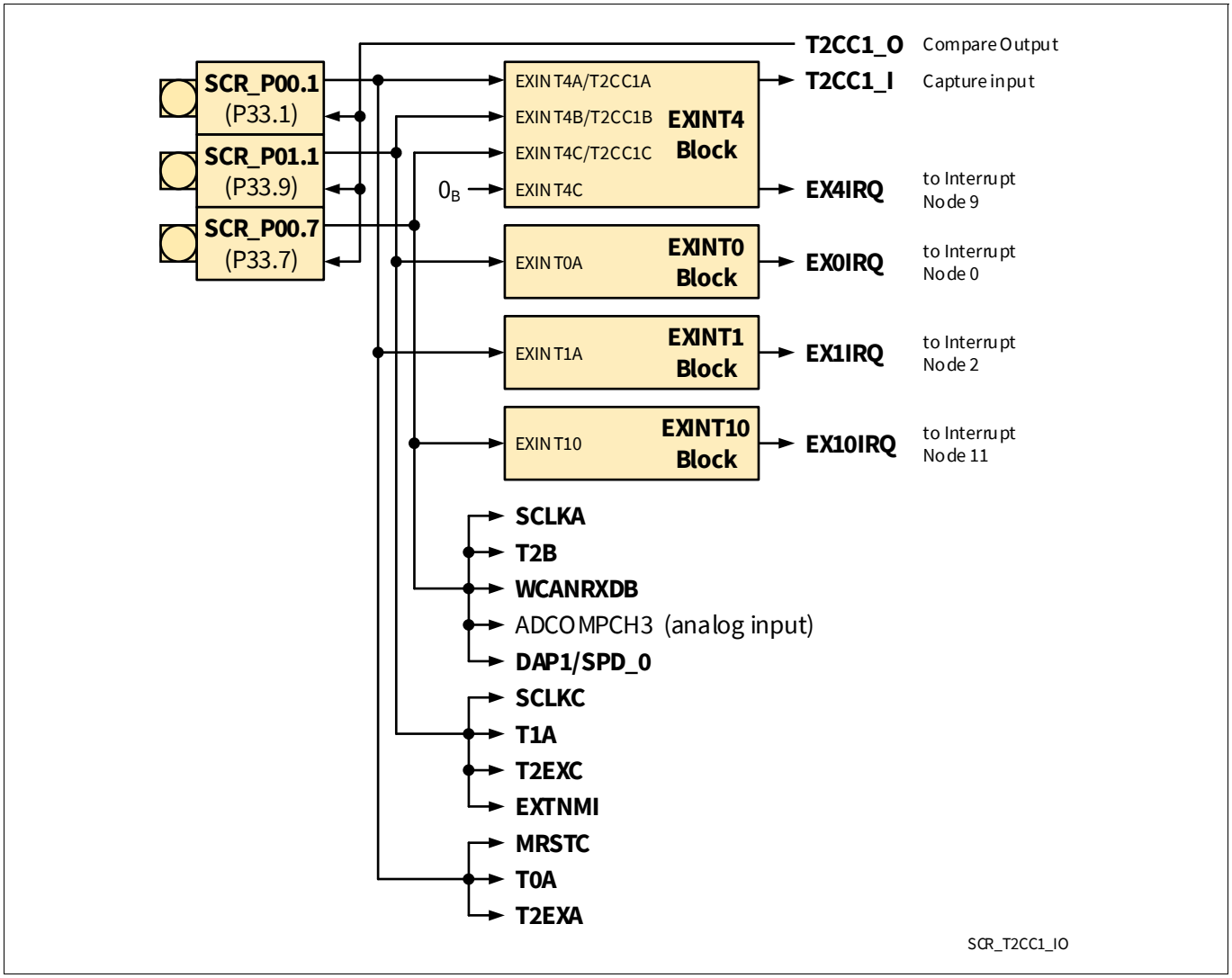


Figure 879 Interrupt and I/O Connections for Capture/Compare Channel 1

49.13.4.4.3 Capture/Compare Channel 2

Figure 880 illustrates the connections of capture/compare channel 2 to the ports, interrupt request blocks, and other SCR modules.

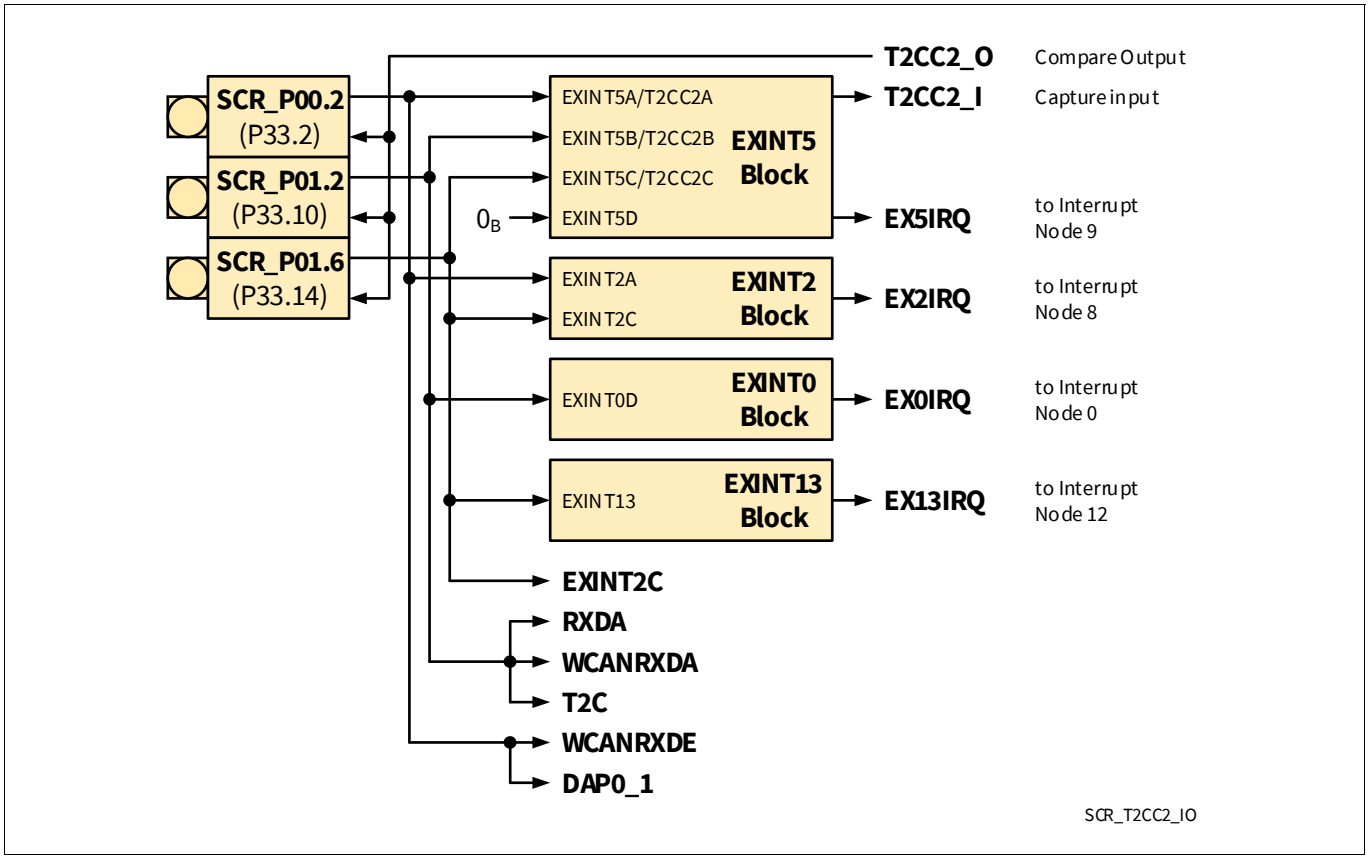


Figure 880 Interrupt and I/O Connections for Capture/Compare Channel 2

49.13.4.4.4 Capture/Compare Channel 3

Figure 881 illustrates the connections of capture/compare channel 3 to the ports, interrupt request blocks, and other SCR modules.

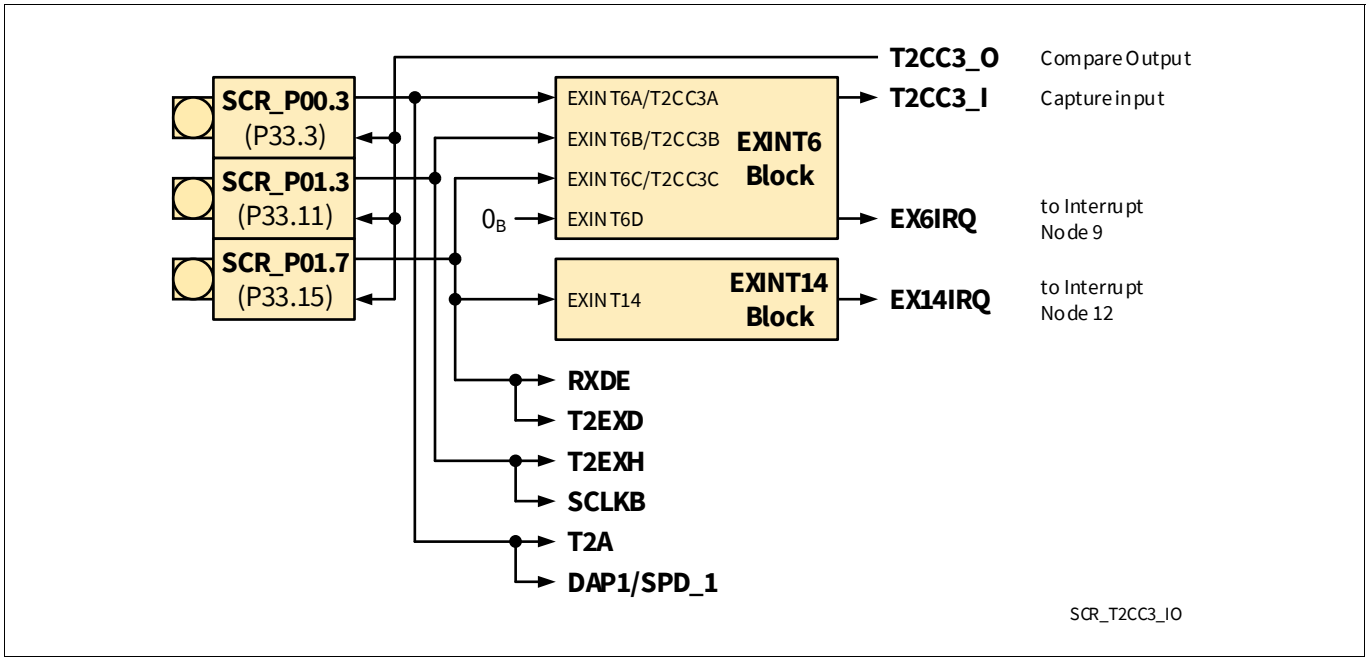


Figure 881 Interrupt and I/O Connections for Capture/Compare Channel 3

49.13.4.4.5 Compare Channel 4

Figure 882 illustrates the connections of compare channel 4 to the ports, interrupt request blocks, and other SCR modules.

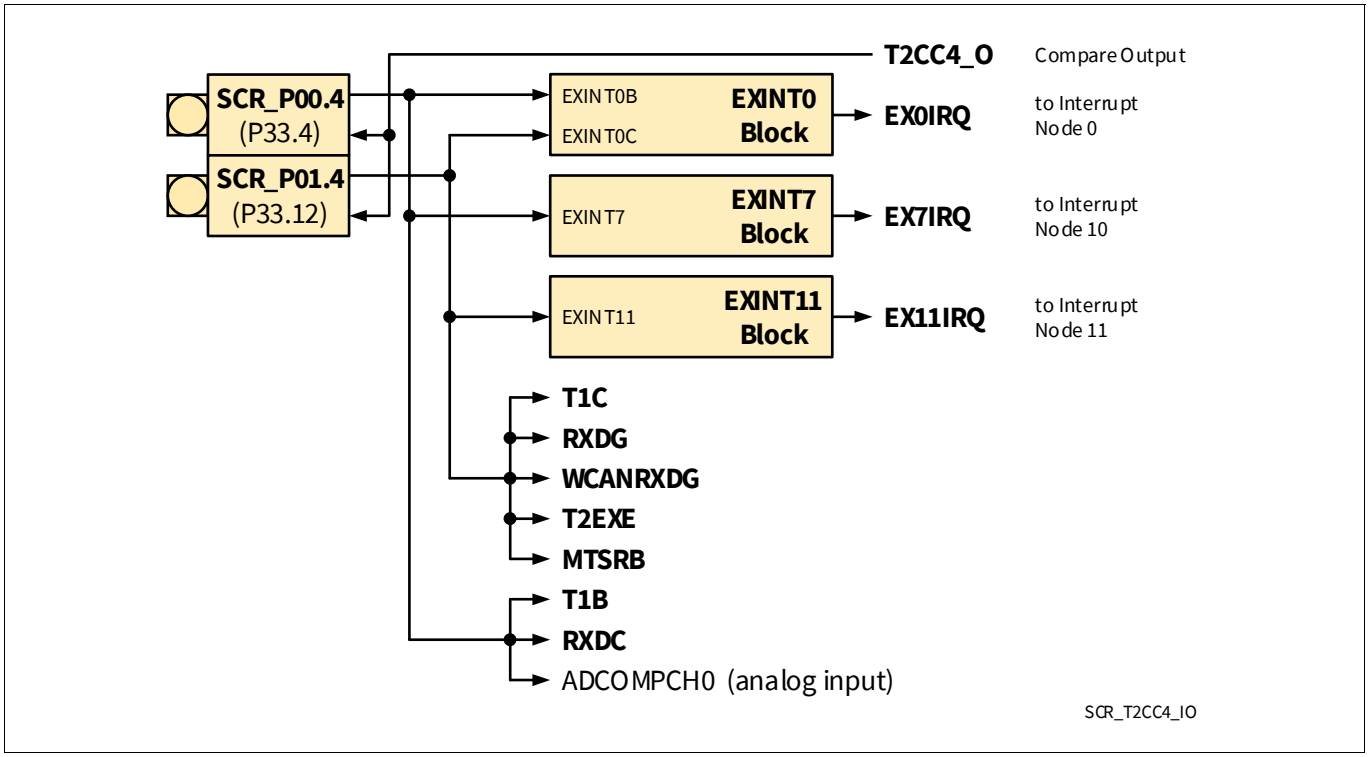


Figure 882 Interrupt and I/O Connections for Compare Channel 4

49.13.4.4.6 Compare Channel 5

Figure 883 illustrates the connections of compare channel 5 to the ports, interrupt request blocks, and other SCR modules.

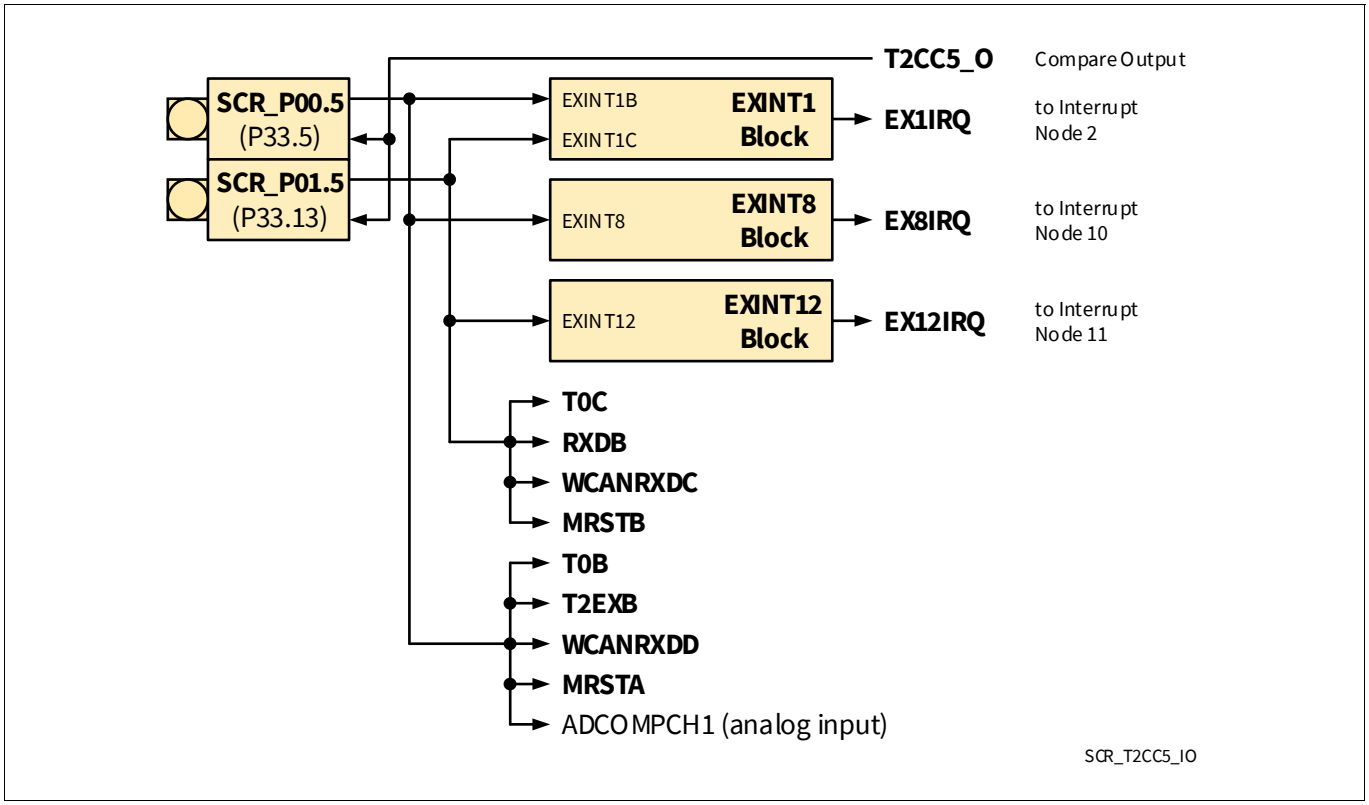


Figure 883 Interrupt and I/O Connections for Compare Channel 5

49.13.5 T2CCU Registers

This section describes all T2CCU special function registers. The Timer 2 and interrupt related SFRs are also listed, as shown in the table below.

Note: All T2CCU register names shall be referenced fully with the module name prefix "T2CCU_".

Table 688 Register Overview - T2 (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
T2CCU_CCEN	T2CCU Capture/Compare Enable Register	0	T2CCU_PAGE=1	$0C0_H$	247
T2CCU_CCTBSEL	T2CCU Capture/Compare Time Base Select Register	0	T2CCU_PAGE=1	$0C1_H$	241
T2CCU_CCTCON	T2CCU Capture/Compare Timer Control Register	0	T2CCU_PAGE=1	$0C6_H$	244
T2CCU_CCTDTCH	T2CCU Capture/Compare Timer Dead-Time Control Register High	0	T2CCU_PAGE=4	$0C3_H$	246
T2CCU_CCTDTCL	T2CCU Capture/Compare Timer Dead-Time Control Register Low	0	T2CCU_PAGE=4	$0C2_H$	246
T2CCU_CCTH	T2CCU Capture/Compare Timer Register High	0	T2CCU_PAGE=1	$0C5_H$	244
T2CCU_CCTL	T2CCU Capture/Compare Timer Register Low	0	T2CCU_PAGE=1	$0C4_H$	243
T2CCU_CCTRELH	T2CCU Capture/Compare Timer Reload Register High	0	T2CCU_PAGE=1	$0C3_H$	242
T2CCU_CCTRELL	T2CCU Capture/Compare Timer Reload Register Low	0	T2CCU_PAGE=1	$0C2_H$	242
T2CCU_CCxH	T2CCU Capture/Compare Register x High	0	T2CCU_PAGE=2	$0C2_H$+x*2	253
T2CCU_CCxH	T2CCU Capture/Compare Register x High	0	T2CCU_PAGE=3	$0C2_H$+(x-3)*2	253
T2CCU_CCxL	T2CCU Capture/Compare Register x Low	0	T2CCU_PAGE=2	$0C1_H$+x*2	251
T2CCU_CCxL	T2CCU Capture/Compare Register x Low	0	T2CCU_PAGE=3	$0C1_H$+(x-3)*2	252
T2CCU_COCON	T2CCU Compare Control Register	0	T2CCU_PAGE=3	$0C0_H$	250
T2CCU_COSHDW	T2CCU Compare Shadow Register	0	T2CCU_PAGE=2	$0C0_H$	248
T2CCU_PAGE	Page Register for T2CCU	0	X	$0C7_H$	240
T2_CON	Timer 2 Control Register	0	T2CCU_PAGE=0	$0C0_H$	208
T2_CON1	Timer 2 Control Register 1	0	T2CCU_PAGE=0	$0C6_H$	209

Table 688 Register Overview - T2 (sorted by Name) (cont'd)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
T2_MOD	Timer 2 Mode Register	0	T2CCU_PAGE=0	$0C1_H$	207
T2_RC2H	Timer 2 Reload/Capture Register, High Byte	0	T2CCU_PAGE=0	$0C3_H$	211
T2_RC2L	Timer 2 Reload/Capture Register, Low Byte	0	T2CCU_PAGE=0	$0C2_H$	210
T2_T2H	Timer 2, High Byte	0	T2CCU_PAGE=0	$0C5_H$	212
T2_T2L	Timer 2, Low Byte	0	T2CCU_PAGE=0	$0C4_H$	211

49.13.5.1 T2CCU Page Register

The BPI of the Timer 2 with T2CCU supports local address extension mechanism. The SFRs of the Timer 2 are accessed in Page 0, while SFRs of the T2CCU are accessed from Page 1 to Page 4.

Page Register for T2CCU

T2CCU_PAGE
Page Register for T2CCU **($0C7_H$)** **Reset Value: Table 689**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
OP		**STNR**		**0**	**PAGE**		
w		w		r	rw		

Field	Bits	Type	Description
PAGE	2:0	rw	**Page Bits** When written, the value indicates the new page address. When read, the value indicates the currently active page = addr[y:x+1]. 000_B **PAGE0**, Selection is PAGE0 001_B **PAGE1**, Selection is PAGE1 010_B **PAGE2**, Selection is PAGE2 011_B **PAGE3**, Selection is PAGE3 100_B **PAGE4**, Selection is PAGE4 101_B **PAGE5**, Selection is PAGE5 110_B **PAGE6**, Selection is PAGE6 111_B **PAGE7**, Selection is PAGE7
STNR	5:4	w	**Storage Number** This number indicates which storage bitfield is the target of the operation defined by bit OP. If OP = 10_B, the contents of PAGE are saved in STx before being overwritten with the new value. If OP = 11_B, the contents of PAGE are overwritten by the contents of STx. The value written to bitfield PAGE is ignored. 00_B **MOD_ST0**, ST0 is selected 01_B **MOD_ST1**, ST1 is selected 10_B **MOD_ST2**, ST2 is selected 11_B **MOD_ST3**, ST3 is selected

Field	Bits	Type	Description
OP	7:6	w	**Operation** 00_B **PAGE_MANUAL0**, Manual page mode. The value of STNR is ignored and PAGE is directly written. 01_B **PAGE_MANUAL1**, Manual page mode. The value of STNR is ignored and PAGE is directly written 10_B **PAGE_SAVE**, New page programming with automatic page saving. The value written to the bit positions of PAGE is stored. In parallel, the former contents of PAGE are saved in the storage bit field STx indicated by STNR. 11_B **PAGE_RESTORE**, Automatic restore page action. The value written to the bit positions PAGE is ignored and instead, PAGE is overwritten by the contents of the storage bit field STx indicated by STNR.
0	3	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 689 Reset Values of T2CCU_PAGE

Reset Type	Reset Value	Note
LVD Reset	0000 $X000_B$	
Generated Reset	0000 $–000_B$	

49.13.5.2 T2CCU Registers Description

The following describes the Special Function Registers of the T2CCU kernel.

T2CCU Capture/Compare Time Base Select Register

The register is addressable via T2CCU_PAGE.

T2CCU_CCTBSEL
T2CCU Capture/Compare Time Base Select Register($0C1_H$) **Reset Value: Table 690**
RMAP: 0, PAGE: T2CCU_PAGE=1

7	6	5	4	3	2	1	0
CASC	**CCTTOV**	**CCTB5**	**CCTB4**	**CCTB3**	**CCTB2**	**CCTB1**	**CCTB0**
rw	rwh	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
CCTBx (x=0-5)	x	rw	**Channel x Time Base Select** 0_B Selects Timer 2 1_B Selects Capture/Compare Timer (CCT)
CCTTOV	6	rwh	**Trigger CCT Timer Overflow Event** On set, this bit is held for one PCLK, then cleared by hardware. Reading this bit always returns 0. 0_B No action 1_B CCT overflow event is triggered immediately

Field	Bits	Type	Description
CASC	7	rw	**Cascade Timers** 0_B Timer 2 and CCT timer operate independently 1_B Timer 2 is cascaded with CCT timer

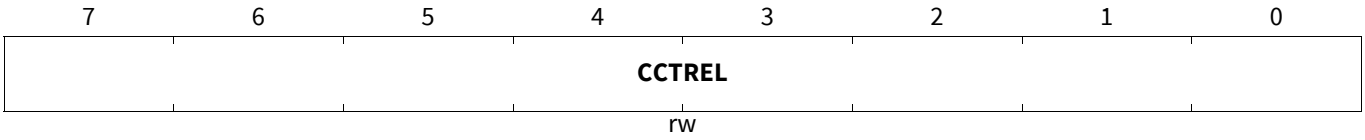
Table 690 Reset Values of T2CCU_CCTBSEL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Timer Reload Register Low

The register is addressable via T2CCU_PAGE.

T2CCU_CCTRELL
T2CCU Capture/Compare Timer Reload Register Low($0C2_H$) **Reset Value: Table 691**
RMAP: 0, PAGE: T2CCU_PAGE=1



Field	Bits	Type	Description
CCTREL	7:0	rw	**Capture/Compare Timer Low Byte Reload Value [7:0]** The contents of the registers are loaded into the CCT timer low byte register upon an overflow condition.

Table 691 Reset Values of T2CCU_CCTRELL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Timer Reload Register High

The register is addressable via T2CCU_PAGE.

T2CCU_CCTRELH
T2CCU Capture/Compare Timer Reload Register High($0C3_H$) **Reset Value: Table 692**
RMAP: 0, PAGE: T2CCU_PAGE=1

7	6	5	4	3	2	1	0
CCTREL							
rw							

Field	Bits	Type	Description
CCTREL	7:0	rw	**Capture/Compare Timer High Byte Reload Value [15:8]** The contents of the registers are loaded into the CCT timer high byte register upon an overflow condition.

Table 692 Reset Values of T2CCU_CCTRELH

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Timer Register Low

The register is addressable via T2CCU_PAGE.

T2CCU_CCTL
T2CCU Capture/Compare Timer Register Low ($0C4_H$) **Reset Value: Table 693**
RMAP: 0, PAGE: T2CCU_PAGE=1

7	6	5	4	3	2	1	0
CCT							
rwh							

Field	Bits	Type	Description
CCT	7:0	rwh	**Capture/Compare Timer Low Byte Value [7:0]** The register represents the low byte counter value of the Capture/Compare Timer. When CCT is not running, a write to CCTL updates the lower byte of timer. To (re-)initialize CCT when it is running, CCTH has to be written first. On write to CCTL, the values CCTH and CCTL will be updated to the timer, which then counts from this value.

Table 693 Reset Values of T2CCU_CCTL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Timer Register High

The register is addressable via T2CCU_PAGE.

T2CCU_CCTH
T2CCU Capture/Compare Timer Register High (0C5$_H$) **Reset Value: Table 694**
RMAP: 0, PAGE: T2CCU_PAGE=1

7	6	5	4	3	2	1	0
CCT							
rwh							

Field	Bits	Type	Description
CCT	7:0	rwh	**Capture/Compare Timer High Byte Value [15:8]** The register represents the high byte counter value of the Capture/Compare Timer. When CCT is not running, a write to CCTH updates the upper byte of timer. To (re-)initialize CCT when it is running, CCTH has to be written first. On write to CCTL, the values CCTH and CCTL will be updated to the timer, which then counts from this value.

Table 694 Reset Values of T2CCU_CCTH

Reset Type	Reset Value	Note
LVD Reset	00$_H$	
Generated Reset	00$_H$	

T2CCU Capture/Compare Timer Control Register

The register is addressable via T2CCU_PAGE.

T2CCU_CCTCON
T2CCU Capture/Compare Timer Control Register(0C6$_H$) **Reset Value: Table 695**
RMAP: 0, PAGE: T2CCU_PAGE=1

7	6	5	4	3	2	1	0
CCTPRE				CCTOVF	CCTOVEN	TIMSYN	CCTST
rw				rwh	rw	rw	rw

Field	Bits	Type	Description
CCTST	0	rw	**Capture/Compare Timer Start/Stop Control** *Note: Setting CCTST while the bit was already set will not disturb the running CCT timer operation.* 0_B Stop Capture/Compare Timer 1_B Start Capture/Compare Timer. If bit TIMSYN = 1 and CCTST was 0, start also Timer 2 (if it was not already running).
TIMSYN	1	rw	**Enable Synchronized Timer Starts** 0_B Disable synchronized timer starts 1_B Enable synchronized timer starts
CCTOVEN	2	rw	**Capture/Compare Timer Overflow Interrupt Enable** Set to enable CCT overflow interrupt.
CCTOVF	3	rwh	**Capture/Compare Timer Overflow Flag** Set by hardware on CCT overflow. This bit has to be cleared by software. *Note: In case of software and hardware access to the flag at the same time, hardware will have higher priority.*
CCTPRE	7:4	rw	**Compare/Compare Timer Prescaler Bitfield** Selects the input clock for CCT, which is derived from the peripheral clock. *Note: It is not recommended to change CCTPRE when CCTST = 1.* 0_H **DIV**, fCCT = fPCLK 1_H **DIV2**, fCCT = fPCLK / 2 2_H **DIV4**, fCCT = fPCLK / 4 3_H **DIV8**, fCCT = fPCLK / 8 4_H **DIV16**, fCCT = fPCLK / 16 5_H **DIV32**, fCCT = fPCLK / 32 6_H **DIV64**, fCCT = fPCLK / 64 7_H **DIV128**, fCCT = fPCLK / 128 8_H **DIV256**, fCCT = fPCLK / 256 9_H **DIV512**, fCCT = fPCLK / 512 A_H **DIV1024**, fCCT = fPCLK / 1024 B_H **DIV2048**, fCCT = fPCLK / 2048

Table 695 Reset Values of T2CCU_CCTCON

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Timer Dead-Time Control Register Low

The register is addressable via T2CCU_PAGE.

T2CCU_CCTDTCL
T2CCU Capture/Compare Timer Dead-Time Control Register Low($0C2_H$) **Reset Value: Table 696**
RMAP: 0, PAGE: T2CCU_PAGE=4

7	6	5	4	3	2	1	0
DTM							
rw							

Field	Bits	Type	Description
DTM	7:0	rw	**Dead-Time Value** This 8-bit value defines the programmable delay between switching from the passive state to the active state of the selected outputs. The switching from the active state to the passive state is not delayed.

Table 696 Reset Values of T2CCU_CCTDTCL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Timer Dead-Time Control Register High

The register is addressable via T2CCU_PAGE.

T2CCU_CCTDTCH
T2CCU Capture/Compare Timer Dead-Time Control Register High($0C3_H$) **Reset Value: Table 697**
RMAP: 0, PAGE: T2CCU_PAGE=4

7	6	5	4	3	2	1	0
DTRES	DTR2	DTR1	DTR0	DTLEV	DTE2	DTE1	DTE0
rwh	rh	rh	rh	rw	rw	rw	rw

Field	Bits	Type	Description
DTEx (x=0-2)	x	rw	**Dead-Time Enable for Channel Pair x** *Note: This bit shall only be set after enabling the corresponding channels in compare mode 0 with CCT timer as the time base.* 0_B Dead-time generation is disabled 1_B Dead-time generation is enabled for channel pair x. The corresponding outputs switch from the passive state to the active state (according to compare status) with the delay programmed in DTM.
DTLEV	3	rw	**Dead-Time Level** This bit defines the active level of all DTCx during dead-time counting. 0_B Low (switches are active high) 1_B High (switches are active low)
DTRx (x=0-2)	x+4	rh	**Dead-Time Run Indication Bits** 0_B The value of the corresponding dead-time counter is 0 1_B The value of the corresponding dead-time counter is not 0
DTRES	7	rwh	**Reset Dead-Time Counters** On set, this bit is held for one PCLK, then cleared by hardware. Reading this bit always returns 0. 0_B No action 1_B The three dead-time counters are reset to 0

Table 697 Reset Values of T2CCU_CCTDTCH

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Enable Register

The register is addressable via T2CCU_PAGE.

Note: The T2CCU output is only actively driven on enabling the compare function.

T2CCU_CCEN
T2CCU Capture/Compare Enable Register ($0C0_H$) Reset Value: Table 698
RMAP: 0, PAGE: T2CCU_PAGE=1

7	6	5	4	3	2	1	0
CCM3		**CCM2**		**CCM1**		**CCM0**	
rw		rw		rw		rw	

Field	Bits	Type	Description
CCM0	1:0	rw	**Capture/Compare Enable for Channel 0** 00_B **CC_DIS**, Capture/Compare disabled 01_B **CAPTURE_ACT_EDGE**, Capture on active edge at pin T2CC0 10_B **COMP_EN**, Compare enabled 11_B **CAPTURE_ON_WRITE**, Capture on write operation into register CC0L
CCM1	3:2	rw	**Capture/Compare Enable for Channel 1** 00_B **CC_DIS**, Capture/Compare disabled 01_B **CAPTURE_ACT_EDGE**, Capture on active edge at pin T2CC1 10_B **COMP_EN**, Compare enabled 11_B **CAPTURE_ON_WRITE**, Capture on write operation into register CC1L
CCM2	5:4	rw	**Capture/Compare Enable for Channel 2** 00_B **CC_DIS**, Capture/Compare disabled 01_B **CAPTURE_ACT_EDGE**, Capture on active bedge at pin T2CC2 10_B **COMP_EN**, Compare enabled 11_B **CAPTURE_ON_WRITE**, Capture on write operation into register CC2L
CCM3	7:6	rw	**Capture/Compare Enable for Channel 3** 00_B **CC_DIS**, Capture/Compare disabled 01_B **CAPTURE_ACT_EDGE**, Capture on active edge at pin T2CC3 10_B **COMP_EN**, Compare enabled 11_B **CAPTURE_ON_WRITE**, Capture on write operation into register CC3L

Table 698 Reset Values of T2CCU_CCEN

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Compare Shadow Register

The register is addressable via T2CCU_PAGE.

T2CCU_COSHDW
T2CCU Compare Shadow Register **($0C0_H$)** **Reset Value: Table 699**
RMAP: 0, PAGE: T2CCU_PAGE=2

7	6	5	4	3	2	1	0
ENSHDW	**TXOV**	**COOUT5**	**COOUT4**	**COOUT3**	**COOUT2**	**COOUT1**	**COOUT0**
rwh	rw	rwh	rwh	rwh	rwh	rwh	rwh

Field	Bits	Type	Description
COOUTx (x=0-5)	x	rwh	**Compare Channel x Output** ENSHDW = 0: Any access to bit COOUTx accesses the corresponding bit of the internal 6-bit compare output register. Writing to bit COOUTx will change the corresponding bit of the internal compare output register. Reading from bit COOUTx returns the value of the bit in the internal compare output register. The content of the internal compare output register is output on the corresponding T2CCx pins when a valid compare match occurs for a channel in compare mode 1 or concurrent compare mode. *Note: A concurrent write operation to clear bit ENSHDW and set/clear bit(s) COOUTx is effective for initializing the internal compare output register.* ENSHDW = 1: In this case, accessing bit COOUTx accesses the actual SFR bits. If COOUTx is set, the contents of register CCx will be transferred to corresponding channel x internal compare register. On set, COOUTx is held for one PCLK, then cleared automatically by hardware. *Note: A concurrent write operation to set bit ENSHDW and bit(s) COOUTx (and clear bit TXOV) is effective for starting shadow transfer(s).*
TXOV	6	rw	**Enable Shadow Transfer On Timer Overflow** Set to enable for all compare channels, transfer of shadow value in T2CCU_CCx to the corresponding internal compare registers on overflow of their respective timer base.
ENSHDW	7	rwh	**Enable Shadow Transfer With COOUTx** Clear to enable access of the internal compare output register via the COOUTx bits. Set to enable use of COOUTx bits for transfer of shadow value in T2CCU_CCx to the corresponding channel x internal compare register. *Note: When setting bit TXOV, bit ENSHDW will be cleared by internal hardware. To ensure bit ENSHDW is set, user must ensure bit TXOV is cleared.*

Table 699 Reset Values of T2CCU_COSHDW

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Compare Control Register

The register is addressable via T2CCU_PAGE.

T2CCU_COCON
T2CCU Compare Control Register **($0C0_H$)** **Reset Value: Table 700**
RMAP: 0, PAGE: T2CCU_PAGE=3

7	6	5	4	3	2	1	0
CCM5	**CCM4**	**CM5F**	**CM4F**	**POLB**	**POLA**	**COMOD**	
rw	rw	rwh	rwh	rw	rw	rw	

Field	Bits	Type	Description
COMOD	1:0	rw	**Compare Mode Control (for compare channels)** 00_B Compare Mode 0 selected 01_B Compare Mode 1 selected 10_B Concurrent Compare Mode selected 11_B Reserved
POLA	2	rw	**Compare Active Level for Channel Group A** In compare mode 0, the bit defines the active level of the outputs of the compare channels belonging to the group. 0_B High on Compare Match 1_B Low on Compare Match
POLB	3	rw	**Compare Active Level for Channel Group B** In compare mode 0, the bit defines the active level of the outputs of the compare channels belonging to the group. 0_B High on Compare Match 1_B Low on Compare Match
CM4F	4	rwh	**Compare Channel 4 Interrupt Flag** Set on a compare match event on compare channel 4. This bit has to be cleared by software. *Note: In case of software and hardware access to a flag bit at the same time, hardware will have higher priority.*
CM5F	5	rwh	**Compare Channel 5 Interrupt Flag** Set on a compare match event on compare channel 5. This bit has to be cleared by software. *Note: In case of software and hardware access to a flag bit at the same time, hardware will have higher priority.*

Field	Bits	Type	Description
CCM4	6	rw	**Compare Enable for Channel 4** *Note: Compare channels 4 and 5 should be enabled in a separate step after setting the compare mode via bits COMOD (and active level POLA/B if compare mode 0). This is to ensure defined compare output according to the selected mode on starting the respective compare channel.* *Note: The interrupt enable for compare channel 4 is controlled by the respective bit of SCU SFR MODIEN.CM4EN.* 0_B Compare disabled 1_B Compare enabled
CCM5	7	rw	**Compare Enable for Channel 5** *Note: Compare channels 4 and 5 should be enabled in a separate step after setting the compare mode via bits COMOD (and active level POLA/B if compare mode 0). This is to ensure defined compare output according to the selected mode on starting the respective compare channel.* *Note: The interrupt enable for compare channel 5 is controlled by the respective bit of SCU SFR MODIEN.CM5EN.* 0_B Compare disabled 1_B Compare enabled

Table 700 Reset Values of T2CCU_COCON

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Register x Low

The register is addressable via T2CCU_PAGE.

T2CCU_CCxL (x=0-2)
T2CCU Capture/Compare Register x Low ($0C1_H$+x*2) Reset Value: Table 701
RMAP: 0, PAGE: T2CCU_PAGE=2

7	6	5	4	3	2	1	0
CCVALL							
rwh							

Field	Bits	Type	Description
CCVALL	7:0	rwh	**Capture/Compare Low Byte Value for Channel x** Capture: On read, CCVALL returns the low byte of the last captured value. Compare: CCVALL holds the low byte written. CCx acts as a shadow register to the channel x internal compare register. The value written is only transferred to the internal compare register when the corresponding bit T2CCU_COSHDW.COOUTx is being set (with ENSHDW = 1), or on overflow of the channel timer base. The read value always return the last written value.

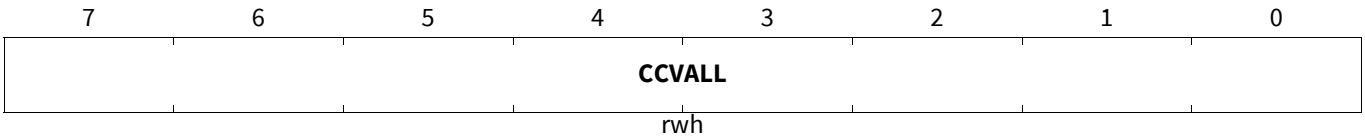
Table 701 Reset Values of T2CCU_CCxL (x=0-2)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Register x Low

The register is addressable via T2CCU_PAGE.

T2CCU_CCxL (x=3-5)
T2CCU Capture/Compare Register x Low **($0C1_H$+(x-3)*2)** **Reset Value: Table 702**
RMAP: 0, PAGE: T2CCU_PAGE=3



Field	Bits	Type	Description
CCVALL	7:0	rwh	**Capture/Compare Low Byte Value for Channel x** Capture: On read, CCVALL returns the low byte of the last captured value. Compare: CCVALL holds the low byte written. CCx acts as a shadow register to the channel x internal compare register. The value written is only transferred to the internal compare register when the corresponding bit T2CCU_COSHDW.COOUTx is being set (with ENSHDW = 1), or on overflow of the channel timer base. The read value always return the last written value. *Note: For T2CCU_CC4L/H and T2CCU_CC5L/H, only the compare function is available.*

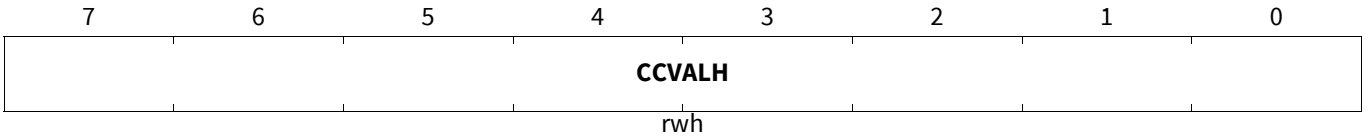
Table 702 Reset Values of T2CCU_CCxL (x=3-5)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Register x High

The register is addressable via T2CCU_PAGE.

T2CCU_CCxH (x=0-2)
T2CCU Capture/Compare Register x High ($0C2_H$+x*2) Reset Value: Table 703
RMAP: 0, PAGE: T2CCU_PAGE=2



Field	Bits	Type	Description
CCVALH	7:0	rwh	**Capture/Compare High Byte Value for Channel x** Capture: On read, CCVALH returns the high byte of the last captured value. Compare: CCVALH holds the high byte written. CCx acts as a shadow register to the channel x internal compare register. The value written is only transferred to the internal compare register when the corresponding bit T2CCU_COSHDW.COOUTx is being set (with ENSHDW = 1), or on overflow of the channel timer base. The read value always return the last written value.

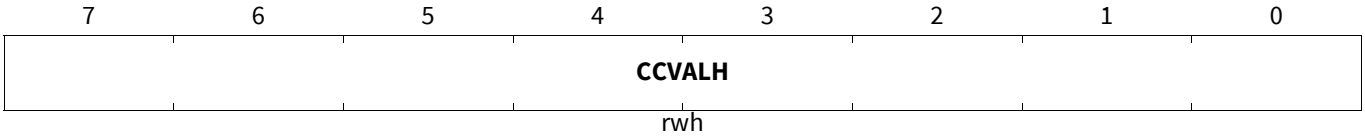
Table 703 Reset Values of T2CCU_CCxH (x=0-2)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

T2CCU Capture/Compare Register x High

The register is addressable via T2CCU_PAGE.

T2CCU_CCxH (x=3-5)
T2CCU Capture/Compare Register x High **($0C2_H$+(x-3)*2)** **Reset Value: Table 704**
RMAP: 0, PAGE: T2CCU_PAGE=3



Field	Bits	Type	Description
CCVALH	7:0	rwh	**Capture/Compare High Byte Value for Channel x** Capture: On read, CCVALH returns the high byte of the last captured value. Compare: CCVALH holds the high byte written. CCx acts as a shadow register to the channel x internal compare register. The value written is only transferred to the internal compare register when the corresponding bit T2CCU_COSHDW.COOUTx is being set (with ENSHDW = 1), or on overflow of the channel timer base. The read value always return the last written value. *Note: For T2CCU_CC4L/H and T2CCU_CC5L/H, only the compare function is available.*

Table 704 Reset Values of T2CCU_CCxH (x=3-5)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.13.6 T2CCU Clocking Configuration

If the T2CCU functionality is not required at all, it can be completely disabled by gating off its clock input for maximal power reduction. This is done by setting bit T2CCU_DIS in register PMCON1 (in SCU_PAGE) as described below.

Peripheral Management Control Register 1

SCU_PMCON1
Peripheral Management Control Register 1 ($0FB_H$) Reset Value: Table 705
RMAP: 0, PAGE: SCU_PAGE=1

7	6	5	4	3	2	1	0
0	**OCDS_DIS**	**LIN_DIS**	**WDT_DIS**	**WCAN_DIS**	**RTC_DIS**	**T2CCU_DIS**	**SSC_DIS**
r	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
SSC_DIS	0	rw	**SSC Disable Request (active high)** 0_B SSC is in normal operation 1_B Request to disable the SSC (default)
T2CCU_DIS	1	rw	**T2CCU Disable Request (active high)** 0_B T2CCU is in normal operation 1_B Request to disable the T2CCU (default)
RTC_DIS	2	rw	**RTC Disable Request (active high)** 0_B RTC is in normal operation 1_B Request to disable the RTC (default)
WCAN_DIS	3	rw	**WCAN Disable Request (active high)** 0_B WCAN is in normal operation 1_B Request to disable the WCAN (default)
WDT_DIS	4	rw	**Watchdog Disable Request (active high)** 0_B Watchdog is in normal operation 1_B Request to disable the Watchdog (default)
LIN_DIS	5	rw	**LIN Disable Request (active high)** 0_B LIN baudrate generator and detector are in normal operation 1_B Request to disable the LIN baudrate generator and detector (default)
OCDS_DIS	6	rw	**OCDS Disable Request (active high)** 0_B OCDS and Debug System is in normal operation (this is the default value, and the clock to OCDS and Debug System is enabled by default) 1_B Request to disable the OCDS and Debug System (during normal operation, application may choose to gate the OCDS and Debug System clock to save power)
0	7	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 705 Reset Values of SCU_PMCON1

Reset Type	Reset Value	Note
LVD Reset	X011 1111_B	
Generated Reset	–011 1111_B	

49.13.7 Implementation Details of T2CCU

This section describes the implementation details of the T2CCU module.

49.13.7.1 Interfaces of the T2CCU

Figure 884 gives an overview of the interconnections of the T2CCU.

The interrupt lines of the T2CCU are connected to the CPU interrupt controller.

The General Purpose IO (GPIO) Ports provide the interface from the T2CCU to the external world.

The T2CCU timers can be suspendedwhen OCDS enters Monitor Mode and has the Debug-Suspend signal activated, provided the respective timer suspend bits, T2SUSP and CCTSUSP (in register DBG_MODSUSP), are set. Refer to the OCDS chapter.

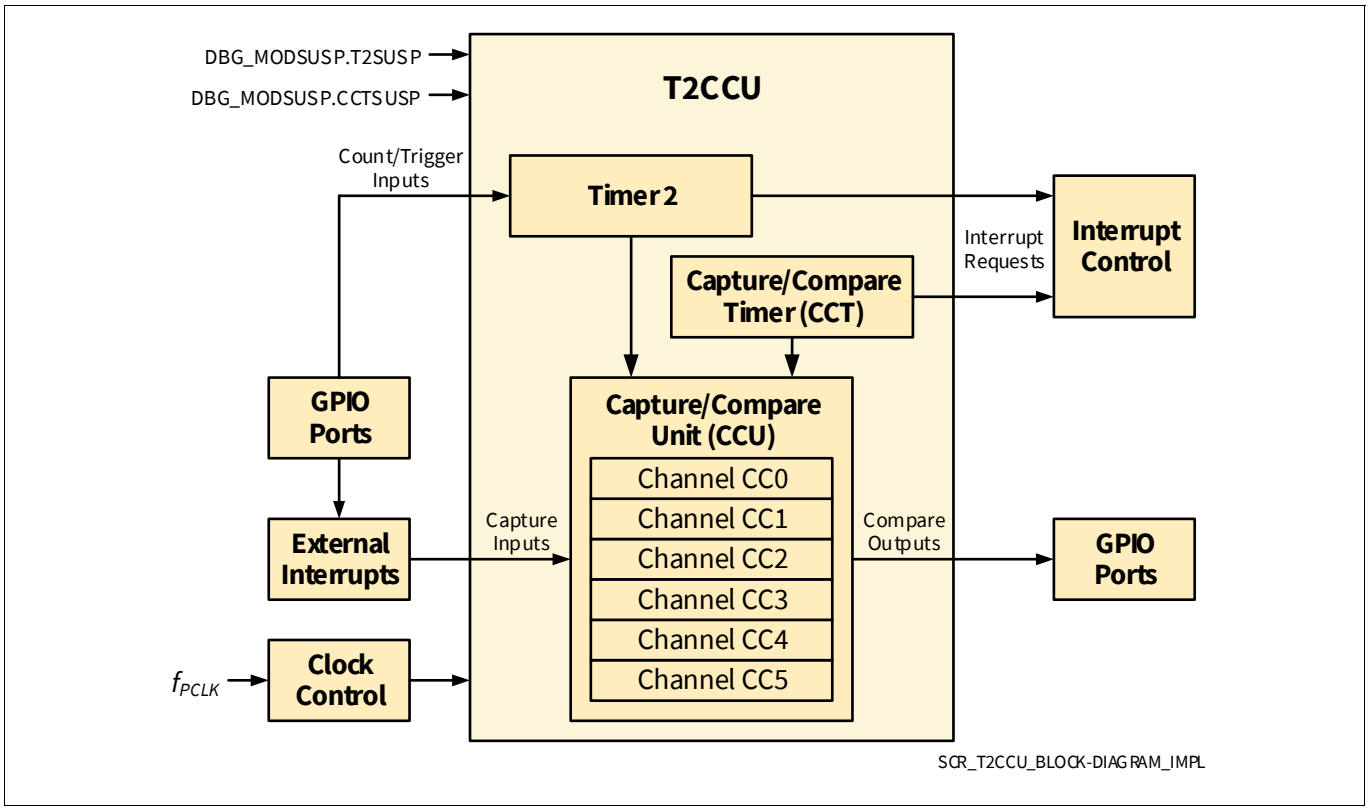


Figure 884 Interconnections of T2CCU

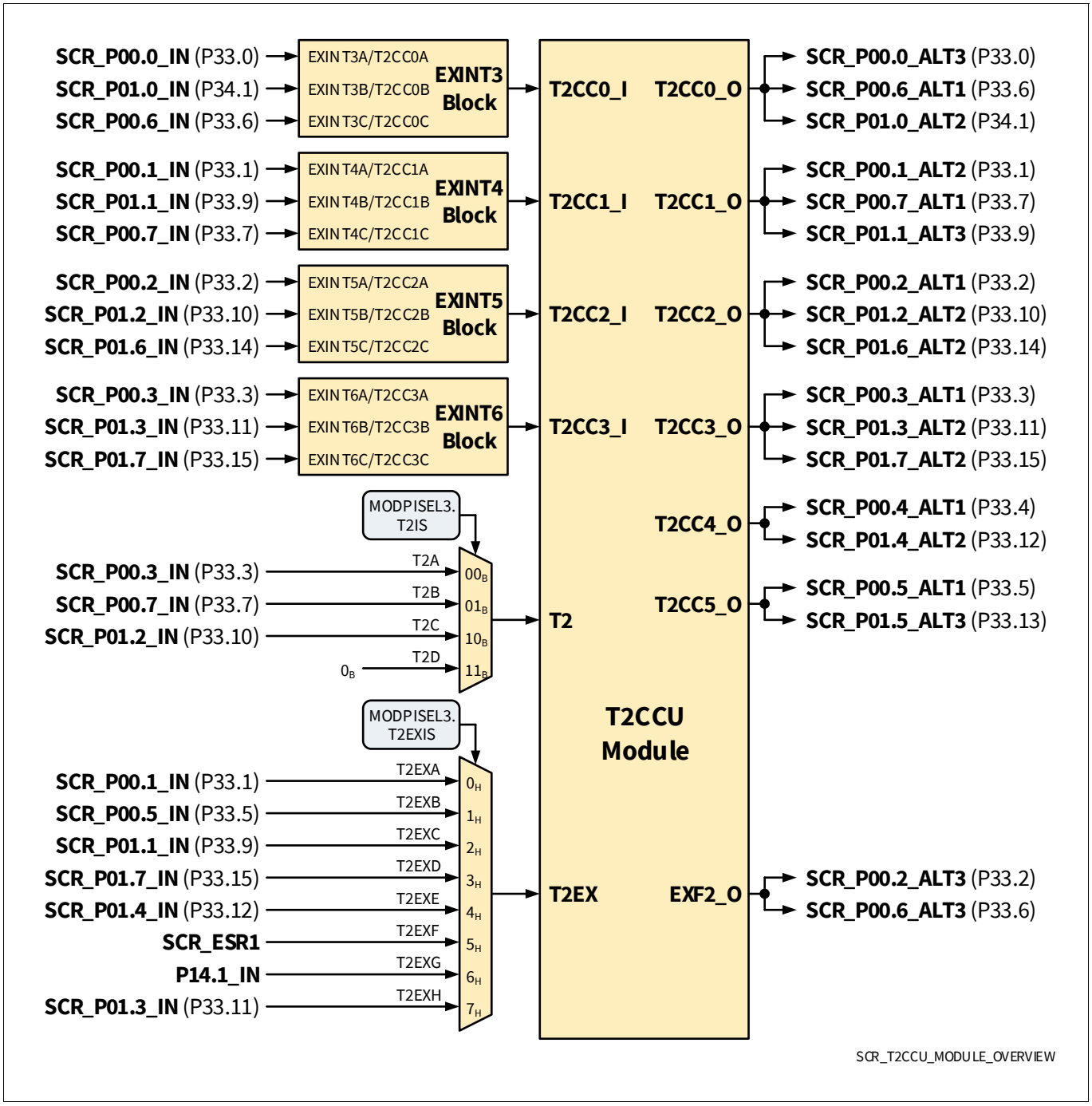


Figure 885 T2CCU Connections to the GPIO Ports

49.13.7.1.1 Interrupt Events and Assignment

Table 706 lists the interrupt event sources from the T2CCU, and the corresponding event interrupt enable bit and flag bit.

Table 706 T2CCU Interrupt Events

Event	Event Interrupt Enable Bit	Event Flag Bit
Timer 2 Overflow	T2_CON1.TF2EN	T2_CON.TF2

Table 706 T2CCU Interrupt Events (cont'd)

Event	Event Interrupt Enable Bit	Event Flag Bit
Timer 2 External Input	T2_CON1.EXF2EN	T2_CON.EXF2
CCT Timer Overflow	T2CCU_CCTCON.CCTOVEN	T2CCU_CCTCON.CCTOVF

Table 707 shows the interrupt node assignment for each T2CCU interrupt source.

Table 707 T2CCU Events' Interrupt Node Control

Event	Interrupt Node Enable Bit	Interrupt Node Flag Bit	Vector Address
Timer 2 Overflow	IEN0.ET2	–	$2B_H$
Timer 2 External Input			
CCT Timer Overflow			

49.13.7.1.2 Module Suspend Control

When the On-Chip Debug Support (OCDS) is in Monitor Mode (DBG_MMCR2.MMODE = 1) and the Debug-Suspend signal is active (DBG_MMCR2.DSUSP = 1), the timers in the T2CCU module can be suspended based on the settings of their corresponding module suspend bits in register DBG_MODSUSP. When suspended, only the timer stops counting, since the counter input clock is gated off. The module is still clocked, so that module registers are accessible.

49.13.8 Revision History

Table 708 Revision History

Reference	Change to Previous Version	
v4.4		
	First Official Release of completely reworked SCR chapter	
v4.5		
Page 221, **Page 225**	Detailled description for first timer period after initialization in compare mode; added notes to dead-time operation	
Figure 870, **Figure 873**, **Figure 874**	Corrected figures concerning internal and shadow compare register (naming was reversed)	

49.14 Universal Asynchronous Receiver/Transmitter (UART)

49.14.1 Overview

The UART provides a full-duplex asynchronous receiver/transmitter, i.e., it can transmit and receive simultaneously. It is also receive-buffered, i.e., it can commence reception of a second byte before a previously received byte has been read from the receive register. However, if the first byte still has not been read by the time reception of the second byte is complete, one of the bytes will be lost.

Note: A module cycle of the UART consists of 2 PCLK cycles.

Note: In this chapter, the input and output signals from/to the GPIO ports are referred to with their generic name, such as TXD and RXD. However, the actual input and output signals use separate lines from/to the GPIO, since they are connected in different ways to the pin logic (pin input line; alternate data output line). Only where explicitly applicable, the generic signal names are extended with an "_I" (input line) or "_O" (output line) identifier.

UART Features:

- Full-duplex asynchronous modes
  - 8-bit or 9-bit data frames, LSB first
  - fixed or variable baudrate
- Receive buffered
- Multiprocessor communication
- Interrupt generation on the completion of a data transmission or reception

49.14.2 UART Modes

The UART can be used in four different modes. In mode 0, it operates as an 8-bit shift register. In mode 1, it operates as an 8-bit serial port. In modes 2 and 3, it operates as a 9-bit serial port. The only difference between mode 2 and mode 3 is the baudrate, which is fixed in mode 2 but variable in mode 3. The variable baudrate is set by the underflow rate on the dedicated baudrate generator.

The different modes are selected by setting bits SM0 and SM1 to their corresponding values, as shown in **Table 709**.

Table 709 UART Modes

SM0	SM1	Operating Mode	Baudrate
0	0	Mode 0: 8-bit shift register	$f_{PCLK}/2$
0	1	Mode 1: 8-bit shift UART	Variable
1	0	Mode 2: 9-bit shift UART	$f_{PCLK}/64$ or $f_{PCLK}/32$
1	1	Mode 3: 9-bit shift UART	Variable

49.14.2.1 Mode 0, 8-Bit Shift Register, Fixed Baudrate

In mode 0, the serial port behaves as an 8-bit shift register. Data is shifted in through RXD, and out through RXDO, while the TXD line is used to provide a shift clock which can be used by external devices to clock data in and out.

The transmission cycle is activated by a write to SBUF. One module cycle later, the data has been written to the transmit shift register with a 1 at the 9th bit position. For the next seven module cycles, the contents of the

transmit shift register are shifted right one position and a zero shifted in from the left so that when the MSB of the data byte is at the output position, it has a 1 and a sequence of zeros to its left. The control block then executes one last shift beforesetting the TI bit. If SCON.TI is set, the next data can be written to SBUF, without destroying the ongoing transmission. If SCON.TI is not set, the ongoing transmission maybe corrupted.

Reception is started by the condition REN = 1 and RI = 0. At the start of the reception cycle, 11111110_B is written to the receive shift register. In each module cycle that follows, the contents of the shift register are shifted left one position and the value sampled on the RXD line in the same module cycle is shifted in from the right. When the 0 of the initial byte reaches the leftmost position, the control block executes one last shift, loads SBUF and sets the RI bit.

The baudrate for the transfer is fixed at $f_{PCLK}/2$, where f_{PCLK} is the input clock frequency, i.e. one bit per module cycle.

Note: The bit SCON.RI is set at the beginning of the 8th bit.

49.14.2.2 Mode 1, 8-Bit UART, Variable Baudrate

In mode 1, the UART behaves as an 8-bit serial port. A start bit (0), 8 data bits, and a stop bit (1) are transmitted on TXD or received on RXD at a variable baudrate.

The transmission cycle is activated by a write to SBUF. The data is transferred to the transmit register and a 1 is loaded to the 9th bit position (as in mode 0). At phase 1 of the module cycle after the next rollover in the divide-by-16 counter, the start bit is copied to TXD, and data is activated one bit time later. One bit time after the data is activated, the data starts getting shifted right with zeros shifted in from the left. When the MSB gets to the output position, the control block executes one last shift and sets the TI bit.

Reception is started by a high to low transition on RXD (sampled at 16 times the baudrate). The divide-by-16 counter is then reset and $1111\ 1111_B$ is written to the receive register. If a valid start bit (0) is then detected (based on two out of three samples), it is shifted into the register followed by 8 data bits. If the transition is not followed by a valid start bit, the controller goes back to looking for a high to low transition on RXD. When the start bit reaches the leftmost position, the control block executes one last shift, then loads SBUF with the 8 data bits, loads RB8 (SCON.2) with the stop bit, and sets the RI bit, provided RI = 0, and either SM2 = 0 (see **Section 49.14.3**) or the received stop bit = 1. If none of these conditions is met, the received byte is lost.

The associated timings for transmit/receive in mode 1 are illustrated in **Figure 886**.

Note: The bit SCON.RI is set at the end of the 8th bit.

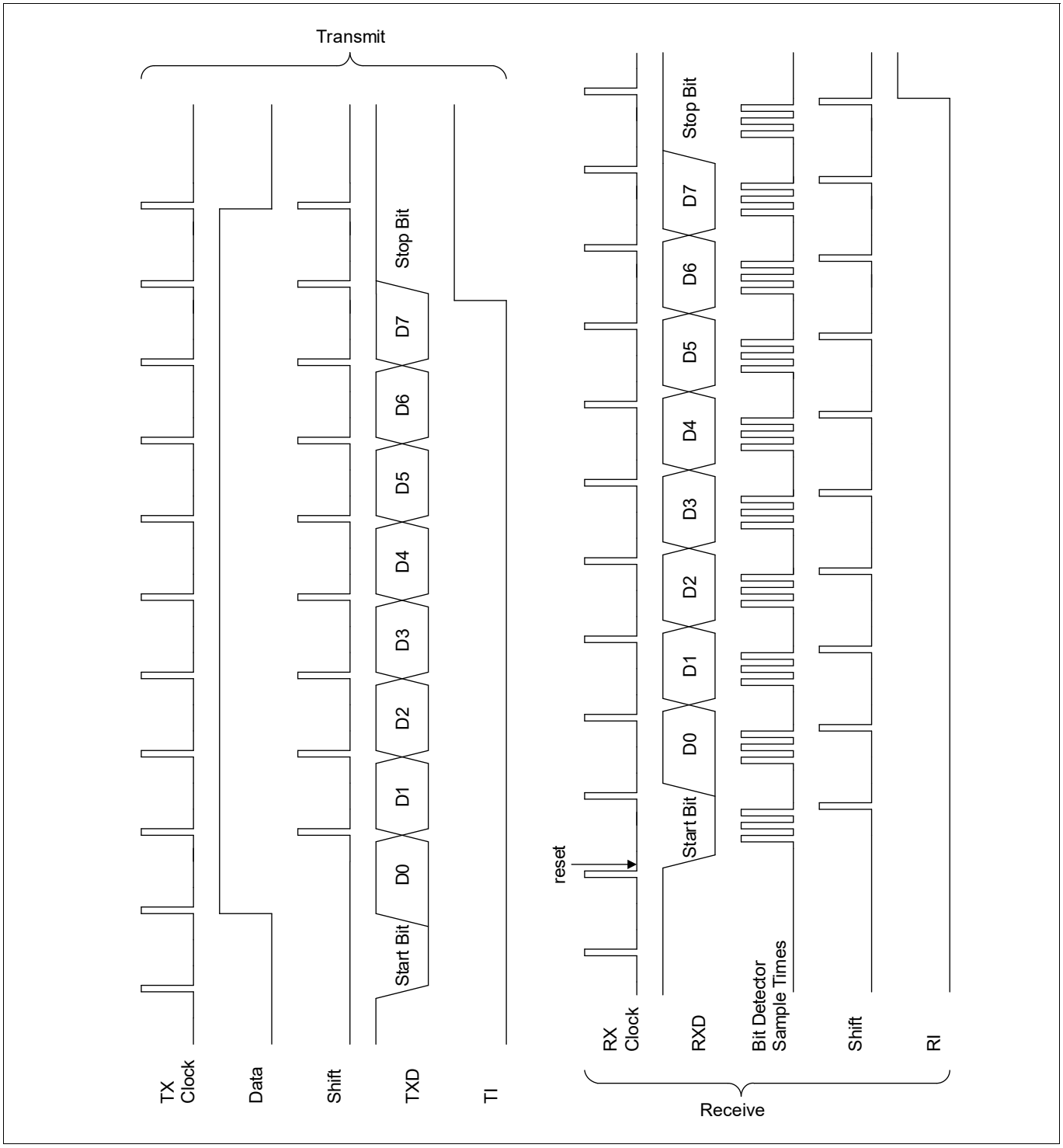


Figure 886 Serial Interface, Mode 1, Timing Diagram

49.14.2.3 Mode 2, 9-Bit UART, Fixed Baudrate

In mode 2, the UART behaves as a 9-bit serial port. A start bit (0), 8 data bits plus a programmable 9th bit and a stop bit (1) are transmitted on TXD or received on RXD. The 9th bit for transmission is taken from TB8 (SCON.3) while for reception, the 9th bit received is placed in RB8 (SCON.2).

The transmission cycle is activated by a write to SBUF. The data is transferred to the transmit register and TB8 is copied into the 9th bit position. At phase 1 of the module cycle following the next rollover in the divide-by-16 counter, the start bit is copied to TXD and data is activated one bit time later. One bit time after the data is activated, the data starts shifting right. For the first shift, a stop bit (1) is shifted in from the left and for subsequent shifts, zeros are shifted in. When the TB8 bit gets to the output position, the control block executes one last shift and sets the TI bit.

Reception is started by a high to low transition on RXD (sampled at 16 times the baudrate). The divide-by-16 counter is then reset and $1111\ 1111_B$ is written to the receive register. If a valid start bit (0) is then detected (based on two out of three samples), it is shifted into the register followed by 8 data bits. If the transition is not followed by a valid start bit, the controller goes back to looking for a high to low transition on RXD. When the start bit reaches the leftmost position, the control block executes one last shift, then loads SBUF with the 8 data bits, loads RB8 (SCON.2) with the 9th data bit, and sets the RI bit, provided RI = 0, and either SM2 = 0 (see **Section 49.14.3**) or the 9th bit = 1. If none of these conditions is met, the received byte is lost.

The baudrate for the transfer is either $f_{PCLK}/64$ or $f_{PCLK}/32$, depending on the setting of the top bit (SMOD) of the PCON (Power Control) register, which acts as a Double Baudrate selector.

Note: The bit SCON.RI is set at the end of the 9th bit.

49.14.2.4 Mode 3, 9-Bit UART, Variable Baudrate

Mode 3 is the same as mode 2 in all respects except that the baudrate is variable.

In all modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in the modes by the incoming start bit if REN = 1.

The serial interface also provides interrupt requests when transmission or reception of the frames has been completed. The corresponding interrupt request flags are TI or RI, respectively. If the serial interrupt is not used (i.e., serial interrupt not enabled), TI and RI can also be used for polling the serial interface.

The associated timings for transmit/receive in modes 2 and 3 are illustrated in **Figure 887**.

Note: The bit SCON.RI is set at the end of the 9th bit.

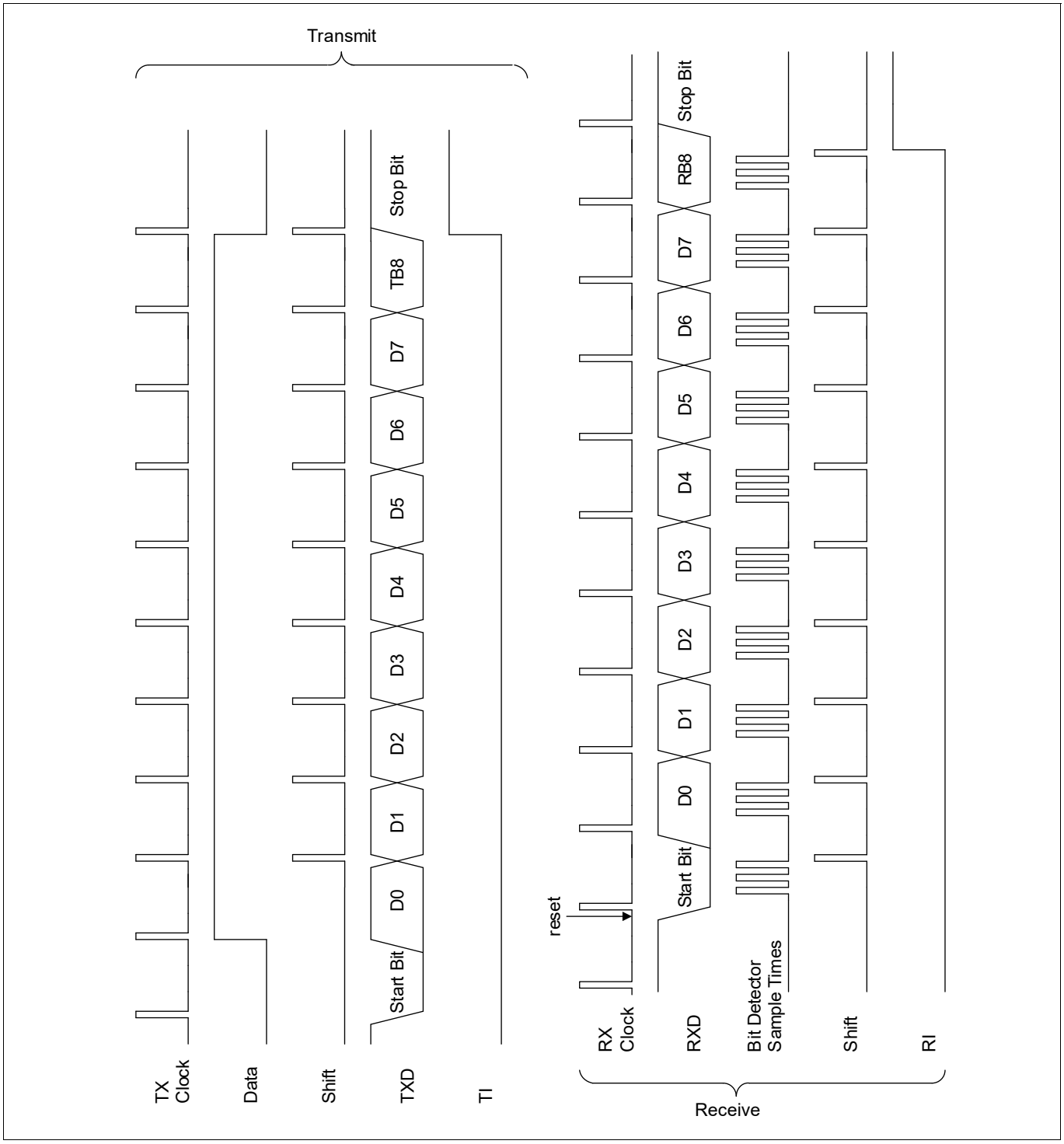


Figure 887 Serial Interface, Modes 2 and 3, Timing Diagram

49.14.3 Multiprocessor Communication

Modes 2 and 3 have a special provision for multiprocessor communication using a system of address bytes with bit 9 = 1 and data bytes with bit 9 = 0. In these modes, 9 data bits are received. The 9th data bit goes into RB8. The communication always ends with one stop bit. The port can be programmed such that when the stop bit is received, the serial port interrupt will be activated only if RB8 = 1.

This feature is enabled by setting bit SM2 in SCON. One of the ways to use this feature in multiprocessor systems is described in the following paragraph.

When the master processor wants to transmit a block of data to one of several slaves, it first sends out an address byte that identifies the target slave. An address byte differs from a data byte in that the 9th bit is 1 in an address byte and 0 in a data byte. With SM2 = 1, no slave will be interrupted by a data byte. An address byte, however, will interrupt all slaves, so that each slave can examine the received byte and see if it is being addressed. The addressed slave will clear its SM2 bit and prepare to receive the data bytes that will be coming. The slaves that were not being addressed retain their SM2s as set and ignore the incoming data bytes.

Bit SM2 has no effect in mode 0. SM2 can be used in mode 1 to check the validity of the stop bit. In a mode 1 reception, if SM2 = 1, the receive interrupt will not be activated unless a valid stop bit is received.

49.14.4 Baudrate Generation

There are several ways to generate the baudrate clock for the serial ports, depending on the mode in which they are operating.

The baudrates in modes 0 and 2 are fixed, so they use the fixed clock, (see **Section 49.14.4.1**). In modes 1 and 3, the variable baudrate is generated using the UART baudrate generator (see **Section 49.14.4.2**).

“Baudrate clock” and “baudrate” must be distinguished from each other. The serial interface requires a clock rate that is 16 times the baudrate, to accommodate for the 16-times oversampling on reception. Therefore, the UART baudrate generator must provide a “baudrate clock” to the serial interface, where it is divided by 16 to obtain the actual “baudrate”. The abbreviation f_{PCLK} refers to the input clock frequency.

49.14.4.1 Fixed Clock

The baudrates in modes 0 and 2 are fixed. However, while the baudrate in mode 0 can only be $f_{PCLK}/2$, the baudrate in mode 2 can be selected as either $f_{PCLK}/64$ or $f_{PCLK}/32$ depending on bit SMOD in the PCON register.

Bit SMOD acts as a double baudrate selector, as shown in **Figure 888**. If SMOD = 0 (value after reset), the baudrate is 1/64 of the input clock frequency f_{PCLK}. If SMOD = 1, the baudrate is 1/32 of f_{PCLK}.

$$\text{Mode 2 Baudrate} = \frac{2^{SMOD}}{64} \times f_{PCLK}$$

uart_baudrate_formula_fixed.vsd

Figure 888 Mode 2 Baudrate

49.14.4.2 UART Baudrate Generator

The UART baudrate generator is used to generate the variable baudrate for the UART in modes 1 and 3. It has programmable 11-bit reload value, 3-bit prescaler, and 5-bit fractional divider.

The baudrate generator is clocked derived via a prescaler (f_{DIV}) from the input clock f_{PCLK}. The baudrate timer counts downwards and can be started or stopped through the baudrate control run bit BCON.R. Each underflow of the timer provides one clock pulse to the serial channel. The timer is reloaded with the 11-bit BR_VALUE stored in its reload register BGH/BGL each time it underflow. The duration between underflows depends on the ‘n’ value in the fractional divider, which can be selected by the bits BGL.FD_SEL. ‘n’ times out of 32, the timer counts one cycle more than specified by BR_VALUE. The prescaler is selected by the bits BCON.BRPRE.

Register BGH/BGL is the dual-function Baudrate Generator/Reload register. Reading BGH/BGL returns the contents of the timer, while writing to BGH/BGL always updates the reload register.

Registers BGH/BGL may be written only when BCON.R is 0. An auto-reload of the timer with the contents of the reload register is performed one instruction cycle after the next time BCON.R is set. Any write to BGH/BGL, while BCON.R is set, is not allowed.

The baudrate of the baudrate generator depends on the following bits and register values:

- Input clock f_{PCLK}.

- Value of bit field BCON.BRPRE.
- Value of bit field BG.FD_SEL.
- Value of the 11-bit reload value BR_VALUE in registers BGH/BGL.

Figure 889 shows a simplified block diagram of the baudrate generator. Note that f_{BR} is 16 times the configured baudrate, due to synchronization and oversampling needs.

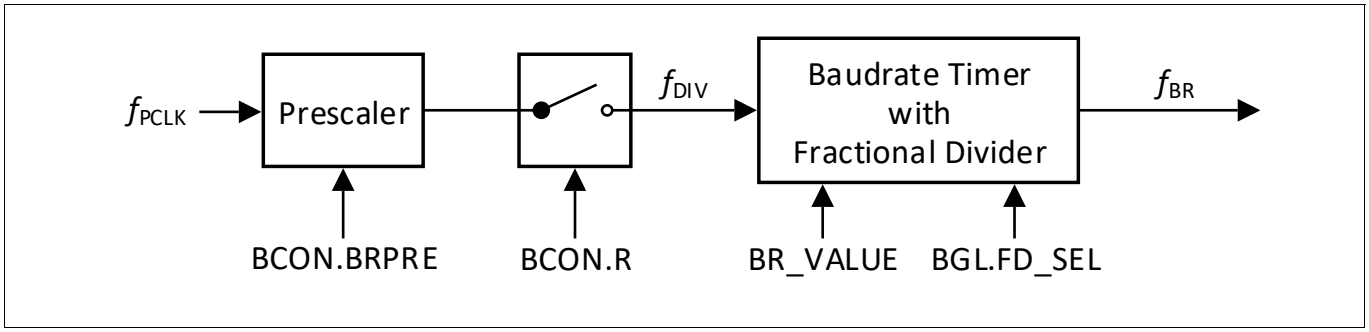


Figure 889 Simplified Baudrate Generator Block Diagram

The formula shown in **Figure 890** calculates the final baudrate.

$$\text{Baudrate} = \frac{f_{BR}}{16} = \frac{f_{PCLK}}{16 \times \text{Prescaler} \times (\text{BR\_VALUE} + n/32)} = \frac{f_{PCLK}}{16 \times (2^{\text{BCON.BRPRE}}) \times (\text{BR\_VALUE} + (\text{BGL.FD\_SEL}/32))}$$

uart_baudrate_formula.vsd

Figure 890 Baudrate Calculation

The value of the prescaler is chosen by the bit field BCON.BRPRE. BR_VALUE represents the contents of the reload value, taken as unsigned 11-bit integer from the two bit fields BR_VALUE in registers BGH and BGL. n/32 is defined by the fractional divider selection in bit field BGL.FD_SEL.

Note that the maximum baudrate that can be generated in this implementation is limited to $f_{PCLK}/48$. Hence, for a module clock of 20 MHz, the maximum achievable baudrate is 416.67 kBaud.

Table 710 lists various commonly used baudrates together with their corresponding parameter settings and the deviation errors compared to the intended baudrate.

Table 710 Typical Baudrates of UART (f_{PCLK} = 20 MHz)

Baudrate (f_{PCLK} = 20 MHz)	BCON.BRPRE	Reload Value (BR_VALUE)	Numerator of Fractional Value (BGL.FD_SEL)	BG Value[1]	Deviation Error
115.2 kBaud	000_B (prescaler = 1)	10 ($0A_H$)	27 ($1B_H$)	$015B_H$	+0.06%
20 kBaud	000_B (prescaler = 1)	62 ($3E_H$)	16 (10_H)	$07D0_H$	+0.00%
19.2 kBaud	000_B (prescaler = 1)	65 (41_H)	3 (03_H)	0823_H	+0.02%
9600 Baud	001_B (prescaler = 2)	65 (41_H)	3 (03_H)	0823_H	+0.02%
4800 Baud	010_B (prescaler = 4)	65 (41_H)	3 (03_H)	0823_H	+0.02%
2400 Baud	011_B (prescaler = 8)	65 (41_H)	3 (03_H)	0823_H	+0.02%

1) The BG value is obtained by concatenating the 11-bit BR_VALUE and 5-bit FD_SEL value to a 16-bit value, which represents the contents of the register pair BGH|BGL.

49.14.5 UART Interrupt Requests

The following figure illustrates the UART receive and transmit interrupt requests. Note that the RI/TI flags are not automatically cleared by the CPU on entry into the respective service routine. They must be cleared by software. See also the section on “Interrupt Structure 1” in chapter “Interrupt System”.

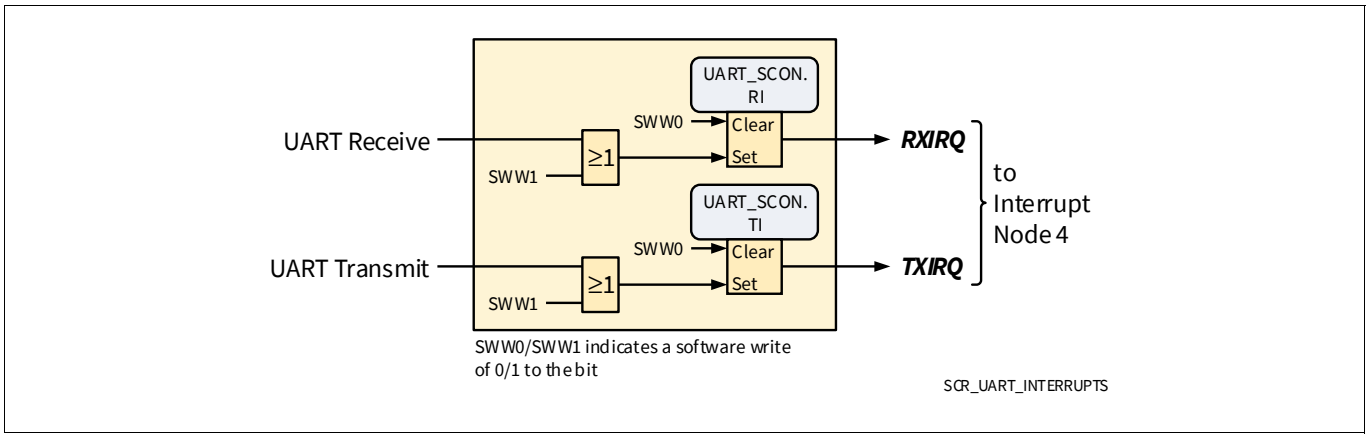


Figure 891 UART Receive and Transmit Interrupt Requests

49.14.6 LIN Support in UART

The UART module can be used to support the Local Interconnect Network (LIN) protocol for both master and slave operations. The LIN baudrate detection feature, which consists of the hardware logic for Break and Synch Field detection, provides the capability to detect the baudrate within LIN protocol using Timer 2. This allows the UART module to be synchronized to the LIN baudrate for data transmission and reception. Timer 2 will be used for BREAK detection and generation.

49.14.6.1 LIN Protocol

LIN is a holistic communication concept for local interconnected networks in vehicles. The communication is based on the SCI (UART) data format, a single-master/multiple-slave concept, a clock synchronization for nodes without stabilized time base. An attractive feature of LIN is self-synchronization of the slave nodes without a crystal or ceramic resonator, which significantly reduces the cost of hardware platform. Hence, the baudrate must be calculated and returned with every message frame.

The structure of a LIN frame is shown in **Figure 892**. The frame consists of the:

- header, which comprises a Break (13-bit time low), Synch Byte (55_H), and ID field
- response time
- data bytes (according to UART protocol)
- checksum

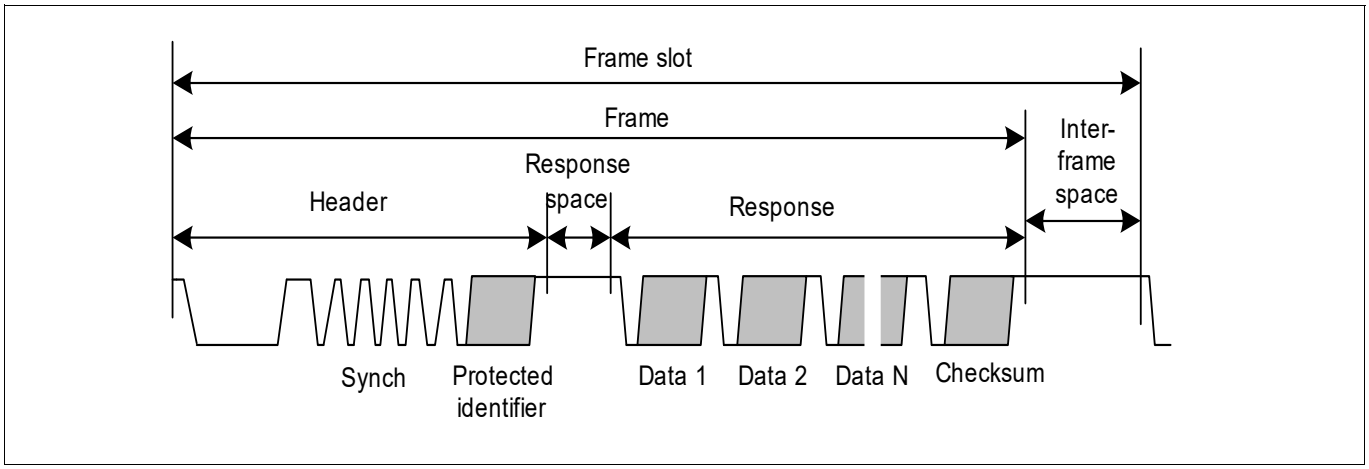


Figure 892 The Structure of LIN Frame

Each byte field is transmitted as a serial byte, as shown in **Figure 893**. The LSB of the data is sent first and the MSB is sent last. The start bit is encoded as a bit with value zero (dominant) and the stop bit is encoded as a bit with value one (recessive).

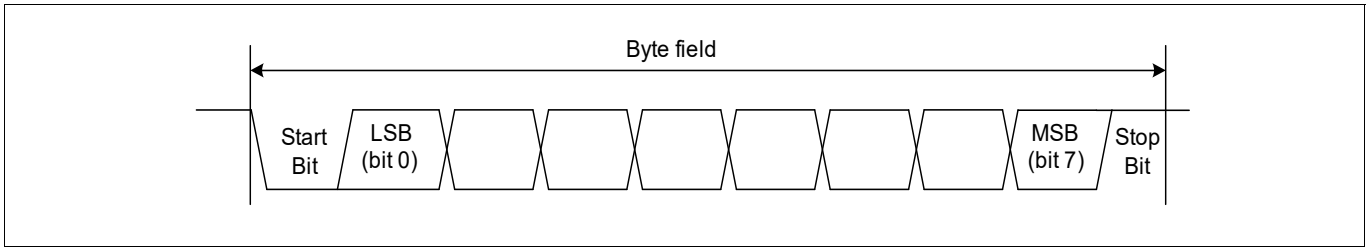


Figure 893 The Structure of Byte Field

The break is used to signal the beginning of a new frame. It is the only field that does not comply with **Figure 893**. A break is always generated by the master task (in the master mode) and it must be at least 13 bits of dominant value, including the start bit, followed by a break delimiter, as shown in **Figure 894**. The maximum length is 26.6 bit times. The break delimiter will be at least one nominal bit time long.

A slave node will use a break detection threshold of 11 nominal bit times.

Figure 894 The Break Field

Synch Byte is a specific pattern for determination of time base. The byte field is with the data value 55_H, as shown in **Figure 895**.

A slave task is always able to detect the Break/Synch sequence, even if it expects a byte field (assuming the byte fields are separated from each other). If this happens, detection of the Break/Synch sequence will abort the transfer in progress and processing of the new frame will commence.

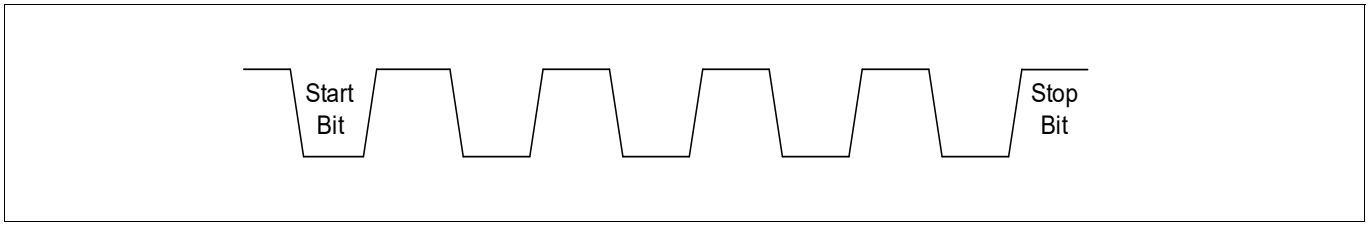


Figure 895 The Synch Byte Field

The slave task will receive and transmit data when an appropriate ID is sent by the master:

1. Slave waits for Synch Break
2. Slave synchronizes on Synch Byte
3. Slave snoops for ID
4. According to ID, slave determines whether to receive or transmit data, or do nothing
5. When transmitting, the slave sends 2, 4 or 8 data bytes, followed by check byte

49.14.6.2 LIN Header Transmission

LIN header transmission is only applicable in master mode. In the LIN communication, a master task decides when and which frame is to be transferred on the bus. It also identifies a slave task to provide the data transported by each frame. The information needed for the handshaking between the master and slave tasks is provided by the master task through the header portion of the frame.

The header consists of a break and synch pattern followed by an identifier. Among these three fields, only the break pattern cannot be transmitted as a normal 8-bit UART data. The break must contain a dominant value of 13 bits or more to ensure proper synchronization of slave nodes.

In the LIN communication, a slave task is required to be synchronized at the beginning of the protected identifier field of frame. For this purpose, every frame starts with a sequence consisting of a break field followed by a synch byte field. This sequence is unique and provides enough information for any slave task to detect the beginning of a new frame and be synchronized at the start of the identifier field.

49.14.6.3 Automatic Synchronization to the Host

Upon entering LIN communication, a connection is established and the transfer speed (baudrate) of the serial communication partner (host) is automatically synchronized in the following steps that are to be included in user software:

STEP 1: Initialize interface for reception and timer for baudrate measurement

STEP 2: Wait for an incoming LIN frame from host

STEP 3: Synchronize the baudrate to the host

STEP 4: Enter for Master Request Frame or for Slave Response Frame

The next section, **Section 49.14.6.4**, provides some hints on setting up the microcontroller for baudrate detection of LIN.

*Note: Re-synchronization and setup of baudrate are always done for **every** Master Request Header or Slave Response Header LIN frame.*

49.14.6.4 Initialization of Break/Synch Field Detection Logic

The LIN baudrate detection feature provides measures to detect the baudrate within the LIN protocol using Timer 2 within the LIN software. Initialization consists of:

- Serial port of the microcontroller set to Mode 1 (8-bit UART, variable baudrate) for communication.
- Provide the baudrate range via bit field BCON.BGSEL.
- Toggle BCON.BRDIS bit (set the bit to 1 before clearing it back to 0) to initialize the Break/Synch detection logic.
- Clear all status flags LINST.BRK, LINST.EOFSYN and LINST.ERRSYN to 0.
- Timer 2 is set to capture mode with falling edge trigger at input T2EX. Bit T2_MOD.EDGESEL is set to 0 by default, and bit T2_CON.CP_RL2 is set to 1.
- Timer 2 external events are enabled. T2_CON. EXEN2 is set to 1 (EXF2 flag is set when a negative transition occurs at input T2EX).
- f_{T2} can be configured by bit field T2_MOD.T2PRE.

Note: It is also recommended to toggle the BCON.BRDIS bit after the reception of each LIN frame (complete or erroneous frame) to avoid a wrong Break field detection in noisy environments (i.e. spikes on the LIN bus).

49.14.6.5 Break Detection

The Break/Synch Field detection logic supports a maximum number of bits in the Break field as defined by the conformance test 26.6 bits.

49.14.6.6 Baudrate Range Selection

If the maximum number of bits in the Break field is exceeded, the internal counter will overflow, which results in a baudrate detection error. Therefore, an appropriate BGSEL value has to be selected for the required baudrate detection range.

The baudrate range defined by different BGSEL settings is shown in **Table 711**.

Each BGSEL setting supports a range of baudrate for detection. If the baudrate used is outside the defined range, the baudrate may not be detected correctly.

Table 711 BGSEL Bit Field Definition for Different Input Frequencies

f_{PCLK}	BGSEL	Baudrate Selection for Detection $f_{PCLK}/(2184*2^{BGSEL})$ to $f_{PCLK}/(74*2^{BGSEL})$
20 MHz	00_B	9.2 kHz to 270.2 kHz
	01_B	4.6 kHz to 135.1 kHz
	10_B	2.3 kHz to 67.5 kHz
	11_B	1.1 kHz to 33.7 kHz

When f_{PCLK} = 20 MHz, the baudrate range between 1.1 kHz to 270.2 kHz can be detected. The following examples serve as a guide to select the BGSEL value:

- If the baudrate falls in the range of 1.2 kHz to 2.3 kHz, the selected BGSEL value is 11_B.
- If the baudrate falls in the range of 2.3 kHz to 4.6 kHz, the selected BGSEL value is 10_B.
- If the baudrate falls in the range of 4.6 kHz to 9.2 kHz, the selected BGSEL value is 01_B.
- If the baudrate falls in the range of 9.2 kHz to 270.2 kHz, the selected BGSEL value is 00_B. If the baudrate is 20 kHz, the possible values of BGSEL that can be selected are 00_B, 01_B, 10_B, and 11_B. However, it is advisable to select 00_B for better detection accuracy.

49.14.6.7 LIN Baudrate Detection

The configuration has to take place in software. The baudrate detection for LIN is shown in **Figure 896**, the Header LIN frame consists of the:

- SYN Break (13 bit times low)
- SYN byte (55_H)
- Protected ID field

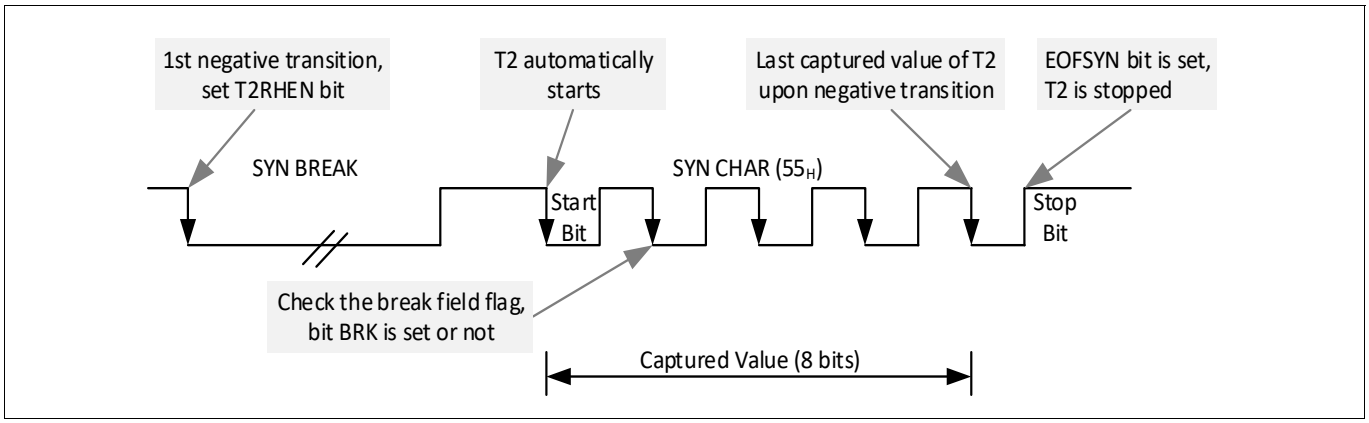


Figure 896 LIN Auto Baudrate Detection

With the first falling edge:

- The Timer 2 External Start Enable bit (T2_MOD.T2RHEN) is set. The falling edge at input T2EX is selected by default for Timer 2 External Start (bit T2_MOD.T2REGS is 0).

With the second falling edge:

- Start Timer 2 by the hardware.

With the third falling edge:

- Timer 2 captures the timing of 2 bits of SYN byte.
- Check the Break Field Flag bit LINST.BRK.

If the Break Field Flag LINST.BRK is set, software may continue to capture 4/6/8 bits of SYN byte. Finally, the End of SYN Byte Flag (LINST.EOFSYN) is set, Timer 2 is stopped. T2 Reload/Capture register (T2_RC2H/L) is the time taken for 2/4/6/8 bits according to the implementation. Then the LIN routine calculates the actual baudrate, sets the PRE and BG values if the UART module uses the baudrate generator for baudrate generation.

After the third falling edge, the software may discard the current operation and continue to detect the next header LIN frame if the following conditions were detected:

- The Break Field Flag LINST.BRK is not set, or
- The SYN Byte Error Flag LINST.ERRSYN is set, or
- The Break Field Flag LINST.BRK is set, but the End of SYN Byte Flag LINST.EOFSYN and the SYN Byte Error Flag LINST.ERRSYN are not set.

49.14.6.8 LIN Interrupt Requests

The following figure illustrates the LIN interrupt requests. Note that the flags must be cleared by software. See also the section on “Interrupt Structure 2” in chapter “Interrupt System”.

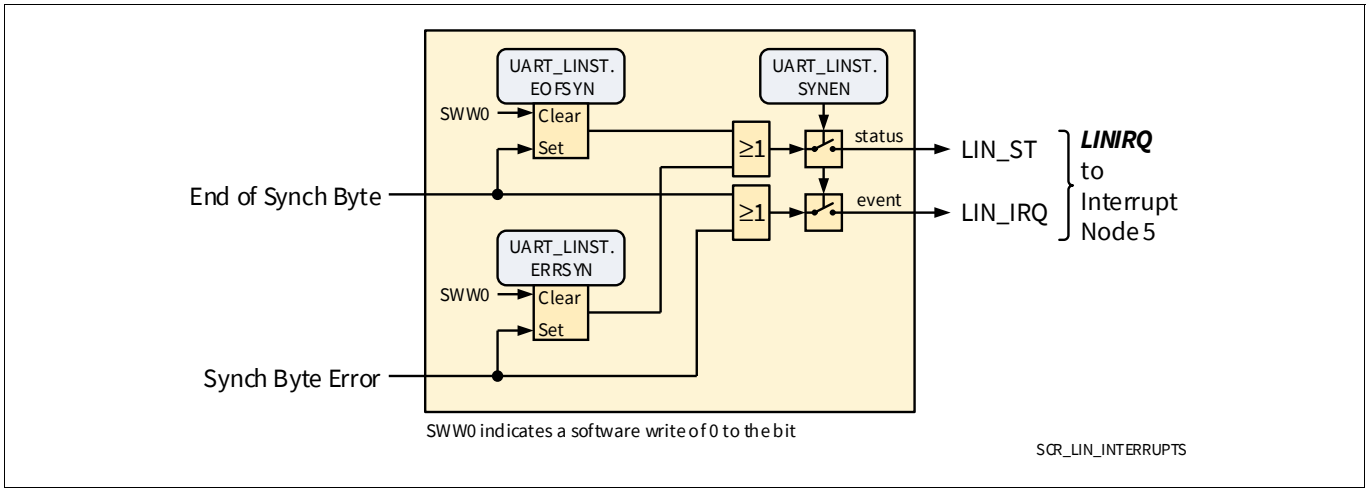


Figure 897 LIN Interrupt Requests

49.14.7 UART Connections to GPIO

The following figure illustrates the signal connections of the UART to the GPIO ports.

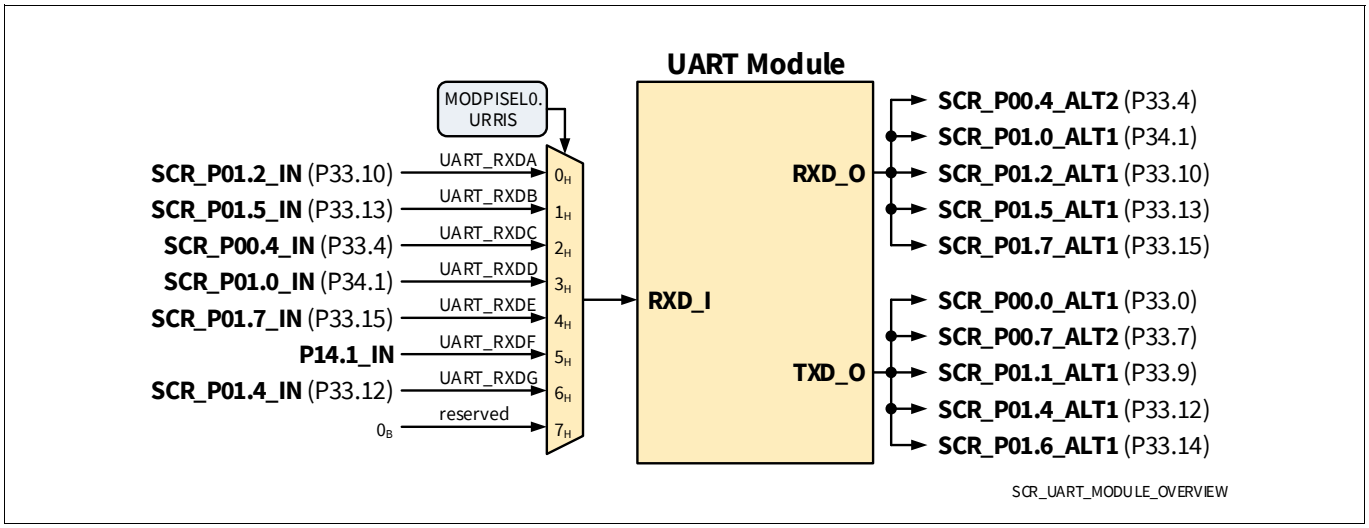


Figure 898 UART Connections to GPIO Ports

49.14.8 Register Description

The UART registers can be accessed from both, the standard (non-mapped) and mapped SFR area.

Table 712 and Table 713 list the addresses of these SFRs.

Table 712 Register Overview - UART (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
UART_SBUF	Serial Data Buffer	0	X	$0BB_H$	276
UART_SCON	Serial Channel Control Register	0	X	$0BA_H$	276

Table 713 Register Overview - LIN (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
UART_BCON	Baudrate Control Register	0	X	$0BD_H$	279
UART_BGH	Baudrate Timer/Reload Register, High Byte	0	X	$0BF_H$	283
UART_BGL	Baudrate Timer/Reload Register, Low Byte	0	X	$0BC_H$	282
UART_LINST	LIN Status Register	0	X	$0BE_H$	281

49.14.8.1 UART Registers

UART contains the two Special Function Registers (SFRs), SCON and SBUF. SCON is the control register and SBUF is the data register. On reset, both SCON and SBUF return 00_H. The serial port control and status register is the SFR SCON. This register contains not only the mode selection bits, but also the 9th data bit for transmit and receive (TB8 and RB8) and the serial port interrupt bits (TI and RI).

SBUF is the receive and transmit buffer of the serial interface. Writing to SBUF loads the transmit register and initiates transmission. This register is used for both transmit and receive data. Transmit data is written to this location and receive data is read from this location, but the two paths are independent.

Reading out SBUF accesses a physically separate receive register.

Serial Data Buffer

Note: Register UART_SBUF may only be written to when no transmission is in progress (a previously commenced transmission is finished when bit UART_SCON.TI is set to 1). Otherwise, an ongoing transmission may be corrupted.

UART_SBUF
Serial Data Buffer **($0BB_H$)** **Reset Value: Table 714**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
VAL							
rwh							

Field	Bits	Type	Description
VAL	7:0	rwh	**Serial Interface Buffer Register**

Table 714 Reset Values of UART_SBUF

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

Serial Channel Control Register

UART_SCON
Serial Channel Control Register **($0BA_H$)** **Reset Value: Table 715**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
SM01	SM01	SM2	REN	TB8	RB8	TI	RI
rw	rw	rw	rw	rw	rwh	rwh	rwh

Field	Bits	Type	Description
RI	0	rwh	**Receive Interrupt Flag** This is set by hardware at the end of the 8th bit on mode 0, or at the half point of the stop bit in modes 1, 2, and 3. This bit can also be set by software. Must be cleared by software. *Note: In case of an ongoing reception and a mode change at the very same time, the flag RI will have an undefined behaviour, as it is dependent from the point in time of the mode change.*
TI	1	rwh	**Transmit Interrupt Flag** This is set by hardware at the end of the 8th bit in mode 0, or at the beginning of the stop bit in modes 1, 2, and 3. This bit can also be set by software. Must be cleared by software. *Note: If a transmission has been started and TI is still 0, a further write to UART_SBUF will destroy the ongoing transmission. If TI is set to 1, the next transmission can be started.*
RB8	2	rwh	**Serial Port Receiver Bit 9** In modes 2 and 3, this is the 9th data bit received. In mode 1, this is the stop bit received. In mode 0, this bit is not used.
TB8	3	rw	**Serial Port Transmitter Bit 9** In modes 2 and 3, this is the 9th data bit sent.
REN	4	rw	**Enable Receiver of Serial Port** *Note: The operating mode of the UART must not be changed while bit REN is set (or while a transmission is in progress). Before changing the mode of the UART, bit REN shall be cleared to 0.* 0_B Serial reception is disabled 1_B Serial reception is enabled
SM2	5	rw	**Enable Serial Port Multiprocessor Communication in Modes 2 and 3** In mode 2 or 3, if SM2 is set to 1, RI will not be activated if the received 9th data bit (RB8) is 0. In mode 1, if SM2 is set to 1, RI will not be activated if a valid stop bit (RB8) was not received. In mode 0, SM2 should be 0. *Note: The operating mode of the UART must not be changed while bit REN is set or while a transmission is in progress. Before changing the mode of the UART, make sure that no transmissions are in progress, and that bit REN is cleared to 0.*

Field	Bits	Type	Description
SM01	7:6	rw	**Serial Port Operating Mode Selection** *Note: The operating mode of the UART must not be changed while bit REN is set or while a transmission is in progress. Before changing the mode of the UART, make sure that no transmissions are in progress, and that bit REN is cleared to 0.* 00_B **MODE_0**, 8-bit shift register, fixed baud rate (fPCLK/2) 01_B **MODE_1**, 8-bit UART, variable baud rate 10_B **MODE_2**, 9-bit UART, fixed baud rate (fPCLK/64 or fPCLK/32) 11_B **MODE_3**, 9-bit UART, variable baud rate

Table 715 Reset Values of UART_SCON

Reset Type	Reset Value	Note
Generated Reset	00_H	
LVD Reset	00_H	

49.14.8.2 Baudrate Generator Control and Status Registers

Power Control Register

PCON
Power Control Register **($0D9_H$)** **Reset Value: Table 716**
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
SMOD		**0**		**GF1**	**GF0**	**0**	**IDLE**
rw		r		rw	rw	r	rw

Field	Bits	Type	Description
IDLE	0	rw	**Idle Mode Enable** 0_B Do not enter Idle Mode 1_B Enter Idle Mode
GF0	2	rw	**General Purpose Flag Bit 0**
GF1	3	rw	**General Purpose Flag Bit 1**
SMOD	7	rw	**Double Baud Rate Enable** 0_B Do not double the baud rate of serial interface in mode 2 1_B Double the baud rate of serial interface in mode 2
0	1, 6:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 716 Reset Values of PCON

Reset Type	Reset Value	Note
Generated Reset	$0XXX\ 00X0_B$	
LVD Reset	$0–––\ 00–0_B$	

Baudrate Control Register

UART_BCON
Baudrate Control Register **($0BD_H$)** **Reset Value: Table 717**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
BGSEL		**0**	**BRDIS**		**BRPRE**		**R**
rw		r	rw		rw		rw

Field	Bits	Type	Description
R	0	rw	**Baudrate Generator Run Control Bit** *Note: BR_VALUE shall be written if, and only if, R = 0.* 0_B Baudrate generator disabled 1_B Baudrate generator enabled
BRPRE	3:1	rw	**Prescaler Bit** Selects the input clock for fDIV, which is derived from the peripheral clock. 000_B **DIV**, fDIV = fPCLK 001_B **DIV2**, fDIV = fPCLK / 2 010_B **DIV4**, fDIV = fPCLK / 4 011_B **DIV8**, fDIV = fPCLK / 8 100_B **DIV16**, fDIV = fPCLK / 16 101_B **DIV32**, fDIV = fPCLK / 32 **others**, Reserved
BRDIS	4	rw	**Baudrate Detection Disable** *Note: It is recommended to toggle bit BRDIS after the reception of each LIN frame (complete or erroneous frame) to avoid a wrong Break field detection in noisy environments (i.e. spikes on the LIN bus).* 0_B Break/Sync detection is enabled 1_B Break/Sync detection is disabled
BGSEL	7:6	rw	**Baudrate Select for Detection** For different values of BGSEL, the baud rate range for detection is defined by the following formula: fpclk/(2184*2^BGSEL) < baud rate range < fpclk/(74*2^BGSEL); where BGSEL = 00_B, 01_B, 10_B, 11_B. See table "BGSEL Bit Field Definition for Different Input Frequencies" for bit field BGSEL definition for different input frequencies.
0	5	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 717 Reset Values of UART_BCON

Reset Type	Reset Value	Note
LVD Reset	$00X0\ 0000_B$	
Generated Reset	$00–0\ 0000_B$	

LIN Status Register

UART_LINST
LIN Status Register **($0BE_H$)** **Reset Value: Table 718**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0	**SYNEN**	**ERRSYN**	**EOFSYN**	**BRK**		**0**	
r	rw	rwh	rwh	rwh		r	

Field	Bits	Type	Description
BRK	3	rwh	**Break Field Flag** This bit is set by hardware and can only be cleared by software. 0_B Break Field is not detected 1_B Break Field is detected
EOFSYN	4	rwh	**End of SYN Byte Interrupt Flag** This bit is set by hardware and can only be cleared by software. 0_B End of SYN Byte is not detected 1_B End of SYN Byte is detected
ERRSYN	5	rwh	**SYN Byte Error Interrupt Flag** This bit is set by hardware and can only be cleared by software. 0_B Error is not detected in SYN Byte 1_B Error is detected in SYN Byte
SYNEN	6	rw	**End of SYN Byte and SYN Byte Error Interrupts Enable** 0_B End of SYN Byte and SYN Byte Error Interrupts are not enabled 1_B End of SYN Byte and SYN Byte Error Interrupts are enabled
0	2:0, 7	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 718 Reset Values of UART_LINST

Reset Type	Reset Value	Note
LVD Reset	$X000\ 0XXX_B$	
Generated Reset	$-000\ 0---_B$	

49.14.8.3 Baudrate Generator Timer/Reload Registers

The low and high bytes of the baudrate timer/reload register BG contain the 11-bit reload value for the baudrate timer and the 5-bit fractional divider selection.

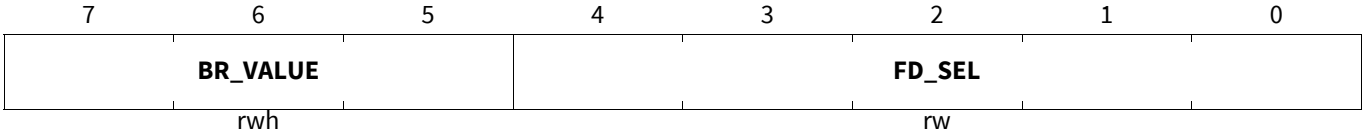
Reading the low byte of register BG returns the content of the lower three bits of the baudrate timer and the FD_SEL setting, while reading the high byte returns the content of the upper 8 bits of the baudrate timer.

Writing to register BG loads the baudrate timer with the reload and fractional divider values from the BG register, the first instruction cycle after UART_BCON.R is set.

BG may only be written if R = 0.

Baudrate Timer/Reload Register, Low Byte

UART_BGL
Baudrate Timer/Reload Register, Low Byte (0BC$_H$) **Reset Value: Table 719**
RMAP: 0, PAGE: X



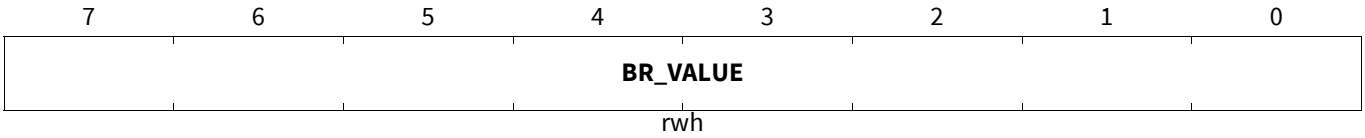
Field	Bits	Type	Description
FD_SEL	4:0	rw	**Fractional Divider Selection** Selects the fractional divider to be n/32, where n is the value of FD_SEL, and is in the range of 0 to 31. For example, writing 0001_B to FD_SEL selects the fractional divider to be 1/32. *Note: Fractional divider has no effect if the 11-bit BR_VALUE = 000_H.*
BR_VALUE	7:5	rwh	**Baudrate Timer/Reload Value** The lower three bits of the 11-bit Baudrate Timer/Reload value. See also description in BGH register. *Note: BR_VALUE shall be written if, and only if, BCON.R = 0.*

Table 719 Reset Values of UART_BGL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Baudrate Timer/Reload Register, High Byte

UART_BGH
Baudrate Timer/Reload Register, High Byte (0BF$_H$) **Reset Value: Table 720**
RMAP: 0, PAGE: X



Field	Bits	Type	Description
BR_VALUE	7:0	rwh	**Baudrate Timer/Reload Value** The upper 8 bits of the 11-bit Baudrate Timer/Reload value. When the 11-bit BR_VALUE is 000$_H$, the baudrate timer is bypassed. *Note: BR_VALUE shall be written if, and only if, BCON.R = 0.*

Table 720 Reset Values of UART_BGH

Reset Type	Reset Value	Note
LVD Reset	00$_H$	
Generated Reset	00$_H$	

49.14.9 Revision History

Table 721 Revision History

Reference	Change to Previous Version	
v2.8		
	First Official Release of completely reworked SCR chapter	
	No change	

49.15 Synchronous Serial Channel

This chapter describes the Synchronous Serial Channel (SSC).

49.15.1 Overview

The Synchronous Serial Channel (SSC) supports both full-duplex and half-duplex serial synchronous communication. The serial clock signal can be generated by the SSC itself (Master Mode) through its own 16-bit baud-rate generator, or can be received from an external master (Slave Mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices using other synchronous serial channel. Transmission and reception of data is double-buffered. In slave mode, the SSC is limited to 1 MBaud/s.

Features

- Master and Slave Mode operation
  - Full-duplex or half-duplex operation
- Transmit and receive buffered; four-stage FIFO for receive
- Flexible data format
  - Programmable number of data bits: 4 to 8 bits (1, 2 and 3 bit transfers are programmable, but forbidden)
  - Programmable shift direction: Least Significant Bit (LSB) or Most Significant Bit (MSB) shift first
  - Programmable clock polarity: idle low or high state for the shift clock
  - Programmable clock/data phase: data shift with leading or trailing edge of the shift clock
- Variable baud-rate
- Compatible with Serial Peripheral Interface (SPI)
- Interrupt generation
  - On a transmitter empty condition
  - On a receive condition
  - On a receiver FIFO full and empty condition.
  - On an error condition (receive, baud-rate, transmit error)

Figure 899 shows the block diagram of the SSC.

Note: In this chapter, the input and output signals from/to the GPIO ports are referred to with their generic name, such as MTSR, MRST, or SCLK. However, the actual input and output signals use separate lines from/to the GPIO, since they are connected in different ways to the pin logic (pin input line; alternate data output line). Only where explicitly applicable, the generic signal names are extended with an "_I" (input line) or "_O" (output line) identifier.

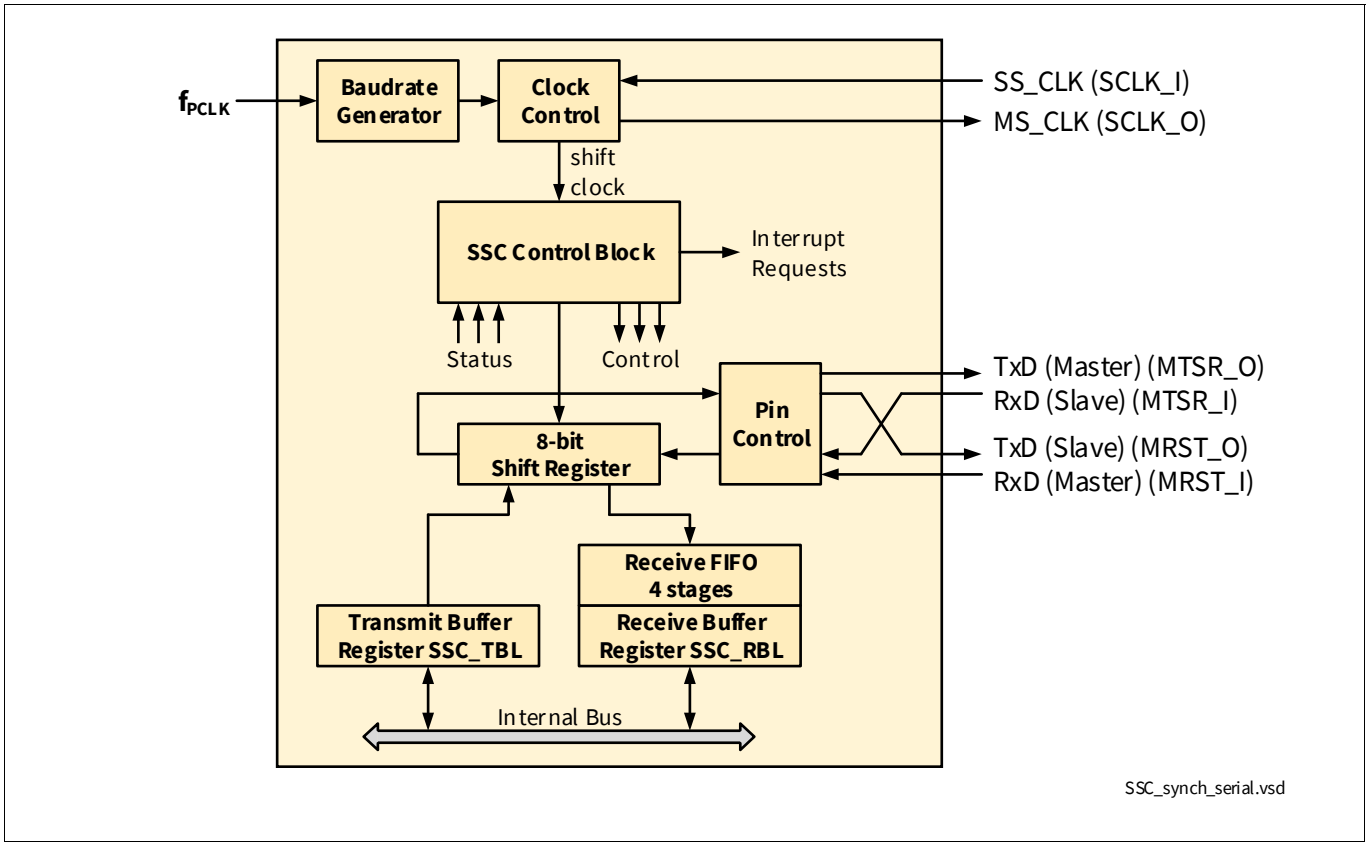


Figure 899 SSC Block Diagram

49.15.2 General Operation

The SSC supports full-duplex and half-duplex synchronous communication up to 3.33 MBaud (@ 20 MHz module clock). The serial clock signal can be generated by the SSC itself (Master Mode) or can be received from an external master (Slave Mode). Data width, shift direction, clock polarity, and phase are programmable. This allows communication with SPI-compatible devices. Transmission and reception of data is double-buffered. A 16-bit baud-rate generator provides the SSC with a separate serial clock signal.

The SSC can be configured in a very flexible way, so it can be used with other synchronous serial channels, can serve for master/slave or multimaster interconnections, or can operate compatible with the popular SPI interface. Thus, the SSC can be used to communicate with shift registers (I/O expansion), peripherals (e.g. EEPROMs, etc.) or other controllers (networking). The SSC supports half-duplex and full-duplex communication. Data is transmitted or received on lines TXD and RXD, normally connected with pins MTSR (Master Transmit/Slave Receive) and MRST (Master Receive/Slave Transmit). The clock signal is output via line MS_CLK (Master Serial Shift Clock) or input via line SS_CLK (Slave Serial Shift Clock). Both lines are normally connected to pin SCLK.

49.15.2.1 Operating Mode Selection

The operating mode of the serial channel SSC is controlled by two register pairs, sharing the same addresses:

- During programming (SSC disabled; SSC_CONPH.EN = 0), access to register pair SSC_CONPH/SSC_CONPL is available, providing access to a set of control bits.
- During operation (SSC enabled; SSC_CONOH.EN = 1), access to register pair SSC_CONOH/SSC_CONOL is available, providing access to a set of control/status flags.

The two register pairs share the same two addresses. The selection between the registers is depending on the state of the enable bit EN prior to an access to the register pairs.

The shift register of the SSC is connected to both the transmit lines and the receive lines via the pin control logic. Transmission and reception of serial data are synchronized and take place at the same time, i.e. the same number of transmitted bits is also received. Transmit data is written into the Transmit Buffer (SSC_TBL) and it is moved, on the next shifting edge, to the shift register as soon as this is empty. An SSC master (SSC_CONOH.MS = 1) immediately begins transmitting, while an SSC slave (SSC_CONOH.MS = 0) will wait for an active shift clock. When the transfer starts, the busy flag SSC_CONOH.BSY is set and the Transmit Interrupt Request line TIR will be activated within the timeframe of the first bit to indicate that register SSC_TBL may be reloaded again. When the programmed number of bits (4…8) has been transferred, the contents of the shift register are moved to the Receive FIFO and the Receive Interrupt Request line RIR will be activated for the first received frame. If no further transfer is to take place (SSC_TBL is empty), SSC_CONOH.BSY will be cleared (Master Mode). In Slave Mode, SSC_CONOH.BSY will always be cleared after a single transfer. Software shall not modify SSC_CONOH.BSY, as this flag is hardware controlled.

Note: The SSC starts transmission and sets SSC_CONOH.BSY minimum two clock cycles after transmit data is written into SSC_TBL. Therefore, it is not recommended to poll SSC_CONOH.BSY to check for the start and end of a single transmission. Instead, interrupt service routines should be used if interrupts are enabled, or the interrupt flags IRCON1.TIR and IRCON1.RIR should be polled if interrupts are disabled.

Note: Only one serial interface can be master at a given time.

The transfer of serial data bits can be programmed in many respects:

- The data width can be specified from 4 bits to 8 bits
- A transfer may start with either the LSB or the MSB
- The shift clock may be idle low or idle high
- The data bits may be shifted with the leading edge or the trailing edge of the shift clock signal
- The baud-rate may be set from 152.59 Baud up to 3.33 MBaud (@ 20 MHz module clock)
- The shift clock can be generated (MS_CLK) or can be received (SS_CLK)

These features allow the adaptation of the SSC to a wide range of applications requiring serial data transfer.

The Data Width Selection supports the transfer of frames of any data length, from 4-bit “characters” up to 8-bit “characters”. Starting with the LSB (SSC_CONPL.HB = 0) allows communication with SSC devices in Synchronous Mode or with 8051-like serial interfaces, for example. Starting with the MSB (SSC_CONPL.HB = 1) allows operation compatible with the SPI interface.

Regardless of the data width selected and whether the MSB or the LSB is transmitted first, the transfer data is always right-aligned in registers SSC_TBL and SSC_RBL, with the LSB of the transfer data in bit 0 of these registers. The data bits are rearranged for transfer by the internal shift register logic. The unselected bits of SSC_TBL are ignored; the unselected bits of SSC_RBL will not be valid and should be ignored by the receiver service routine.

The Clock Control allows the adaptation of transmit and receive behavior of the SSC to a variety of serial interfaces. A specific shift clock edge (rising or falling) is used to shift out transmit data, while the other shift clock edge is used to latch in receive data. Bit SSC_CONPL.PH selects the leading edge or the trailing edge for each function. Bit SSC_CONPL.PO selects the level of the shift clock line in the idle state. Thus, for an idle-high clock, the leading edge is a falling one, a 1-to-0 transition (see **Figure 900**).

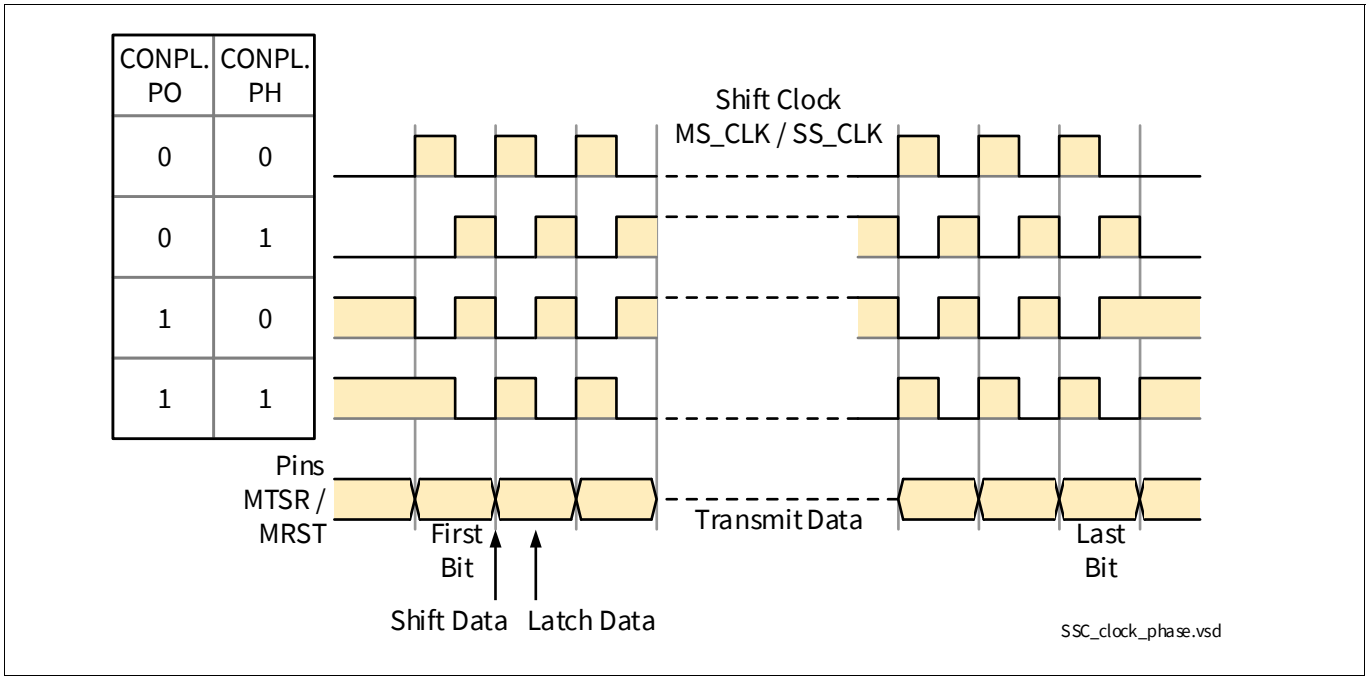


Figure 900 Serial Clock Phase and Polarity Options

49.15.2.2 Full-Duplex Operation

The various devices are connected through three lines. The definition of these lines is always determined by the master: the line connected to the master's data output line TXD is the transmit line; the receive line is connected to its data input line RXD; the shift clock line is either MS_CLK or SS_CLK. Only the device selected for master operation generates and outputs the shift clock on line MS_CLK. Since all slaves receive this clock, their pin SCLK must be switched to input mode. The output of the master's shift register is connected to the external transmit line, which in turn is connected to the slaves' shift register input. The output of the slaves' shift register is connected to the external receive line in order to enable the master to receive the data shifted out of the slave. The external connections are hard-wired, the function and direction of these pins is determined by the master or slave operation of the individual device.

Note: The shift direction shown in the figure applies for MSB-first operation as well as for LSB-first operation.

When initializing the devices in this configuration, one device must be selected for master operation while all other devices must be programmed for slave operation. Initialization includes the operating mode of the device's SSC and also the function of the respective port lines.

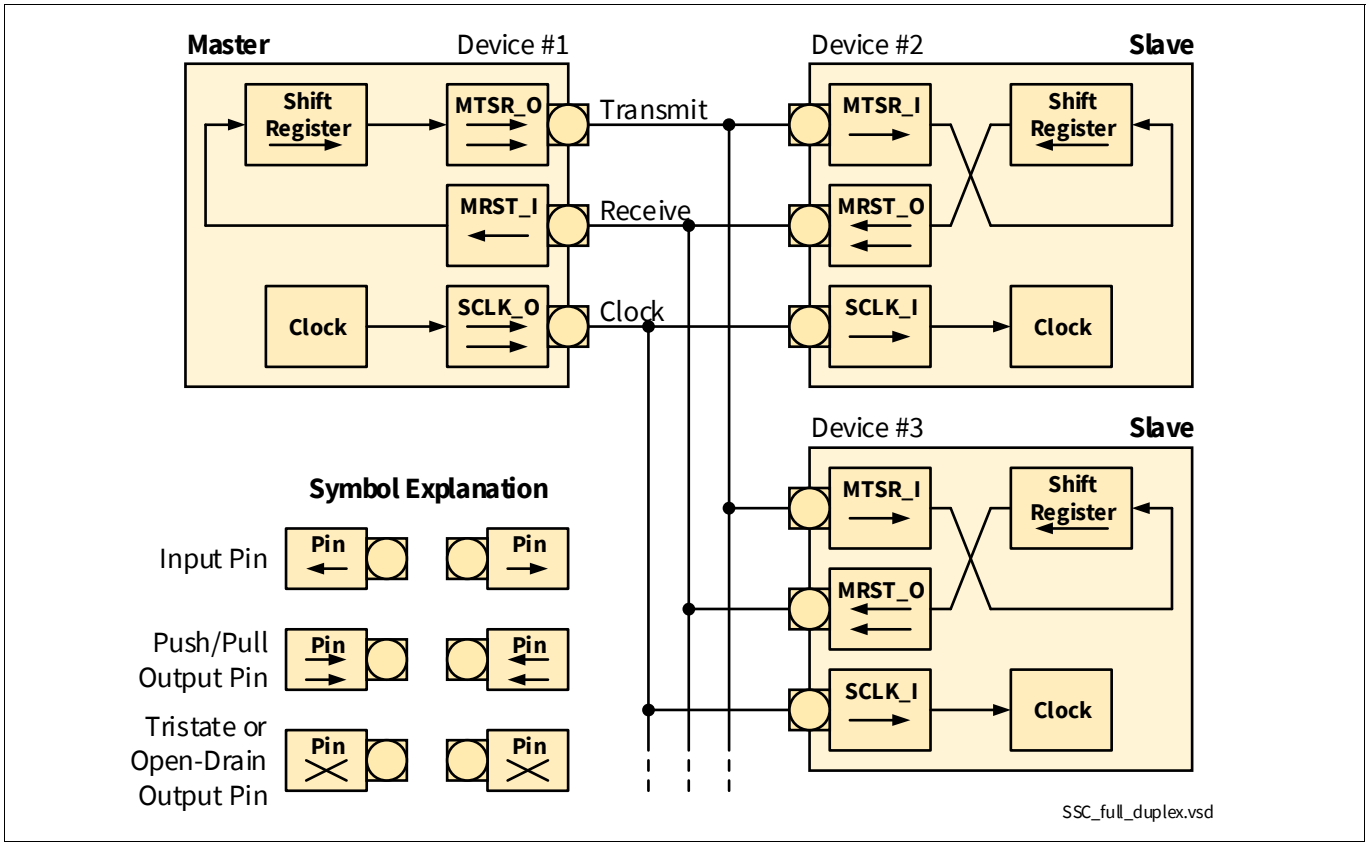


Figure 901 SSC Full-Duplex Configuration

The data output pins MRST of all slave devices are connected together onto the one receive line in the configuration shown in **Figure 901**. During a transfer, each slave shifts out data from its shift register. There are two ways to avoid collisions on the receive line due to different slave data:

- Only one slave drives the line, i.e. enables the driver of its MRST pin. All the other slaves must have their MRST pins programmed as input so only one slave can put its data onto the master's receive line. Only receiving data from the master is possible. The master selects the slave device from which it expects data either by separate select lines, or by sending a special command to this slave. The selected slave then switches its MRST line to output until it gets a de-selection signal or command. This option is applicable only if the pin has output driver disabling capability.
- The slaves use open-drain output on MRST. This forms a wired-AND connection. The receive line needs an external pull-up in this case. Corruption of the data on the receive line sent by the selected slave is avoided when all slaves not selected for transmission to the master only send ones (1s). Because this high level is not actively driven onto the line, but only held through the pull-up device, the selected slave can pull this line actively to a low-level when transmitting a zero bit. The master selects the slave device from which it expects data either by separate select lines or by sending a special command to this slave.

After performing the necessary initialization of the SSC, the serial interfaces can be enabled. For a master device, the alternate clock line will now go to its programmed polarity. The alternate data line will go to either 0 or 1 until the first transfer starts.

When the serial interfaces are enabled, the master device can initiate the first data transfer by writing the transmit data into register SSC_TBL. This value is copied into the shift register (assumed to be empty at this time), and the selected first bit of the transmit data will be placed onto the TXD line before the next clock from the baud-rate generator (transmission starts only if SSC_CONPH.EN = 1).

Depending on the selected clock phase, a clock pulse will also be generated on the MS_CLK line. At the same time, with the opposite clock edge, the master latches and shifts in the data detected at its input line RXD. This “exchanges” the transmit data with the receive data. Because the clock line is connected to all slaves, their shift registers will be shifted synchronously with the master’s shift register — shifting out the data contained in the registers, and shifting in the data detected at the input line. After the pre-programmed number of clock pulses (via the data width selection), the data transmitted by the master is contained in all the slaves’ shift registers, while the master’s shift register holds the data of the selected slave. In the master and all slaves, the contents of the shift register are copied into the receive FIFO and the receive interrupt line RIR is activated for the first received frame.

When a slave device is transmitting its data, the bit SSC_CONOH.BSY is not set until the first clock edge at SS_CLK appears. The slave device will not wait for the next clock from the baud-rate generator, as the master does. The reason for this is that, depending on the selected clock phase, the first clock edge generated by the master may already be used to clock in the first data bit. Thus, the slave’s first data bit must already be valid at this time.

Note: For Master and Slave Mode the following applies: In the case SSC_CONPL.PH = 1, the value is placed immediately on the bus, while for SSC_CONPL.PH = 0, it is placed on the output with the first shifting edge. Besides, the value is not immediately copied into the shift register, but with the first shifting edge.

Note: On the SSC, a transmission and a reception takes place at the same time, regardless of whether valid data has been transmitted or received.

Note: The initialization of the SCLK pin on the master requires some attention in order to avoid undesired clock transitions, which may disturb the other devices. Before the clock pin is switched to output mode, the clock output level should be selected in the control register SSC_CONPL, and the alternate output be prepared via the appropriate GPIO register, or the output latch must be loaded with the clock idle level.

49.15.2.3 Half-Duplex Operation

In a Half-Duplex Mode, only one data line is necessary for both receiving **and** transmitting of data (see **Figure 902**).

The master device controls the data transfer by generating the shift clock, while the slave devices receive it. Due to the fact that all transmit and receive pins are connected to the one data exchange line, serial data may be moved between arbitrary stations.

Similar to Full-Duplex Mode, there are two ways to avoid collisions on the data exchange line:

- Only the transmitting device may enable its transmit pin driver. This option is applicable only if the pin has output driver disabling capability.
- The non-transmitting devices use open-drain output and send only ones.

Because the data inputs and outputs are connected together, a transmitting device will clock in its own data at the input pin (MRST for a master device, MTSR for a slave). By this method, any corruptions on the common data exchange line are detected if the received data is not equal to the transmitted data.

The Half-Duplex Mode port configuration using three pins is shown in **Figure 902**.

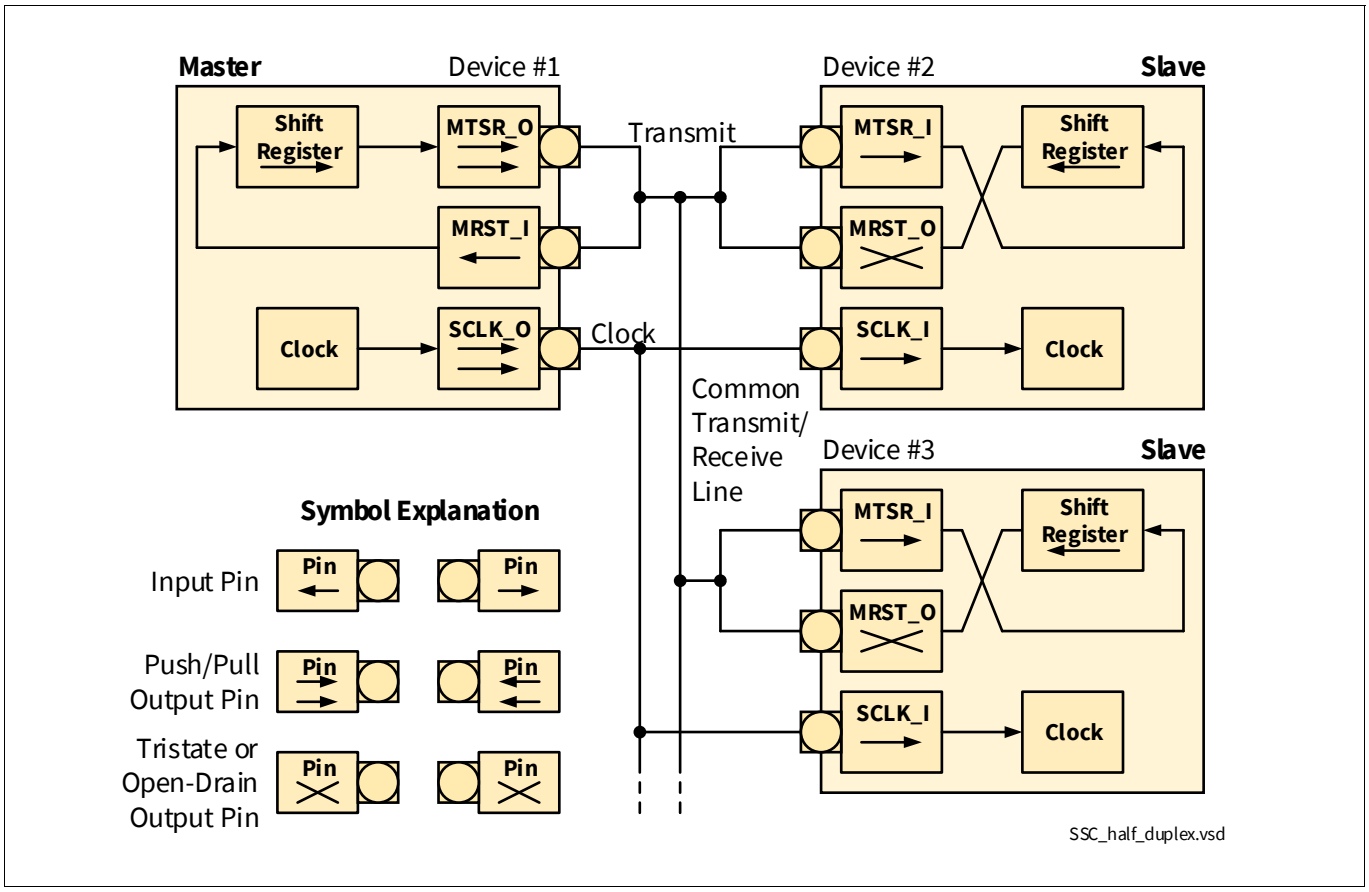


Figure 902 SSC Half-Duplex Configuration

49.15.2.4 Loop Back Mode

To aid debugging, it is useful to receive back the data which is sent out. When SSC_CONPL.LB is set, the receive input is connected to the transmit output. Hence, the data which is sent out is automatically received back. Loop-back mode is supported only in master mode.

49.15.2.5 Continuous Transfers (Master Mode only)

When the transmit interrupt request flag is set, it indicates that the transmit buffer SSC_TBL is empty and ready to be loaded with the next transmit data. If SSC_TBL has been reloaded by the time the current transmission is finished (at least 2 PCLK cycles before the last latching edge), the data is immediately transferred to the shift register and the next transmission will start without any additional delay. In such a case, there is no gap between the two successive frames on the data line. For example, two byte transfers would look the same as one word transfer. This feature can be used to interface with devices that can operate with or require more than 8 data bits per transfer. It is just a matter of software, how long a total data frame length can be. This option can also be used to interface to byte-wide and word-wide devices on the same serial bus, for instance.

Note: Of course, this can happen only in multiples of the selected basic data width, because it would require disabling/enabling of the SSC to reprogram the basic data width on-the-fly. Additionally, if the next data is now written any time after 2 PCLK cycles before the last latching edge, then there may be up to a half SCLK clock delay to start transmission of the new word.

Note: In slave mode, when SSC_CONPL.PH = 0, if the data is written into the transmit buffer at least 4 PCLK cycles before the first SS_CLK edge of the next transfer (whether shifting or latching), then continuous transfer occurs and no transmit error is generated. Otherwise, a transmit error is generated. If SSC_CONPL.PH = 1, then the data has to be provided 7 PCLK cycles before the first SS_CLK edge of the next transfer. Otherwise, a transmit error is generated.

49.15.2.6 Baud-Rate Generation

The serial channel SSC has its own dedicated 16-bit baud-rate generator with 16-bit reload capability, allowing baud-rate generation independent of the timers. **Figure 903** shows the baud-rate generator.

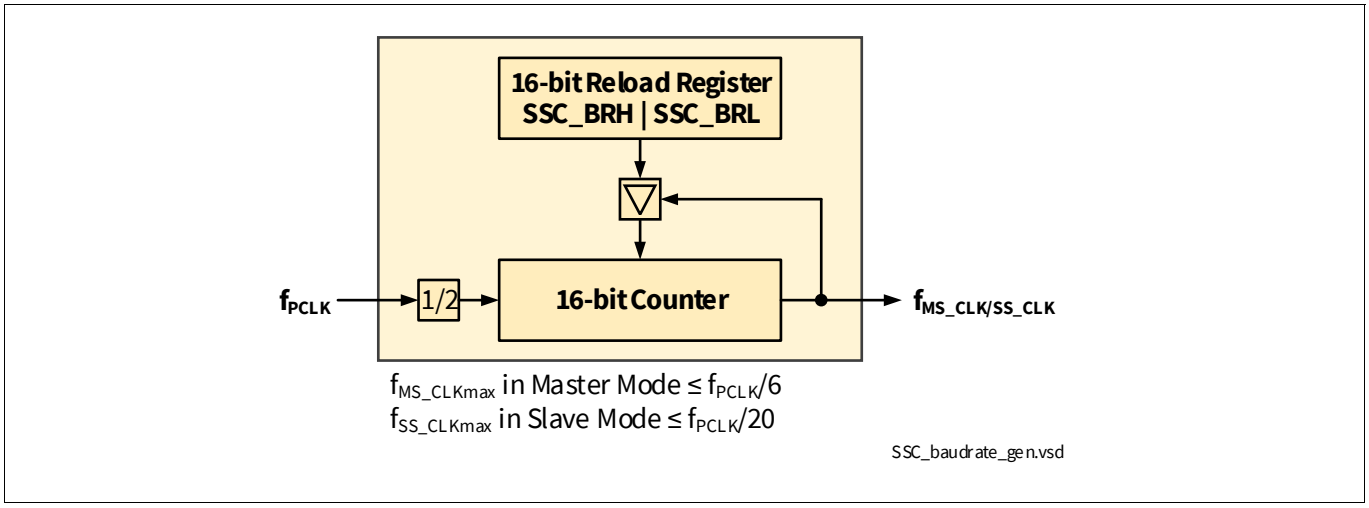


Figure 903 SSC Baud-Rate Generator

The baud-rate generator is clocked with the module clock f_{hw_clk} (which is the PCLK (fPCLK)). The timer counts downwards. On underflow, it is reloaded with the 16-bit value BR contained in register pair SSC_BRH|SSC_BRL.

Note: Never write to BR while the SSC is enabled. The desired value shall be written to BR only when SSC is disabled.

The formulas below calculate either the resulting baud-rate for a given reload value, or the required reload value for a given baud-rate:

Note: For the SSC module in the SCR, f_{hw_clk} is the PCLK (f_{PCLK}).

$$\text{Baudrate} = \frac{f_{hw\_clk}}{2 \times (
 + 1)} \qquad BR = \frac{f_{hw\_clk}}{2 \times \text{Baudrate}} - 1$$

SSC_baudrate_formula.vsd

Figure 904 SSC Baud-Rate Calculation

BR represents the contents of the reload register pair, taken as an unsigned 16-bit integer, while baud-rate is equal to f_{MS_CLK/SS_CLK} as shown in **Figure 903**.

The maximum baud-rate that can be achieved when using a module clock of 20 MHz is 3.33 MBaud in Master Mode (with BR = 0002_H) (the theoretical value is 10 MBaud, but not guaranteed) or 1 MBaud (the theoretical value is 5 MBaud, but this value is not guaranteed) in Slave Mode (with BR = 0009_H).

Table 722 lists some possible baud-rates together with the required reload values and the resulting bit times, assuming a module clock of 20 MHz.

Table 722 Typical Baud-Rates of the SSC (f_{hw_clk} = 20 MHz)

Reload Value	Baud-Rate (= f_{MS_CLK/SS_CLK})	Deviation
0000_H	not allowed	0.0%
0001_H	not allowed	0.0%
0004_H	2 MBaud (only in Master Mode)	0.0%
0009_H	1 MBaud	0.0%
0013_H	500 kBaud	0.0%
0063_H	100 kBaud	0.0%
$FFFF_H$	152.59 Baud	0.0%

49.15.2.7 Error Detection Mechanisms

The SSC is able to detect three different error conditions. Receive Error is detected in all modes; Transmit Error and Baud-Rate Error apply only to Slave Mode. When an error is detected, the respective error flag is/can be set and an error interrupt request will be generated if enabled (see Figure 905). The error interrupt handler may then check the error flags to determine the cause of the error interrupt. The error flags are not reset automatically, but must be cleared by software after servicing. This allows servicing of some error conditions via interrupt, while the others may be polled by software.

Note: The error interrupt handler must clear the associated (enabled) error flag(s) to prevent repeated interrupt requests.

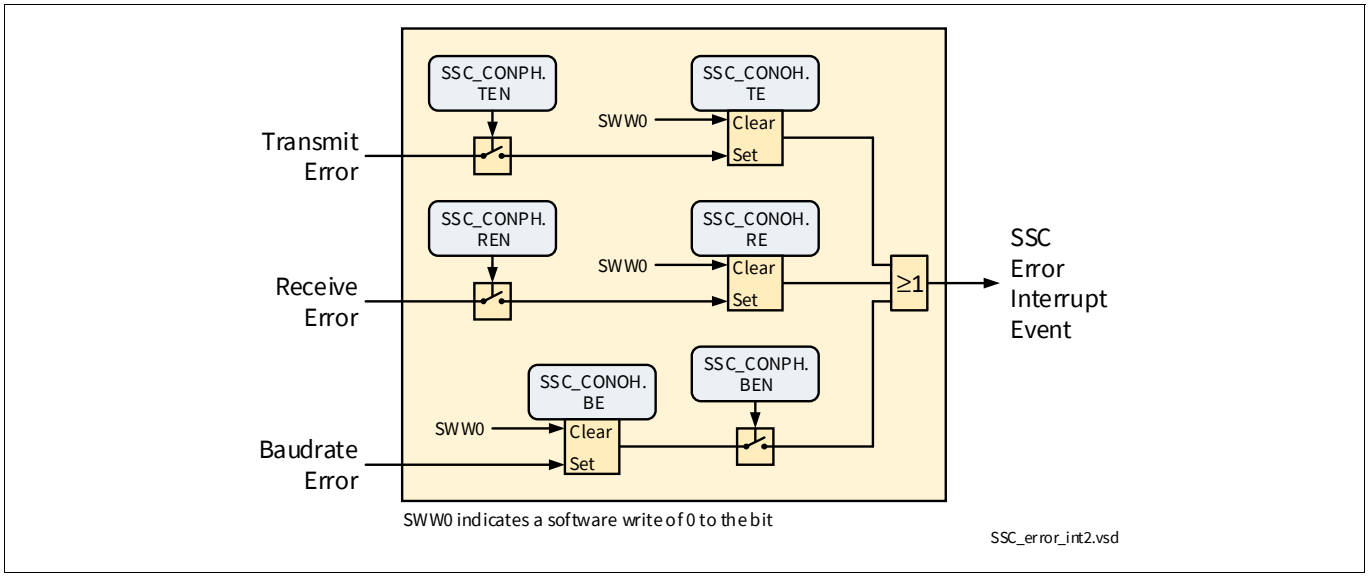


Figure 905 SSC Error Interrupt Control (interrupt pulse no signal)

A **Receive Error** (Master or Slave Mode) is detected when the FIFO is full (FIFO full already set) and another frame is completely received, but the previous data was not read out of the receive FIFO. This condition sets the error flag SSC_CONOH.RE and the error interrupt request line, when enabled via SSC_CONPH.REN. The FIFO has four elements, afterwards, the reception is stopped. FIFO full will be raised. When Receive Error is raised, an additional reception was intended, but as the FIFO is full it is lost.

A **Baud-Rate Error** (Slave Mode) is detected when the incoming clock signal deviates from the programmed baud-rate by more than 100%, i.e. it is either more than double or less than half the expected baud-rate. This condition sets the error flag SSC_CONOH.BE and, when enabled via SSC_CONPH.BEN , the error interrupt request line. Using this error detection capability requires that the slave's baud-rate generator is programmed to the same baud-rate as the master device. This feature detects false additional, or missing pulses on the clock line (within a certain frame).

A **Transmit Error** (Slave Mode) is detected when a transfer was initiated by the master (SS_CLK gets active) but the transmit buffer SSC_TBL of the slave was not updated since the last transfer. This condition sets the error flag SSC_CONOH.TE and the error interrupt request line, when enabled via SSC_CONPH.TEN. If a transfer starts while the transmit buffer is not updated, the slave may shift out invalid data. This may lead to corruption of the data on the transmit/receive line in half-duplex mode (open-drain configuration) if this slave is not selected for transmission. This mode requires that slaves not selected for transmission only shift out ones; that is, their transmit buffers must be loaded with '$FFFF_H$' prior to any transfer.

Note: In order to avoid possible conflicts or misinterpretations, it is recommended to always load the slave's transmit buffer prior to any transfer.

The cause of an error interrupt request (receive, baud-rate, transmit error) can be identified by the error status flags in control register SSC_CONOH.

Note: The error status flags SSC_CONOH.TE , SSC_CONOH.RE, and SSC_CONOH.BE are not reset automatically upon entry into the error interrupt service routine, but must be cleared by software.

49.15.3 SSC Interrupts

The five SSC interrupts can be separately enabled or disabled by setting or clearing their corresponding enable bits in SFR SCU_MODIEN.

For a detailed description of the various interrupts see **Section 49.15.2**. An overview is given in **Table 723**.

Table 723 SSC Interrupt Sources

Interrupt	Signal	Description
Transmission starts	TIR	Indicates that the transmit buffer can be reloaded with new data.
Transmission ends	RIR	The configured number of bits has been transmitted and shifted to the receive buffer. This is only the case for the first reception after FIFO empty.
Receive FIFO full	RFI	The receive FIFO is full and will stop reception. Further information is lost.
Receive FIFO empty	REI	The receive FIFO is empty, further access will provide junk data.
Receive Error	EIR	This interrupt occurs if a new data frame is completely received while the FIFO is full.
Baud-Rate Error (Slave Mode only)	EIR	This interrupt is generated when the incoming clock signal deviates from the programmed baud-rate by more than 100%.
Transmit Error (Slave Mode only)	EIR	TE is generated in case corrupted data is sent. This can be an empty transmit buffer or corrupted data on the serial bus (in case the transmit buffer is overwritten round about transmission start).

Figure 906 shows a diagram of the interrupt request handling of the SSC.

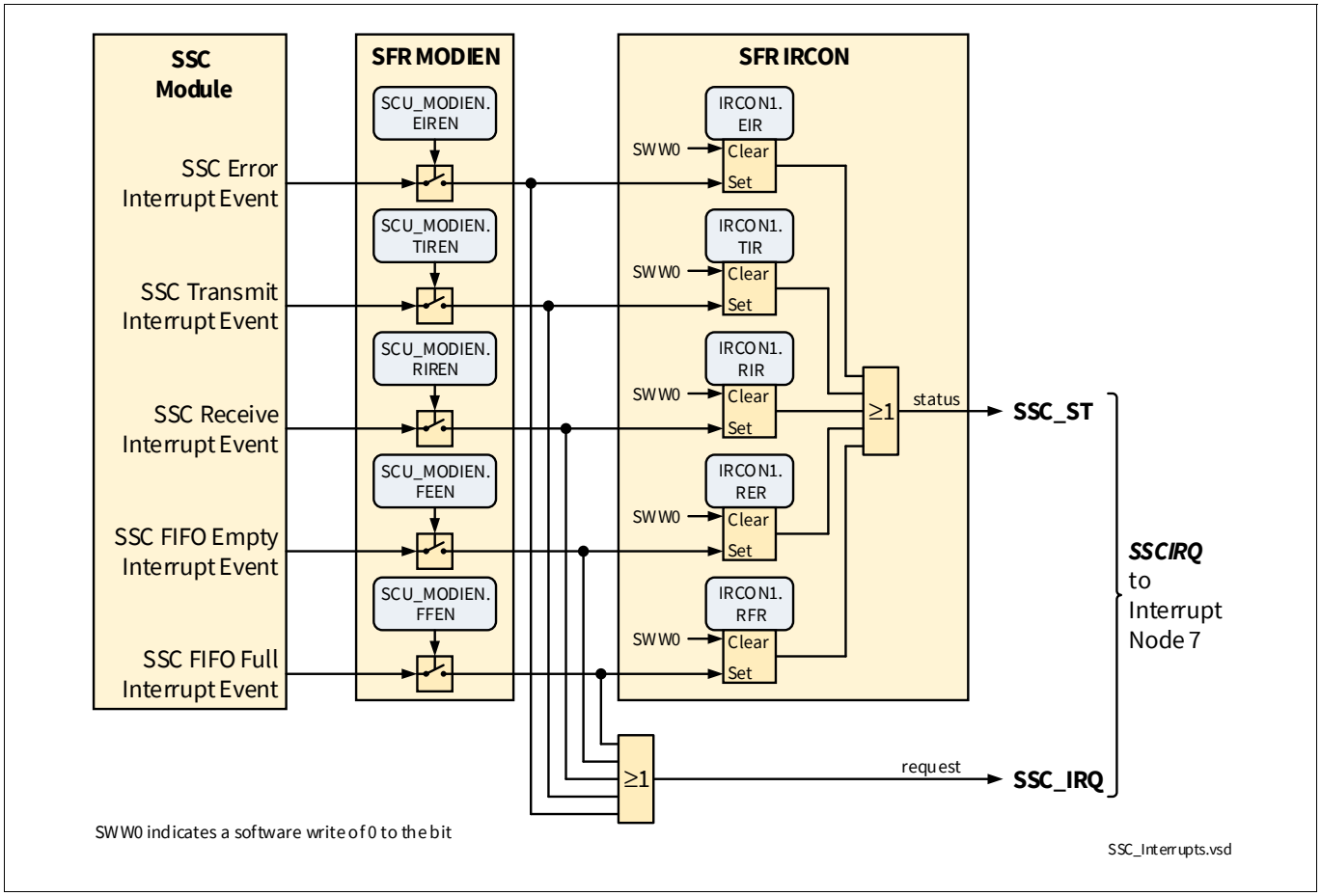


Figure 906 SSC Interrupt Request Overview

49.15.4 SSC Connections to GPIO

The following figure illustrates the signal connections of the SSC to the GPIO ports.

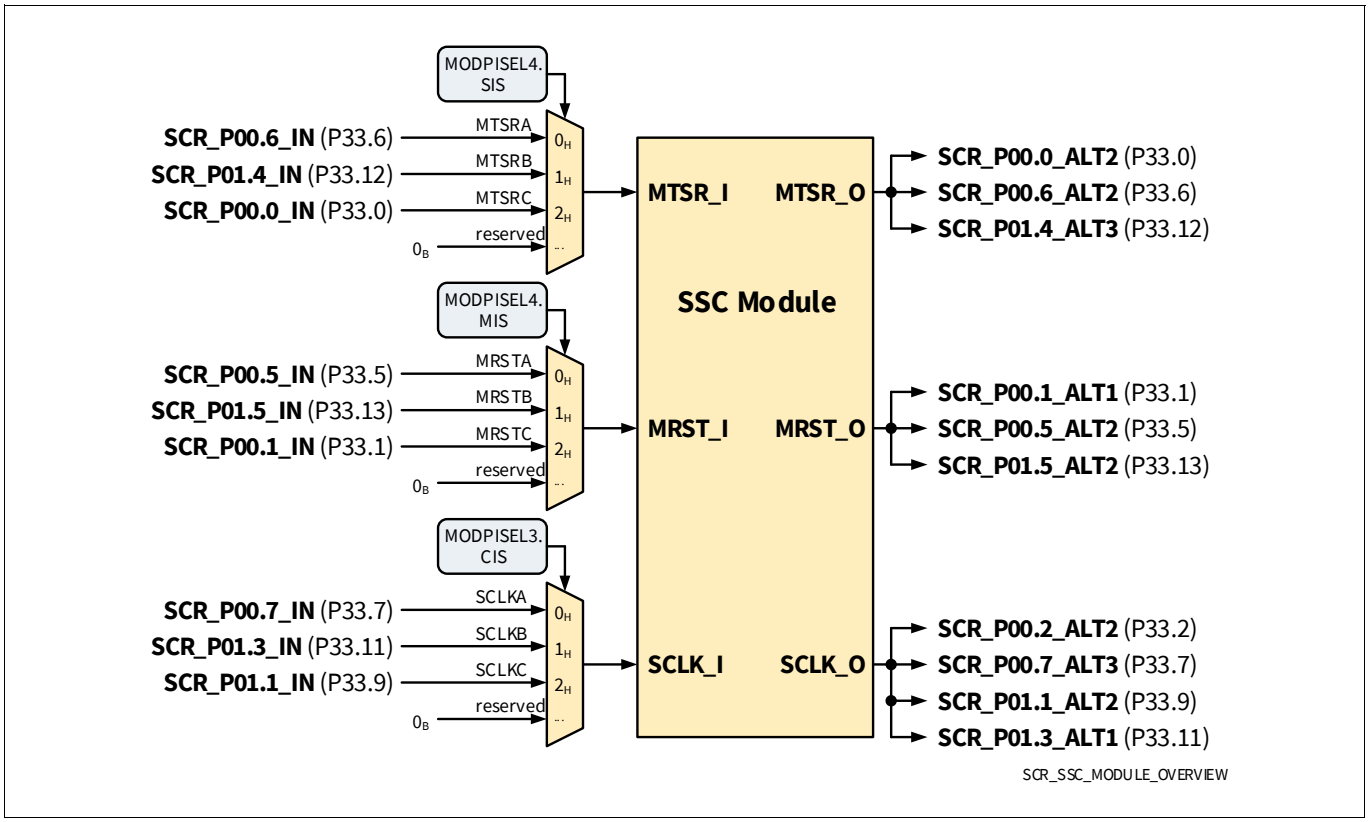


Figure 907 SSC Connections to GPIO Ports

49.15.5 Register Description

The SSCSpecial Function Registers are accessed from the standard (non-mapped) SFR area. The addresses of the SFRs are listed in **Table 724**.

Table 724 Register Overview - SSC (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
SSC_BRH	Baud Rate Timer Reload Register High	0	X	$08E_H$	**303**
SSC_BRL	Baud Rate Timer Reload Register Low	0	X	$08D_H$	**303**
SSC_CONOH	Control Register High [Operation Mode]	0	X	$08A_H$	**301**
SSC_CONOL	Control Register Low [Operation Mode]	0	X	089_H	**300**
SSC_CONPH	Control Register High [Programming Mode]	0	X	$08A_H$	**299**
SSC_CONPL	Control Register Low [Programming Mode]	0	X	089_H	**298**
SSC_RBL	Receiver Buffer Register	0	X	$08C_H$	**304**
SSC_TBL	Transmitter Buffer Register	0	X	$08B_H$	**304**

49.15.5.1 Configuration Register

The operating mode of the serial channel SSC is controlled by two register pairs, sharing the two same addresses:

- Register pair SSC_CONPH/SSC_CONPL provides access to a set of control bits, and is accessible in Programming Mode.
- Register pair SSC_CONOH/SSC_CONOL provides access to a set of control/status flags, and is accessible in Operating Mode.

The selection between the register pairs is depending on the state of the enable bit EN prior to an access to the register. This bit is available at the MSB position in both registers, SSC_CONPH and SSC_CONOH:

- When EN = 0 (Programming Mode), registers SSC_CONPH and SSC_CONPL are accessible.
- When EN = 1 (Operating Mode), registers SSC_CONOH and SSC_CONOL are accessible.

Since the selection is dependent on the state of the EN bit prior to an access, writing, for example, $1100\ 0000_B$ to SSC_CONPH in programming mode (SSC_CONPH.EN = 0) will initialize the SSC (SSC_CONPH.EN was 0) and then turn it on. Then, register pair SSC_CONOH | SSC_CONOL is accessible, with SSC_CONOH.EN = 1.

Control Register Low [Programming Mode]

Note: When initializing the SSC, register SSC_CONPL must be written first, before writing to register SSC_CONPH and setting bit SSC_CONPH.EN to 1.

SSC_CONPL
Control Register Low [Programming Mode] (089_H) **Reset Value: Table 725**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
LB	**PO**	**PH**	**HB**	**0**	**BM**		
rw	rw	rw	rw	r	rw		

Field	Bits	Type	Description
BM	2:0	rw	**Data Width Selection** *Note: The transfer and receive data width is maximum 8 bits. Transfer Data Width is 4...8 bits (<BM>+1).* 000_B Reserved. Do not use this combination. … 010_B Reserved. Do not use this combination. 011_B Transfer Data Width is 4 bits 100_B Transfer Data Width is 5 bits 101_B Transfer Data Width is 6 bits 110_B Transfer Data Width is 7 bits 111_B Transfer Data Width is 8 bits
HB	4	rw	**Heading Control** 0_B **LSB_FIRST**, Transmit/Receive LSB First 1_B **MSB_FIRST**, Transmit/Receive MSB First

Field	Bits	Type	Description
PH	5	rw	**Clock Phase Control** 0_B **LATCH_ON_TRAILING_EDGE**, Shift transmit data on the leading clock edge, latch on trailing edge. 1_B **LATCH_ON_LEADING_EDGE**, Latch receive data on leading clock edge, shift on trailing edge.
PO	6	rw	**Clock Polarity Control** 0_B **IDLE_LOW**, Idle clock line is low, leading clock edge is low-to-high transition. 1_B **IDLE_HIGH**, Idle clock line is high, leading clock edge is high-to-low transition.
LB	7	rw	**Loop Back Control** This bit shall be set by software only in Master mode. 0_B Normal output 1_B Receive input is connected with transmit output (half-duplex mode)
0	3	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 725 Reset Values of SSC_CONPL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Control Register High [Programming Mode]

SSC_CONPH
Control Register High [Programming Mode] ($08A_H$) **Reset Value: Table 726**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
EN	**MS**	**0**	**0**	**BEN**	**0**	**REN**	**TEN**
rw	rw	r	r	rw	r	rw	rw

Field	Bits	Type	Description
TEN	0	rw	**Transmit Error Enable** 0_B Ignore transmit errors 1_B Check transmit errors
REN	1	rw	**Receive Error Enable** 0_B Ignore receive errors 1_B Check receive errors
BEN	3	rw	**Baudrate Error Enable** 0_B Ignore baudrate errors 1_B Check baudrate errors

Field	Bits	Type	Description
MS	6	rw	**Master Select** This bit shall not be changed during operation mode. 0_B **SLAVE**, Slave Mode. Operate on shift clock received via SCLK. 1_B **MASTER**, Master Mode. Generate shift clock and output it via SCLK.
EN	7	rw	**Enable Bit = 0 (needed for programming mode)** Transmission and reception disabled. Access to control bits.
0	2, 4, 5	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 726 Reset Values of SSC_CONPH

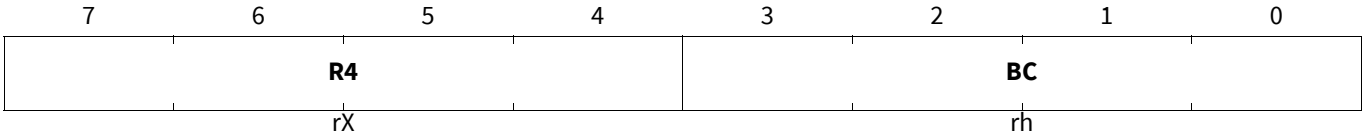
Reset Type	Reset Value	Note
LVD Reset	40_H	
Generated Reset	40_H	

Remark on SSC_CONPH.EN

Note: Between programming SSC_CONPH.EN=1 and writing into SSC_TBL a minimum delay of 4 cycles has to be kept. (see SSC_TBL)

Control Register Low [Operation Mode]

SSC_CONOL
Control Register Low [Operation Mode] (089_H) Reset Value: Table 727
RMAP: 0, PAGE: X



Field	Bits	Type	Description
BC	3:0	rh	**Bit Count Field** This field is loaded with the programmed data width value (BM) when the SSC is enabled. It then counts down with every shift bit. *Note: This bit field is not to be written to.*
R4	7:4	rX	**Reserved**

Table 727 Reset Values of SSC_CONOL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Control Register High [Operation Mode]

Note: Before disabling the SSC by clearing bit SSC_CONOH.EN during operation, received data has to be read out of the FIFO, as otherwise the software can read corrupted data.

SSC_CONOH
Control Register High [Operation Mode] ($08A_H$) Reset Value: Table 728
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
EN	**MS**	**RFF**	**BSY**	**BE**	**RFE**	**RE**	**TE**
rw	rw	rh	rh	rwh	rh	rwh	rwh

Field	Bits	Type	Description
TE	0	rwh	**Transmit Error Flag** 0_B No error 1_B Transfer starts with the slave's transmit buffer not being updated
RE	1	rwh	**Receive Error Flag** 0_B No error 1_B Reception in FIFO is blocked
RFE	2	rh	**Receive FIFO Buffer Empty Flag** 0_B Receive FIFO contains data 1_B Receive FIFO is empty
BE	3	rwh	**Baudrate Error Flag** 0_B No error 1_B More than factor 2 or 0.5 between slave's actual and expected baudrate
BSY	4	rh	**Busy Flag** Set while a transfer is in progress. Note: This bit is not to be written to. 0_B No transfer in progress 1_B Transfer in progress
RFF	5	rh	**Receive FIFO Buffer Full Flag** *Note: FIFO full will be deleted as soon as the receive buffer is read by software.* 0_B Receive FIFO is not full 1_B Receive FIFO is full
MS	6	rw	**Master Select** This bit shall not be modified during operation mode. 0_B Slave Mode. Operate on shift clock received via SCLK. 1_B Master Mode. Generate shift clock and output it via SCLK.
EN	7	rw	**Enable Bit = 1 (needed for operating mode)** Transmission and reception enabled. Access to status flags and M/S control.

Table 728 Reset Values of SSC_CONOH

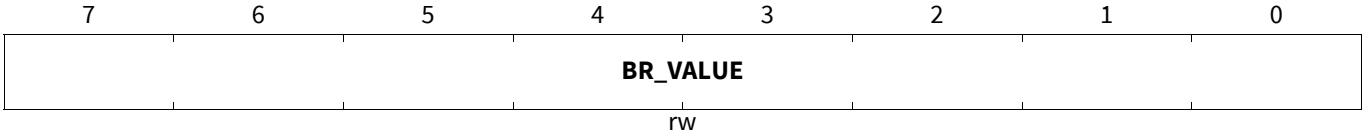
Reset Type	Reset Value	Note
LVD Reset	40_H	
Generated Reset	40_H	

49.15.5.2 Baud-Rate Timer Reload Register

The SSC baud-rate timer reload register pair SSC_BRH | SSC_BRL contains the 16-bit reload value BR for the baud-rate timer.

Baud Rate Timer Reload Register Low

SSC_BRL
Baud Rate Timer Reload Register Low (08D$_H$) Reset Value: Table 729
RMAP: 0, PAGE: X



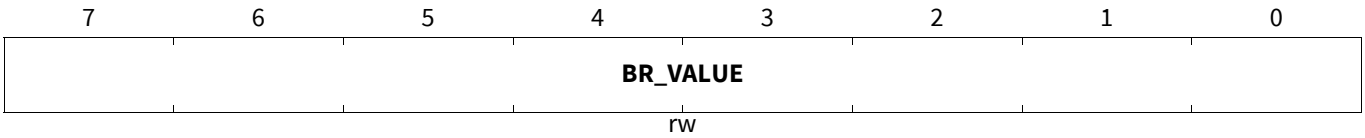
Field	Bits	Type	Description
BR_VALUE	7:0	rw	**Baud Rate Timer Reload Register Value, low byte** Represents the lower 8 bits of the 16-bit baudrate reload value.

Table 729 Reset Values of SSC_BRL

Reset Type	Reset Value	Note
LVD Reset	00$_H$	
Generated Reset	00$_H$	

Baud Rate Timer Reload Register High

SSC_BRH
Baud Rate Timer Reload Register High (08E$_H$) Reset Value: Table 730
RMAP: 0, PAGE: X



Field	Bits	Type	Description
BR_VALUE	7:0	rw	**Baud Rate Timer Reload Register Value, high byte** Represents the upper 8 bits of the 16-bit baudrate reload value.

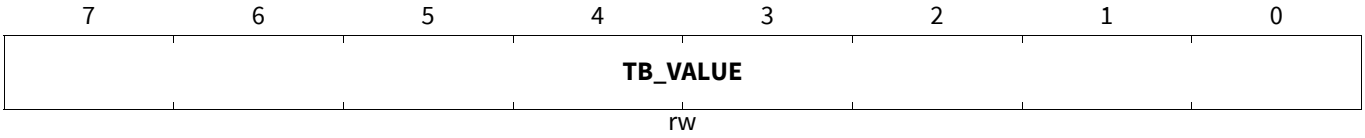
Table 730 Reset Values of SSC_BRH

Reset Type	Reset Value	Note
LVD Reset	00$_H$	
Generated Reset	00$_H$	

49.15.5.3 Transmit Buffer Register

Transmitter Buffer Register

SSC_TBL
Transmitter Buffer Register **($08B_H$)** **Reset Value: Table 731**
RMAP: 0, PAGE: X



Field	Bits	Type	Description
TB_VALUE	7:0	rw	**Transmit Data Register Value** TB_VALUE is the data value to be transmitted. Unselected bits of TB_VALUE are ignored during transmission.

Table 731 Reset Values of SSC_TBL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Note on programming SSC_TBL

Note: Before writing to SSC_TBL a minimum delay of 4 PCLK cycles has to be regarded after writing SSC_CONPH.EN=1.
e.g. SSC_CON_PH.EN=1;
for (i=0;i<4;i++) nop();
SSC_TBL=Byte to be transfered
Instead of nop() more usefull code can be used, as long as a delay of 4 cycles is guaranteed.

Recommendation in case of continuous transfer

It is recommended to use the the transmit interrupt as signal to write the next piece of information to the transmit buffer register. This way, it is guaranteed, that the module has the information on time (in slave mode) to guarantee the transmission.

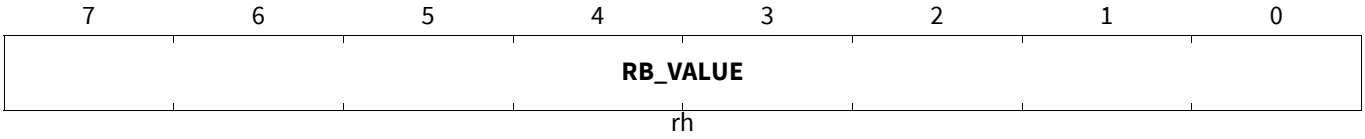
49.15.5.4 Receive Buffer Register

The SSC receive buffer register SSC_RBL contains the received data value. It is the access point to the FIFO. The FIFO depth is 4 bytes. The FIFO stops receiving, when full. Warning flags for FIFO full (RFF) and FIFO empty (RFE) exist in SSC_CONOH.

Receiver Buffer Register

Note: Before disabling the SSC by clearing bit SSC_CONOH.EN during operation, received data has to be read out of the FIFO, as otherwise the software can read corrupted data.

SSC_RBL
Receiver Buffer Register **($08C_H$)** **Reset Value: Table 732**
RMAP: 0, PAGE: X



Field	Bits	Type	Description
RB_VALUE	7:0	rh	**Receive Data Register Value** SSC_RBL contains the received data value RB_VALUE. Unselected bits of RB_VALUE will be not valid and should be ignored. Access point to FIFO. In case of FIFO overflow, the receive FIFO is blocked.

Table 732 Reset Values of SSC_RBL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.15.6 Revision History

Table 733 Revision History

Reference	Change to Previous Version	Change Request
v4.4		
	First Official Release of completely reworked SCR chapter	
	No change	

49.16 ADC Comparator Unit (ADCOMP)

The ADCOMP constitutes an 8-bit successive approximation Analog-to-Digital converter, supporting up to 4 channels. The ADCOMP unit is periodically activated during STANDBY mode to measure certain analog inputs, and is characterized by short activation periods after which the unit is powered off to keep the overall STANDBY current to the minimum. The conversion control involves enabling the ADC Comparator unit, selecting the channel, and subsequently issuing start of conversion via software with the support of a timer unit. After conversion is completed, end of conversion event is indicated and an interrupt may be generated.

49.16.1 Features

The ADCOMP has the following features:

- 8-bit resolution. LSB = 23.077 mV. Range = 0 - 5861.54 mV. VIN = [LSB * (ADCRES-1)]
- Accuracy / TUE = +-4 LSB
- Periodic monitoring of up to 4 ADC channels (CH0: SCR_P00.4, CH1: SCR_P00.5, CH2: SCR_P00.6 and CH3: SCR_P00.7). For the actual mapping of these SCR Ports to the ports of the device, please refer to the SCR GPIO chapter.
- Minimum 200 ns sampling time + 400 ns conversion time. Higher sampling time to be realized using timer (maximum sampling time is limited to 800 ns).

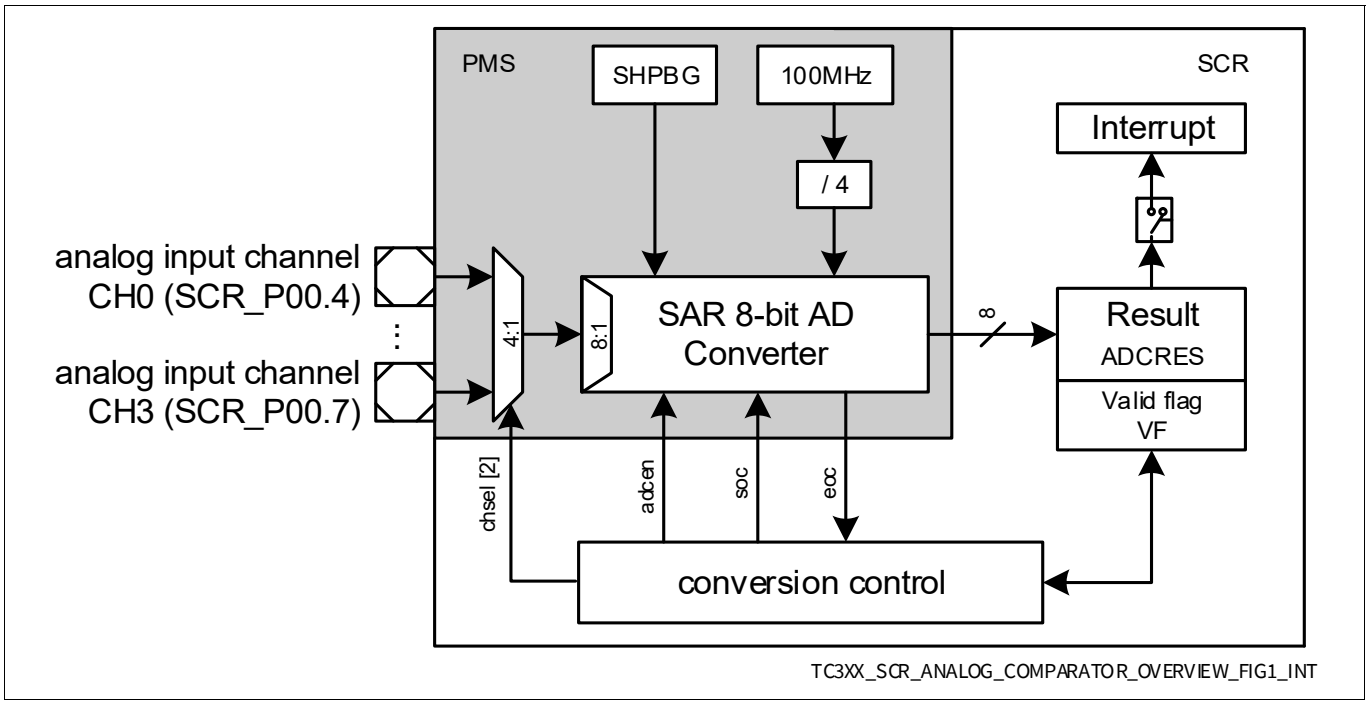


Figure 908 ADC Comparator Overview

49.16.2 Overview

The ADC unit is clocked with 25 MHz clock derived from the 100 MHz back-up clock. The reference voltage for the conversion is derived from the secondary high-precision bandgap. The analog channels are selected via a 4:1 multiplexer. The AD converter is primarily used as a secondary voltage monitor during normal RUN operation of TC3xx, and therefore has an inherent 8:1 multiplexer to route the various supply inputs. During RUN mode, the ADCOMP unit cannot be activated in SCR, and ADCENREQ is ignored. The ADCOMP unit may be enabled via ADCENREQ bit only during STANDBY mode, as during normal operation the unit is used for voltage monitoring. In normal mode, the ADCOMP channels may be converted using the main VADC unit. The conversion control and

result handling part is encapsulated in the SCR digital domain, and is clocked via the PCLK clock derived from the 100 MHz back-up clock. Requests from the SCR during RUN mode of the device will be ignored.

49.16.3 ADC Comparator Operation

In STANDBY mode, the Bandgap and the ADC unit are switched off by default. Before an ADC conversion is triggered, SHPBG Bandgap, ADC unit and 100 MHz back-up clock need to be started. The ADC needs to be enabled via bit ADCOMP_CON.ADCENREQ. The activation of the ADCOMP unit will in turn start the Bandgap, the 100 MHz clock and the SAR ADC unit for the time of conversion and a certain stabilization time need to be waited. In case the PMS is not in STANDBY mode, ADCENREQ is ignored. Once the sub-units are in operational state, ADCOMP_CON.ADCEN and ADCOMP_CON.EOCSAMPLE bits are set to indicate that a start of conversion may be triggered. The conversion and sampling phases are carried out back to back, as shown in **Figure 909**. Foremost, the required channel is selected via ADCOMP_CON.ADCCHSEL. The sample time is defined by the time between End of Conversion (EOCSAMPLE) and the Start of Conversion (SOC). The first conversion is a dummy conversion and is triggered by setting ADCOMP_CON.SOC start of conversion bit. The conversion phase is subsequently started, which takes 10 (EVR) clock cycles resulting into 400 ns in case of a trimmed EVR oscillator. After the conversion phase is completed, bit ADCOMP_CON.EOCSAMPLE is set. EOCSAMPLE triggers an interrupt if activated via ADCOMP_CON.ADCIEN enable bit. The converted result is transferred to ADCOMP_RES result register, and the valid flag ADCOMP_CON.VF is set. This valid flag indicates that a “new” valid data has been stored in the corresponding result register, and can be read out. The valid flag is automatically cleared when the next SOC is given.

The previous end of conversion automatically triggers the next sampling phase. The start of the sampling phase is not under the control of the software, but only the end of the sampling phase can be controlled via the SOC bit. Adequate time need to be waited after EOCSAMPLE to provide the required sample time. The sample time may be realized by using one of the SCR timers. A minimum sample time of 200 ns is required, and may extend up to 800 ns, depending on the external sensor. The sample phase is indicated by bit ADCOMP_CON.EOCSAMPLE. It is intended that the result is read before the subsequent SOC conversion event. When the SOC is given, the result may no longer be valid, and hence the ADCOMP_CON.VF is cleared.

SOC request is not taken if EOCSAMPLE flag is currently not set. Once the sample time has elapsed, a new start of conversion request can be triggered by software. After the conversion is completed, ADCOMP unit may be disabled by clearing ADCOMP_CON.ADCENREQ register bit.

In case ADCENREQ is cleared during ongoing conversion, then the ADC is disabled only after the currently running conversion is ended.

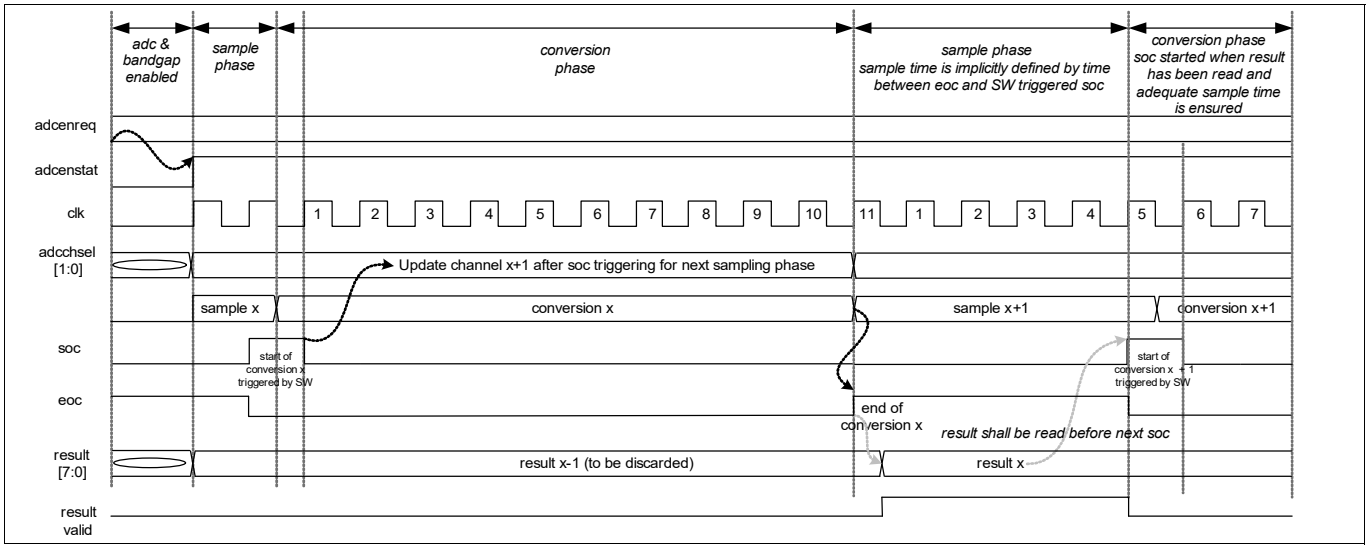


Figure 909 ADC comparator signal diagram

In case of a STANDBY wake-up request, an ongoing conversion of ADCOMP is completed. Consequently, as a wakeup request has been raised, the PMS is woken up, and the control of AD converter is given over to the PMS. This inherently results into a worst case delay of 400 ns (10 EVR clock cycles).

In case of a reset event during STANDBY, the ADCOMP operation is aborted. Consequently, the control is transferred to the PMS on re-start after reset. In case of an SCR reset, the ADCOMP is disabled.

49.16.4 ADCOMP Interrupt Request

Figure 910 shows a diagram of the interrupt request handling of the ADCOMP module.

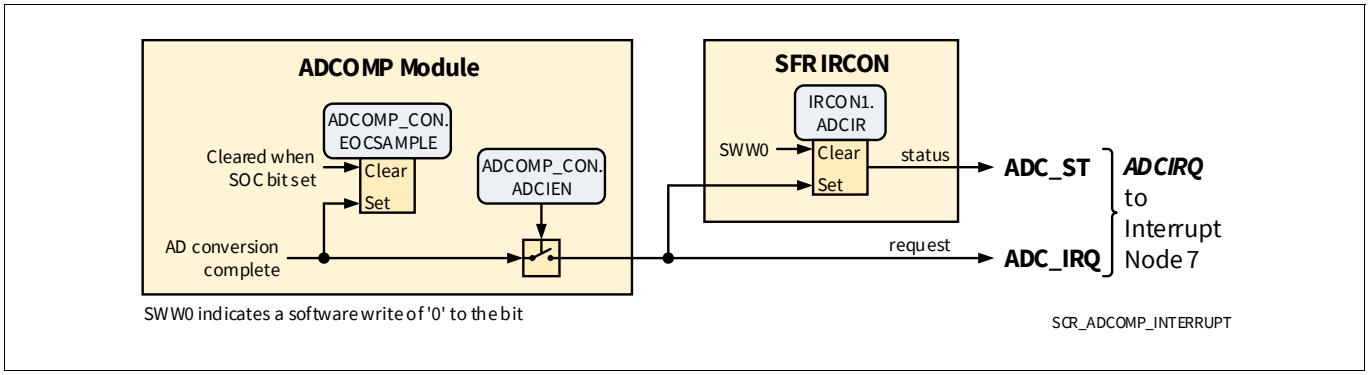


Figure 910 ADCOMP Interrupt Request Overview

49.16.5 ADCOMP Connections to GPIO

The following figure illustrates the signal connections of the ADCOMP module to the GPIO inputs.

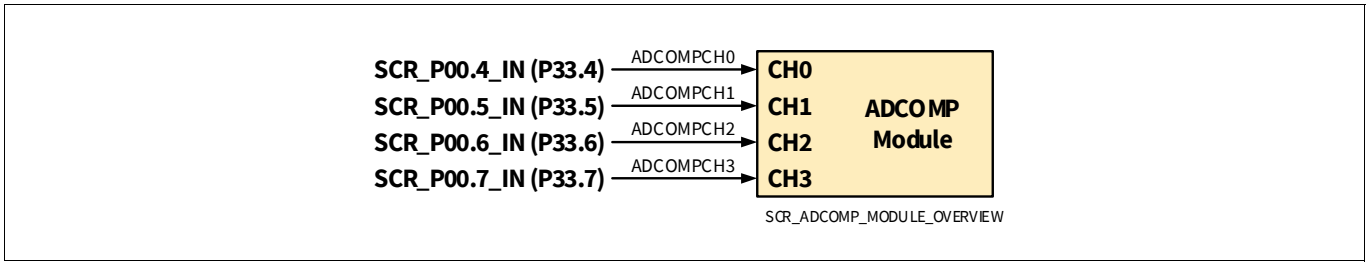


Figure 911 ADCOMP to GPIO Connection Overview

49.16.6 ADCOMP Registers

This section describes all ADCOMP special function registers. The ADCOMP and interrupt related SFRs are also listed, as shown in **Table 734**.

Note: All ADCOMP register names shall be referenced fully with the module name prefix “ADCOMP_”.

Note: The ADCOMP SFRs must be accessed via the paging mechanism at SCU Page 0.

Table 734 Register Overview - ADCOMP (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
ADCOMP_CON	ADCOMP Control Register	0	0	$0FB_H$	311
ADCOMP_RES	ADCOMP Result Register	0	0	$0FA_H$	312

49.16.6.1 ADCOMP Registers Description

The following describes the Special Function Registers of the ADCOMP kernel.

ADCOMP Control Register

ADCOMP_CON
ADCOMP Control Register **($0FB_H$)** **Reset Value: Table 735**
RMAP: 0, PAGE: SCU_PAGE=0

7	6	5	4	3	2	1	0
SOC	**EOCSAMPLE**	**VF**	**ADCCHSEL**		**ADCIEN**	**ADCEN**	**ADCENREQ**
rwh	rh	rh	rw		rw	rh	rw

Field	Bits	Type	Description
ADCENREQ	0	rw	**ADC Comparator Enable Request** This bit activates the ADC unit, the associated bandgap and clock required for the ADC unit. It is required to wait some time for the units to start after the request. In case ADCENREQ is cleared during ongoing conversion, then the ADC is disabled when the currently running conversion is ended. 0_B ADCOMP unit is disabled 1_B ADCOMP unit is enabled
ADCEN	1	rh	**ADC Comparator Enable Status** This bit indicates that the ADC unit, the associated bandgap, and the clock required for the ADC unit are active, and a subsequent conversion can be triggered. It is set after some time after ADCOMP_CON.ANCENREQ is set. 0_B ADCOMP unit is disabled 1_B ADCOMP unit is enabled
ADCIEN	2	rw	**ADC Comparator Interrupt Enable** Software can read this bit to get the status whether the interrupt is currently enabled or disabled. 0_B No interrupt is generated 1_B Interrupt is generated after conversion
ADCCHSEL	4:3	rw	**ADC Channel Select** This bitfield selects the channel and configures the 4:1 ADC multiplexer before a conversion is triggered. After a channel is selected, adequate time needs to be provided for sampling before a conversion is started. 00_B **CH0**, CH0 is selected 01_B **CH1**, CH1 is selected 10_B **CH2**, CH2 is selected 11_B **CH3**, CH3 is selected
VF	5	rh	**ADC Valid Flag for Result Register** Indicates a new result in ADCOMP_RES register. 0_B There is no valid result 1_B A new result is available in the ADCOMP_RES result register. The valid flag is automatically cleared when bit SOC is set.

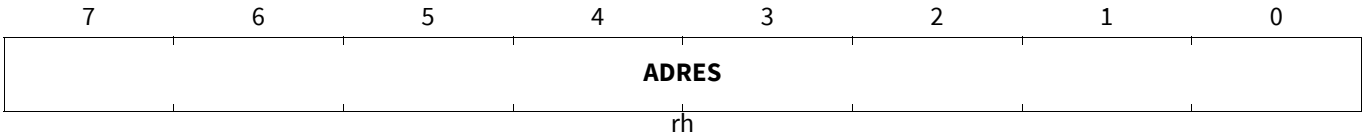
Field	Bits	Type	Description
EOCSAMPLE	6	rh	**ADC End of Conversion / Sampling Indication** This bit indicates that a conversion is completed and the result can be read. A new conversion via SOC can be started only when the EOC from the previous conversion has been set. This bit is automatically cleared when SOC is set. 0_B ADC conversion is currently going on 1_B ADC conversion is completed. ADC is currently sampling.
SOC	7	rwh	**ADC Start of Conversion Request** This bit activates a new conversion. Setting this bit has effect only when both, ADCEN and EOC, are set. This bit is automatically cleared by hardware once the conversion has started. If VF is set, setting SOC to 1 clears VF. 0_B ADC conversion is not started 1_B ADC conversion is started

Table 735 Reset Values of ADCOMP_CON

Reset Type	Reset Value	Note
LVD Reset	40_H	
Generated Reset	40_H	

ADCOMP Result Register

ADCOMP_RES
ADCOMP Result Register **($0FA_H$)** **Reset Value: Table 736**
RMAP: 0, PAGE: SCU_PAGE=0



Field	Bits	Type	Description
ADRES	7:0	rh	**ADC Conversion Result** This register shows the current converted ADC result. Software should ensure that the result is read before starting the next conversion. VIN = [LSB * (ADCRES-1)] ; LSB = 23.077 mV. Full Range : 5861.54 mV. For results of VIN =< 1, VIN is set to 0 V. Otherwise, VIN = [LSB * (ADCRES-1)].

Table 736 Reset Values of ADCOMP_RES

Reset Type	Reset Value	Note
LVD Reset	$F0_H$	
Generated Reset	$F0_H$	

49.16.7 Revision History

Table 737 Revision History

Reference	Change to Previous Version	
v4.3		
	First Official Release of completely reworked SCR chapter	
	No change	

49.17 Wake-Up CAN (WCAN) Filter

This chapter describes the WCAN controller of the SCR. It contains the following sections:

- Overview on the Wake-Up capabilities (see **Page 314**)
- Functional description of the WCAN kernel (see **Page 317**)
- WCAN Kernel register description (see **Page 318**)
- Functional description on the CAN based kernel (see **Page 340**)
- SCR implementation specific details and registers of the WCAN (port connections and control, interrupt control, address decoding, clock control (see **Page 356**)

Note: The WCAN kernel register names described in this chapter will be referenced in the Standby Domain Internal Target Specification by the module name prefix “WCAN_”.

49.17.1 Definitions and Abbreviations

- WUF - Wake-Up Frame
- WUP - Wake-Up Pattern; Wake-Up Pattern detection has to be supported by external CAN transceiver and WUP detection has to be notified by transceiver to WCAN by a falling edge on RXD. To keep the context simple, the term Wake-Up Pattern is used in WCAN specification.

49.17.2 Wake-Up CAN Filter Implementation

49.17.2.1 Overview

This section describes the 11898-6 datalink layer. The Wake-Up CAN module contains 1 CAN node, representing only the receive part of the communication interface. Furthermore, the Wake-Up CAN filter functionality and interfaces in AURIX™ TC3xx Platform.

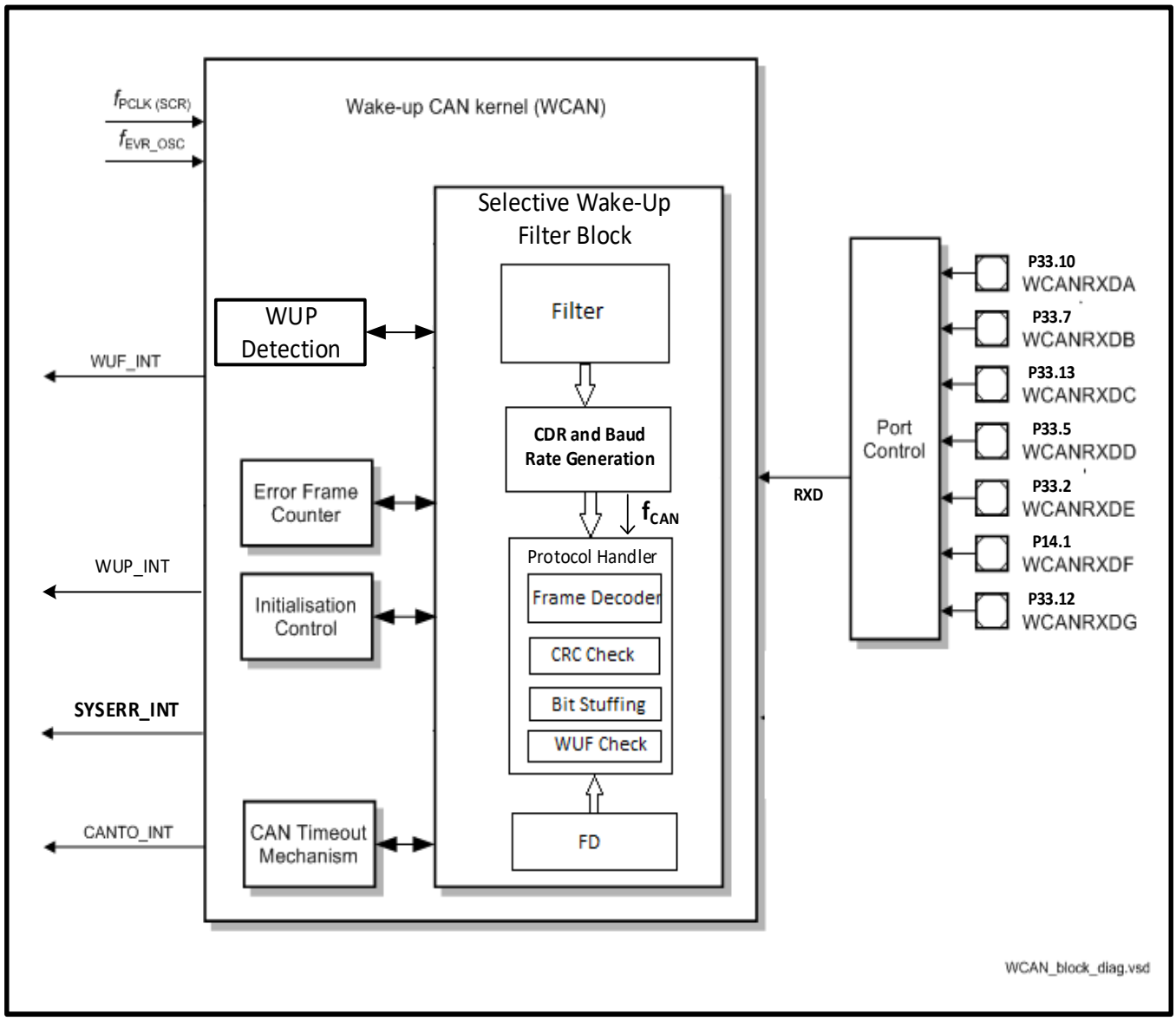


Figure 912 Wake-Up CAN Filter Interfaces

49.17.2.2 WCAN Filter Features

The following are the feature list for WCAN module in SCR:

- Wake-Up CAN functionality in conformance to ISO 11898-6
- 1 Wake-Up CAN node
- Guaranteed data transfer rate up to 500 kBaud, baudrates higher are dependent on customer settings. Default settings for 125 kBaud, 250 kBaud and 500 kBaud are according to the given propagation delays of German OEMs.
- Oscillator tolerance requirement (+/- 4%) improved with Clock and Data Recovery (CDR)
- 1 Wake-up frame configuration registers to the wake-up node
  - Set up to handle frames with 11-bit and/or 29-bit identifiers
  - Programmable acceptance mask register for filtering
  - Programmable wake-up message ID, DLC and Data
- Wake-Up CAN message Time-out feature
- Power saving through Standby mode on CAN message Time-out
- WUP detection by monitoring falling edge on RXD when in standby mode
- Tolerant to CAN FD frames
- Error counter to debounce CAN protocol and wake-up function errors
- Interrupt generation on
  - Successful Wake-up message frame reception
  - Error recognition
  - Wake-Up CAN timeout
  - WUP (falling edge on RXD) detection

49.17.2.3 Introduction

49.17.2.3.1 Feature Overview

Reception and evaluation of CAN frames is handled in accordance to CAN specification ISO 11898-6 data link layer. The received messages will be compared against the programmed wake-up pattern and in case of pattern match, a wake-up will be issued. In case of an erroneous understanding or errors on the bus, the receive error counter can also reach a wake-up threshold. The node will be purely listening to the bus and will not communicate to the outside.The CAN node has Wake-Up Frame (WUF) configuration registers. In these WUF registers, the identifier including a bitwise mask, the corresponding DLC as well as a data mask is included. Only (outside the erroneous case), if all criteria match, a wake-up will be initiated.The bit timings for the CAN nodes are derived from the CAN clock (fCAN) and are programmable up to a data rate of 1 MBaud, nonetheless guaranteed wake-up is only given for special configurations according to the propagation delay defined by the German OEMs.

Features

- Compliant to ISO 11898-6.
- Receive operation according to 11898-6.
- Tolerant to CAN FD frames
- Dedicated control registers for the CAN node.Data transfer rates are guaranteed until 500 kBaud, in case of higher baudrates more wake-up frames might be needed.Transmitter tolerances up to 0.4% are calculated and guaranteed with CDR enabled. Higher transmitter tolerances are not pretested.
  - The message buffer can be configured to accept only standard or only extended Classical frames as valid Wake-Up frame.
- Four interrupt sources do exist, wake-up frame found, tSILENCE expired, error overflow and WUP detected.

49.17.2.4 WCAN Module Control Registers

Wake-Up CAN Configuration Register

WCAN_CFG
Wake-Up CAN Configuration Register **($0B0_H$)** **Reset Value: Table 738**
RMAP: 0, PAGE: WCAN_PAGE=0

7	6	5	4	3	2	1	0
0				**CCE**	**SELWK_EN**	**0**	**WCAN_EN**
r				rw	rwh	r	rw

Field	Bits	Type	Description
WCAN_EN	0	rw	**WCAN Enable** This bit enables or disables the clock to WCAN Selective Wake-Up filter block. 0_B Disable Clock to WCAN Selective Wake-Up filter block 1_B Enable Clock to WCAN Selective Wake-Up filter block
SELWK_EN	2	rwh	**Selective Wake-Up Enable** The Selective Wake-up Enable function enables the protocol handler. The status of the protocol handler is also acknowledged through INTESTAT0.SWACK. The status of the Selective Wakeup Active is given in INTESTAT1.SWKSET. It also serves several functions when enabled. 1. Allows Wake-up source WUF to generate WUF_INT (see Figure "Wake-up Frame Detection"). 2. Allows error counter to generate ERROR_INT (see Figure "Error Counter"), indicated by INTESTAT0.ECOFL. 3. Triggers ERROR_INT and wake-up if there are any changes to the message object registers, indicated by INTESTAT0.MORC. 4. If a SYSERR occurs while SELWK_EN is enabled, then the Selective Wake-up function shall be disabled (ie., this bit field will be reset to 0). It is also indicated by INTESTAT0.ERSEL. 0_B Selective wake disabled 1_B Selective wake enabled
CCE	3	rw	**Configuration Change Enable** This bit field enables the configuration of the Bit Timing Registers. It also acts as soft-reset to the Selective Wake-up filter block. It has to be set back to '0' once the bit timing registers are modified for desired baudrate. 0_B The Bit Timing Registers (BTLn_CTRL), CDR Configuration registers (CDR_CTRL, CDR_UPPER_CTRL, CDR_LOWER_CTRL) and CAN_FD_CTRL may only be read. All attempts to modify them will be ignored. 1_B The Bit Timing Registers and CDR Configuration registers may be read and written
0	1, 7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 738 Reset Values of WCAN_CFG

Reset Type	Reset Value	Note
LVD Reset	XXXX 00X0$_B$	
Generated Reset	---- 00-0$_B$	

Wake-Up CAN Interrupt Mask Register

WCAN_INTMRSLT
Wake-Up CAN Interrupt Mask Register (0B1$_H$) Reset Value: Table 739
RMAP: 0, PAGE: WCAN_PAGE=0

7	6	5	4	3	2	1	0
0				WUPMASK	WUFMASK	ERRMASK	CANTOMASK
r				rw	rw	rw	rw

Field	Bits	Type	Description
CANTOMASK	0	rw	**CAN Time-Out Masking** This bit controls the interrupt masking for CANTO_INT interrupt. 0$_B$ CAN Time-Out is masked - no interrupt is triggered 1$_B$ CAN Time-Out is signaled as an interrupt
ERRMASK	1	rw	**SYSERR Masking** This bit controls the interrupt masking for SYSERR interrupt. 0$_B$ SYSERR is masked - no interrupt is triggered 1$_B$ SYSERR is signaled as an interrupt
WUFMASK	2	rw	**Wake-Up Frame Interrupt Masking** This bit controls the interrupt masking for WUF_INT interrupt. 0$_B$ WUF_INT is masked - no interrupt is triggered 1$_B$ WUF_INT is signaled as an interrupt
WUPMASK	3	rw	**Wake-Up Pattern Detected Interrupt Masking** This bit controls the interrupt masking for WUP_INT interrupt. 0$_B$ WUP_INT is masked - no interrupt is triggered 1$_B$ WUP_INT is signaled as an interrupt
0	7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 739 Reset Values of WCAN_INTMRSLT

Reset Type	Reset Value	Note
LVD Reset	X0$_H$	
Generated Reset	-0$_H$	

Wake-Up CAN Interrupt and Event Status Register 0

WCAN_INTESTAT0
Wake-Up CAN Interrupt and Event Status Register 0($0B0_H$) **Reset Value: Table 740**
RMAP: 0, PAGE: WCAN_PAGE=1

7	6	5	4	3	2	1	0
WUP	**ERSEL**	**MORC**	**ECOFL**	**CANTO**	**SYSERR**	**MODE**	**SWACK**
rh	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
SWACK	0	rh	**Selective Wake-Up Enable Acknowledge** This bit field indicates the acknowledgement of protocol handler enabled/disabled. 0_B Protocol Handler is disabled and configuration of WUF is possible 1_B Protocol Handler is enabled and configuration of WUF should not be done
MODE	1	rh	**WCAN Mode of Operation** This bitfield indicates the WCAN mode of operation. 0_B Normal Mode 1_B Standby Mode
SYSERR	2	rh	**CAN Wake-Up System Error** The SYSERR indicates a configuration error (MORC), and an error counter overflow condition (ECOFL). 0_B Selective Wake Mode is possible 1_B System Error detected, WCAN enabling not possible. It also leads to SYSERR_INT interrupt trigger if enabled through ERRMASK.
CANTO	3	rh	**CAN Time-Out Detection** This bit indicates that there was an inactive CAN bus on RXDCO pin within tSILENCE of 750 ms. See section "CAN Message Timeout". A software write to INTESTATCLR0.CANTOCLR is required to reset this bit. 0_B Normal operation 1_B CAN Time-Out detected. It also leads to CANTO_INT interrupt if enabled through CANTOMASK.
ECOFL	4	rh	**Error Counter Overflow** This bit field shows occurrence of Error Counter overflow. A set condition of this flag by hardware has higher priority than a clear through ECOFLCLR. 0_B Error Counter has not overflowed (<= 31) 1_B Error Counter overflowed (> 31)
MORC	5	rh	**Message Object Register Changed** A set condition of this flag by hardware has higher priority than a clear through MORCCLR. 0_B Message object registers not modified after SELWK_EN is set 1_B Message object registers was modified after SELWK_EN is set

Field	Bits	Type	Description
ERSEL	6	rh	**Error Select** A set condition of this flag by hardware has higher priority than a clear through ERSELCLR. 0_B SYSERR is not set while SELWK_EN is enabled 1_B SYSERR is set while SELWK_EN is enabled
WUP	7	rh	**Wake-Up Pattern Detected** This bit field is set when a falling edge on RXD (or WUP) is detected, when WCAN is in Standby mode. A software write to INTESTATCLR0.WUPCLR is required to clear this bit field. 0_B WUP not detected 1_B WUP detected. It also triggers WUP_INT interrupt when enabled through INTMRSLT.WUPMASK

Table 740 Reset Values of WCAN_INTESTAT0

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Wake-Up CAN Interrupt and Event Status Register 1

WCAN_INTESTAT1
Wake-Up CAN Interrupt and Event Status Register 1($0B1_H$) **Reset Value: Table 741**
RMAP: 0, PAGE: WCAN_PAGE=1

7	6	5	4	3	2	1	0
0	**RXDS**	**RXDF**	**FDF**	**CANSIL**	**SWKSET**	**SYNC**	**WUF**
r	rh	rh	rh	rh	rh	rh	rh

Field	Bits	Type	Description
WUF	0	rh	**Wake-Up Frame Detection (acc. ISO 11898-6)** This bit shows the status whether a Wake-up Frame has been detected. 0_B No Wake-Up Frame detected 1_B Wake-Up Frame detected. It also leads to WUF_INT interrupt trigger if enabled through WUFMASK.
SYNC	1	rh	**Synchronisation (at least one CAN frame without fail must have been received)** This bit is set when a valid Classical CAN frame is detected and is updatedafter every received Classical CAN frame. This flag is not updated when aCAN FD frame is received. Clearing this flag through SYNCCLR has higherpriority than a set condition by hardware. 0_B WCAN function not working or not synchronous to CAN bus 1_B Valid CAN frame received, WCAN function is synchronous to CAN bus

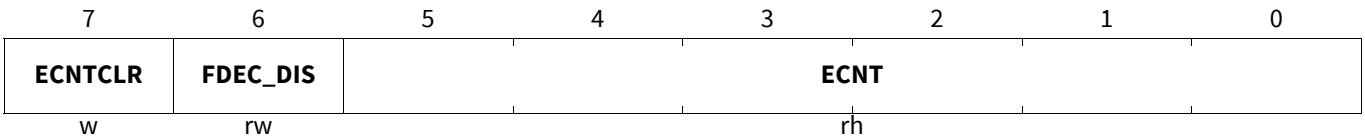
Field	Bits	Type	Description
SWKSET	2	rh	**Selective Wake Activity** 0_B Selective Wake is not active 1_B Selective Wake is activated
CANSIL	3	rh	**CAN Bus Silent Time during Selective Wake Operation** This bit is set if there is no activity on RXDC0 pin on the CAN bus after tSILENCE of 750 ms expired. If any activity is detected on the bus, this bit is reset automatically. tSILENCE is fixed at 750 ms. This bit can also be cleared through bit WCAN_INTESCLR.CANSILCLR. A set condition of this bit has higher priority than a clear through CANSILCLR. 0_B tSILENCE not exceeded 1_B tSILENCE is exceeded
FDF	4	rh	**FD Frame Reception** This bit indicates reception of CAN FD frame. After reception of the CAN FD frame is completed, this bit is cleared to '0'. 0_B No action 1_B FD Frame is being received
RXDF	5	rh	**Filtered Receive Data Input** This bit shows the logic state of the filtered received data input signal (RXD_filt). 0_B Dominant 1_B Recessive
RXDS	6	rh	**Sampled Receive Data Input** This bit shows the logic state of the sampled received data input signal by the protocol handler. 0_B Dominant 1_B Recessive
0	7	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 741 Reset Values of WCAN_INTESTAT1

Reset Type	Reset Value	Note
LVD Reset	X110 0000_B	
Generated Reset	–110 0000_B	

Wake-Up CAN Error Counter Register

WCAN_FRMERRCNT
Wake-Up CAN Error Counter Register **($0B2_H$)** **Reset Value: Table 742**
RMAP: 0, PAGE: WCAN_PAGE=1



Field	Bits	Type	Description
ECNT	5:0	rh	**SWK CAN Frame Error Counter** If WCAN_FD_CTRL.FDEN = 0, the CAN Frame Error Counter is incremented when an error message or CAN FD frame is received, and decremented on a reception of a correct classical CAN frame, if ECNT>0. If FDEN = 1, the behaviour of the error counter is based on FDEC_DIS bit configuration. See section "Error Counter".
FDEC_DIS	6	rw	**Error Count Disable on CAN FD Frame** This bit is used to describe the behaviour of the Error Counter for FD frames. The value of this bit has no effect if WCAN_FD_CTRL.FDEN = 0. *Note: When an FD frame is detected, the error is checked only until the FDF bit of the FD frame.* 0_B Error counter is incremented or decremented based on correctness of the bitfields in arbitration phase of the FD frame 1_B Error counter is active only for Classical CAN frames, and it is frozen for CAN FD frames
ECNTCLR	7	w	**Error Counter Clear Bit** Writing a 1 to this bit clears the error counter FRMERRCNT.ECNT.

Table 742 Reset Values of WCAN_FRMERRCNT

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Wake-Up CAN Interrupt and Event Status Clear Register 0

WCAN_INTESCLR0
Wake-Up CAN Interrupt and Event Status Clear Register 0($0B3_H$) **Reset Value: Table 743**
RMAP: 0, PAGE: WCAN_PAGE=1

7	6	5	4	3	2	1	0
WUPCLR	**ERSELCLR**	**MORCCLR**	**ECOFLCLR**	**CANTOCLR**	**SYSERRCLR**	**0**	
w	w	w	w	w	w	r	

Field	Bits	Type	Description
SYSERRCLR	2	w	**CAN WUP System Error Clear Bit** Writing a 1 to this bit clears the corresponding status bit INTESTAT0.SYSERR.
CANTOCLR	3	w	**CAN Time-Out Detection Clear Bit** Writing a 1 to this bit clears the corresponding status bit INTESTAT0.CANTO.

Field	Bits	Type	Description
ECOFLCLR	4	w	**Error Counter Overflow Clear Bit** Writing a 1 to this bit clears the corresponding status bit INTESTAT0.ECOFL.
MORCCLR	5	w	**Message Object Register Changed Clear Bit** Writing a 1 to this bit clears the corresponding status bit INTESTAT0.MORC.
ERSELCLR	6	w	**Error Select Clear Bit** Writing a 1 to this bit clears the corresponding status bit INTESTAT0.ERSEL.
WUPCLR	7	w	**WUP Clear Bit** Writing a 1 to this bit clears the corresponding status bit INTESTAT0.WUP.
0	1:0	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 743 Reset Values of WCAN_INTESCLR0

Reset Type	Reset Value	Note
LVD Reset	$0000\ 00XX_B$	
Generated Reset	$0000\ 00--_B$	

Wake-Up CAN Interrupt and Event Status Clear Register 1

WCAN_INTESCLR1
Wake-Up CAN Interrupt and Event Status Clear Register 1($0B4_H$) **Reset Value: Table 744**
RMAP: 0, PAGE: WCAN_PAGE=1

7	6	5	4	3	2	1	0
0				**CANSILCLR**	**0**	**SYNCCLR**	**WUFCLR**
r				w	r	w	w

Field	Bits	Type	Description
WUFCLR	0	w	**Wake-up Frame Detection Flag Clear Bit** Writing a 1 to this bit clears the corresponding status bit INTESTAT1.WUF.
SYNCCLR	1	w	**Synchronisation (at least one can frame without fail must have been received) Clear Bit** Writing a 1 to this bit clears the corresponding status bit INTESTAT1.SYNC.
CANSILCLR	3	w	**CAN Silent Time during SWK Operation Clear Bit** Writing a 1 to this bit clears the corresponding status bit INTESTAT1.CANSIL.
0	2, 7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 744 Reset Values of WCAN_INTESCLR1

Reset Type	Reset Value	Note
LVD Reset	XXXX 0X00$_B$	
Generated Reset	---- 0–00$_B$	

49.17.2.5 WCAN Initialization Sequence

Inorder to initialize the WCAN module, the following sequence has to be performed.

- Set **WCAN_CFG**.WCAN_EN = 1b
- Set **WCAN_CFG**.CCE = 1b
- Configure CDR,CAN FD and Baud Rate Configuration registers
- Reset **WCAN_CFG**.CCE = 0b
- Reset **WCAN_CFG**.SELWK_EN = 0b
- Configure WUF Configuration registers
- Set **WCAN_CFG**.SELWK_EN = 1b
- Wait until **WCAN_INTESTAT0**.SWACK = 1b

In order to perform only reconfiguration of the WUF Configuration registers, the following sequence has to be performed

- Set **WCAN_CFG**.WCAN_EN = 1b
- Reset **WCAN_CFG**.SELWK_EN = 0b
- Wait until **WCAN_INTESTAT0**.SWACK =0b
- Configure WUF Configuration registers
- Set **WCAN_CFG**.SELWK_EN = 1b
- Wait until **WCAN_INTESTAT0**.SWACK = 1b

After the initialization/WUF reconfiguration sequence is performed, the application software has to check the **WCAN_INTESTAT1**.SWKSET bit field to verify if the Selective Wake-Up feature is successfully enabled or not. The SWKSET bit field is set to '1' if all of the below mentioned conditions are true, and it is cleared if any of the conditions is not met

- **WCAN_CFG**.WCAN_EN = 1b
- **WCAN_CFG**.CCE = 0b
- **WCAN_INTESTAT0**.SWACK = 1b
- **WCAN_INTESTAT0**.MODE = 0b
- f_{BACK} is switched on

*Note: If application software writes either **WCAN_CFG**.CCE = 1 or **WCAN_CFG**.SELWK_EN = 0b when WCAN is already active, then any ongoing reception will be lost. If **WCAN_CFG**.WCAN_EN = 0 when WCAN is already active, then the selective wake-Up filter block will be in halt state and the application software has to perform re-initialization based on the sequence given above.*

49.17.2.6 Wake-Up Frame Configuration and Detection

The received frame is compared to the configured wake-up frame and incase a match is detected a wake-up request is issued. The wake-up request sets the **WCAN_INTESTAT1**.WUF flag and issues WUF_INT interrupt if not masked by **WCAN_INTMRSLT**.WUFMASK. Only a matched Classical CAN frame can lead to WUF detection.

*Note: In addition to the message object configuration registers (i.e **WCAN_DLC_CTRL**, IDn_CTRL(n=0-3), MASK_IDn_CTRL(n=0-3), **WCAN_DATAn_CTRL (n=0-7)**) the node bit timing configuration is configured in the BTLn_CTRL(n=1-2) register.*

49.17.2.6.1 Configuration of Wake-Up Frame Filter

The message object configuration of the wake-up frame is done in the following registers

- Register IDn_CTRL (n=0-3) contains the CAN identifier filter bits.
- Register MASK_IDn_CTRL (n=0-3) contains the ID mask bits.
- Register **WCAN_DLC_CTRL** are the data length code filter bits.
- Register **WCAN_DATAn_CTRL (n=0-7)** are the data filter bits.

49.17.2.6.2 Detection of Wake-Up Frame Filter

The compare mechanisms are different according to the information field inside the CAN frame.

- The identifier programmed inside IDn_CTRL (n=0-3) and MASK_IDn_CTRL (n=0-3) has to match the data in the received frame and the logic seen in **Figure 913**, a wake-up condition is detected.
- The DLC programmed in **WCAN_DLC_CTRL** together with the data filter bits in **WCAN_DATAn_CTRL (n=0-7)** will be compared to the DLC and data inside the received frame, in case a single '1' match in **WCAN_DATAn_CTRL (n=0-7)**, a wake-up condition within the data is detected. DLC values greater than 8 will be treated as 8. See **Figure 913**.

If and only if, identifier and masking, the **WCAN_DLC_CTRL** and the **WCAN_DATAn_CTRL (n=0-7)** will lead to a match, a valid wake-up Frame (WUF) is detected and a wake-up interrupt will be issued if **WCAN_CFG**.SELWK_EN is enabled.

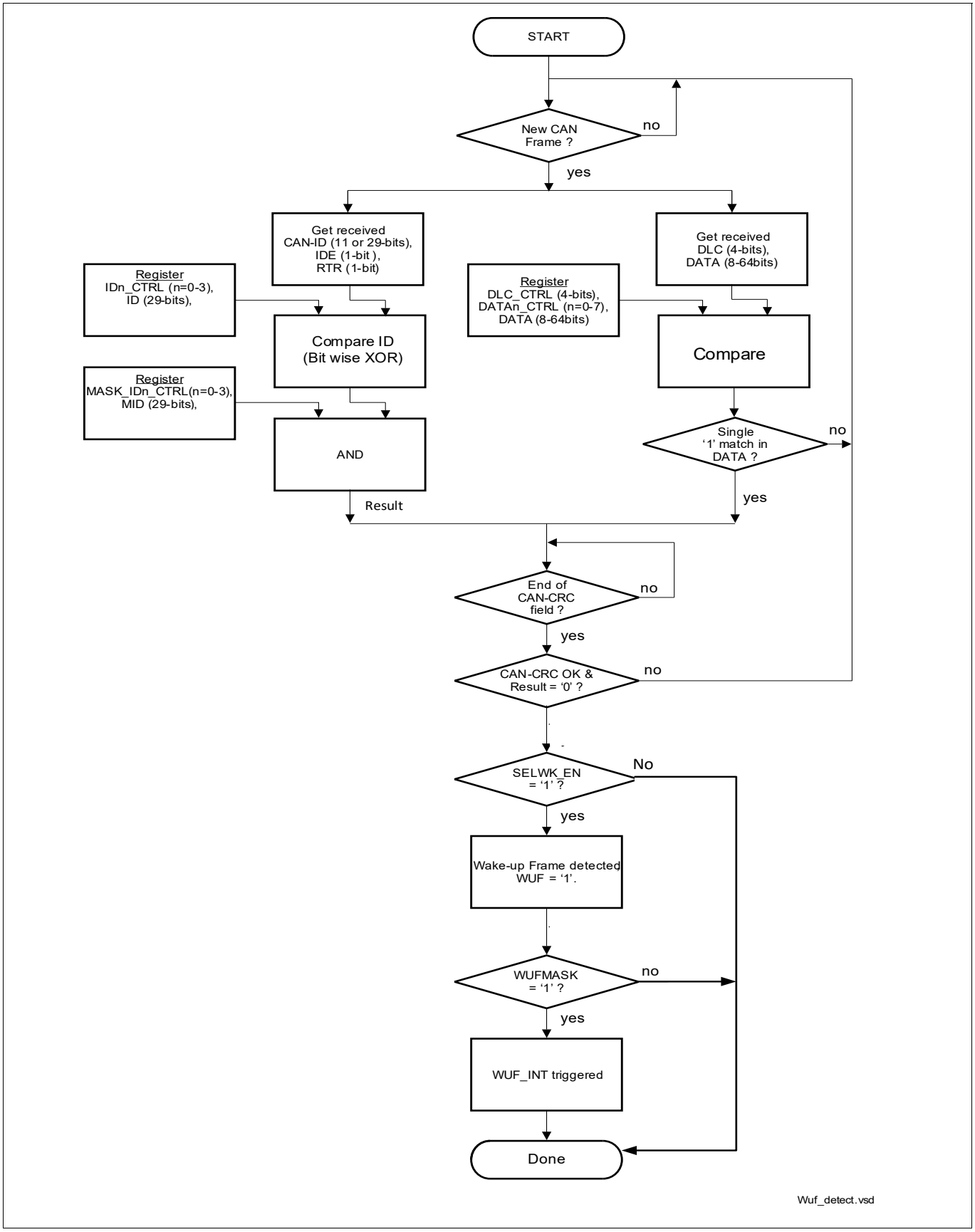


Figure 913 Wake-Up Frame Detection

49.17.2.7 CAN Message Timeout

A timeout mechanism to monitor CAN bus activity is available on the WCAN. The timeout mechanism monitors the RXDC0 pin. When the RXDC0 pin is inactive for a tSILENCE time of 750 ms, both CAN bus silent bit (**WCAN_INTESTAT1**.CANSIL) and if selective Wake-Up is enabled, CAN timeout bit (**WCAN_INTESTAT0**.CANTO) will be set with a CAN time out interrupt (CANTO_INT). An occurrence of CAN Timeout will clear the error counter.

The CAN bus silent bit (**WCAN_INTESTAT1**.CANSIL) is reset when there is activity on the CAN bus where else the CAN timeout bit can be reset by setting CAN time out clear bit (**WCAN_INTESCLR0**.CANTOCLR). When WCAN_EN = 0, the timeout counter remains in the reset state. The **WCAN_INTESTAT1**.CANSIL can also be cleared by setting **WCAN_INTESCLR1**.CANSILCLR.

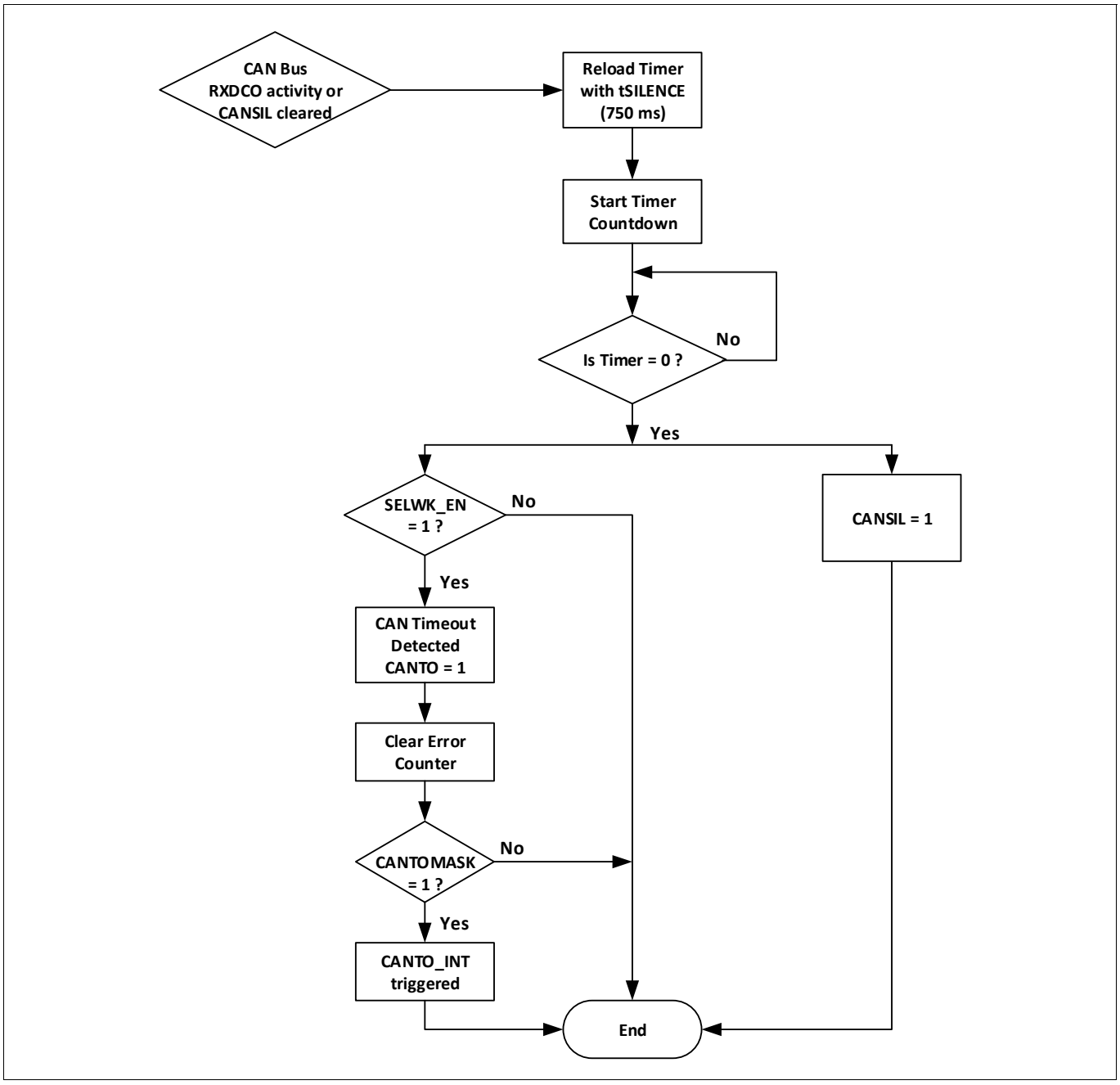


Figure 914 CAN Timeout

49.17.2.8 Error Counter

The error counting mechanism in ISO 11898-6 is different to ISO 11898-1. As the device is only receiving messages and never transmitting it could never get into a bus-off situation. Therefore an ISO11898-6 implementation is incrementing its error counter on a Form, Stuff and a CRC error. In case of a correct message, the error counter will be decremented, if it is greater than 0. In case the error counter is greater than 31 an overflow occurs and a wake-up will be initiated. This shall enable the software to correct the situation. In case of overflow the **WCAN_INTESTAT0**.SYSERR and **WCAN_INTESTAT0**.ECOFL flags are set. The error counter is cleared when timeout has occurred or when **WCAN_CFG**.SELWK_EN is re-enabled (from '0' to '1'). If the CAN FD tolerance feature is enabled and a CAN FD frame is received, the error monitoring is done only until FDF bit field. The behavior of the error counter for CAN FD frames is based on **WCAN_FRMERRCNT**.FDEC_DIS bit field. The conditions at which error counter is incremented or decremented is given in the table below. The **WCAN_INTESTAT0**.ECOFL flags can be cleared by writing **WCAN_INTESCLR0**.ECOFLCLR to '1'.

Note: If ECNTCLR is written to '1' and at the same time an error in received frame has occurred, the error counter will remain cleared and it will not be incremented.

Table 745 Error Counter Conditions

WCAN_FD_CTRL.FDEN	**WCAN_FRMERRCNT.FDEC_DIS**	**Behavior of Error Counter**
0	X	Decrement on: • Valid Classical CAN frame Increment on: • Corrupt Classical CAN frame • CAN FD Frame
1	0	Decrement on: • Valid Classical or CAN FD frame Increment on: • Corrupt CAN FD Frame • Corrupt Classical CAN frame
1	1	Decrement on: • Valid Classical CAN frame Increment on: • Corrupt Classical CAN frame No Action: • CAN FD Frame

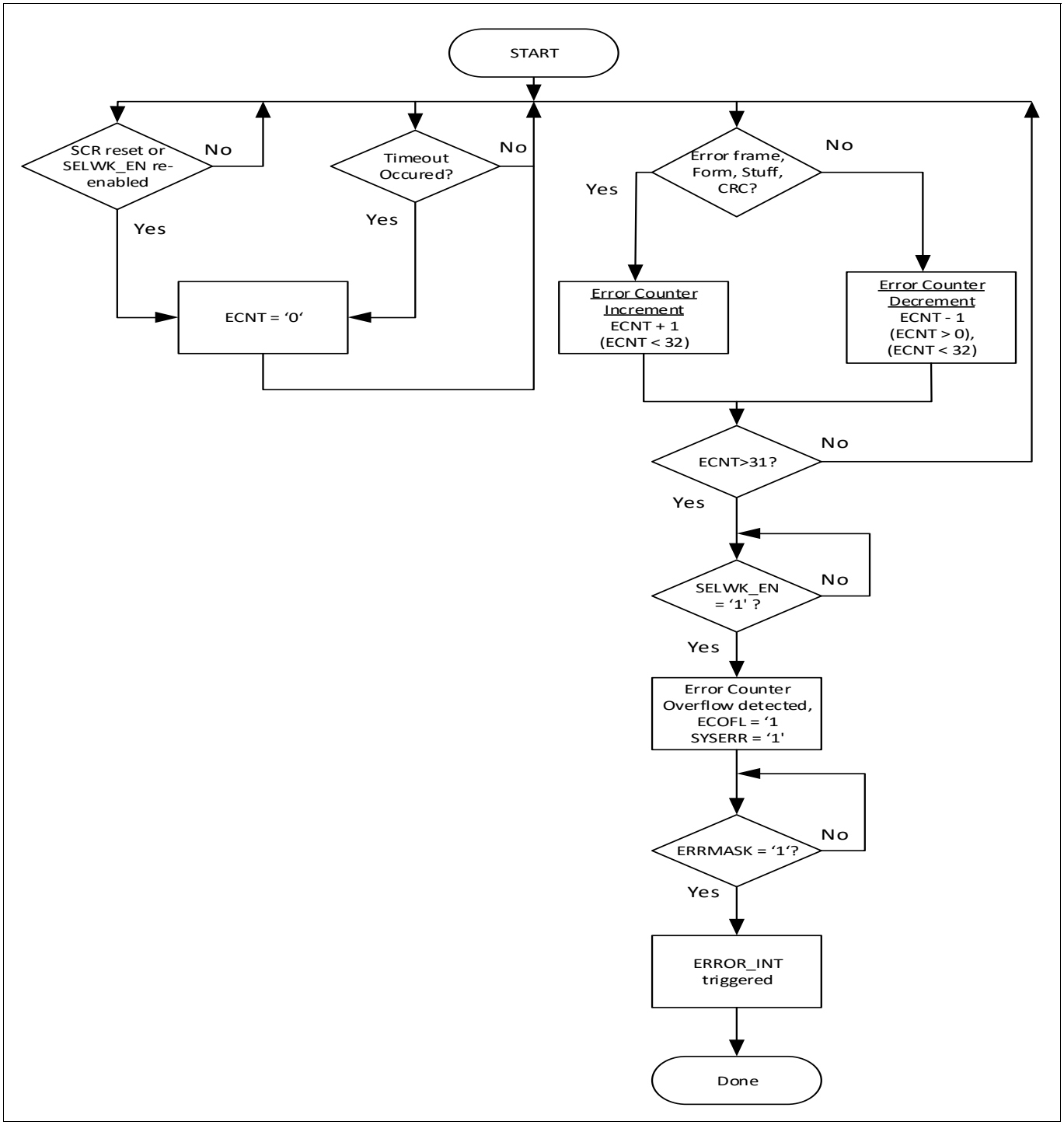


Figure 915 Error Counter

49.17.3 WUP Detection

WCAN supports WUP detection by receiving a falling edge of RXD when a Wake-Up Pattern is detected by the CAN Transceiver. In case the CAN Transceiver doesn't support Wake-Up Pattern detection, then WCAN considers any falling edge on RXD as WUP. In WCAN, the WUP detection is active only when the Selective Wake-Up feature is enabled and CAN timeout has occurred. On occurrence of CAN Timeout, WCAN enters into a Standby mode. When WCAN is in Standby mode,

- the protocol Handler is reset and Selective Wake-Up feature is disabled;
- the Clock to the Protocol Handler is switched off to save power;
- WUP detection is active - waiting for a falling edge on RXD (decoding of the WUP itself should be done by the CAN Transceiver);
- **WCAN_INTESTAT0**.MODE = 1 and **WCAN_INTESTAT1**.SWKSET = 0.

Note: In WCAN Standby mode, the WUP detection is active even when 100 MHz f_{BACK} is switched-off. In such a case, the WUP detection will be performed with 70 kHz clock. In-order to save power when WCAN in standby mode, the application software may disable 100 MHz f_{BACK}, and only 70 kHz clock can be provided. The 100 MHz f_{BACK} is automatically turned-on when WUP is detected.

When WUP is detected in Standby Mode, the WCAN enters into the Normal Mode where,

- selective Wake-Up feature is re-enabled;
- the clock to the protocol handler is active;
- Error Counter is reset to '0';
- 100 MHz f_{BACK} enable is requested (CMCON.OSCPD = '0' when CMCON.OSCWAKE = '1'), when 100 MHz f_{BACK} is disabled in Standby Mode;
- **WCAN_INTESTAT0**.MODE = 0, **WCAN_INTESTAT1**.SWKSET = 1 and **WCAN_INTESTAT0**.WUP = 1;
- WUP_INT is triggered when it is not masked through **WCAN_INTMRSLT**.WUPMASK.

The application software can make WCAN to leave the Standby Mode by SCR Reset or by disabling WCAN (WCAN_EN = 0). The figure below shows the state diagram on transition of WCAN between different modes of operation.

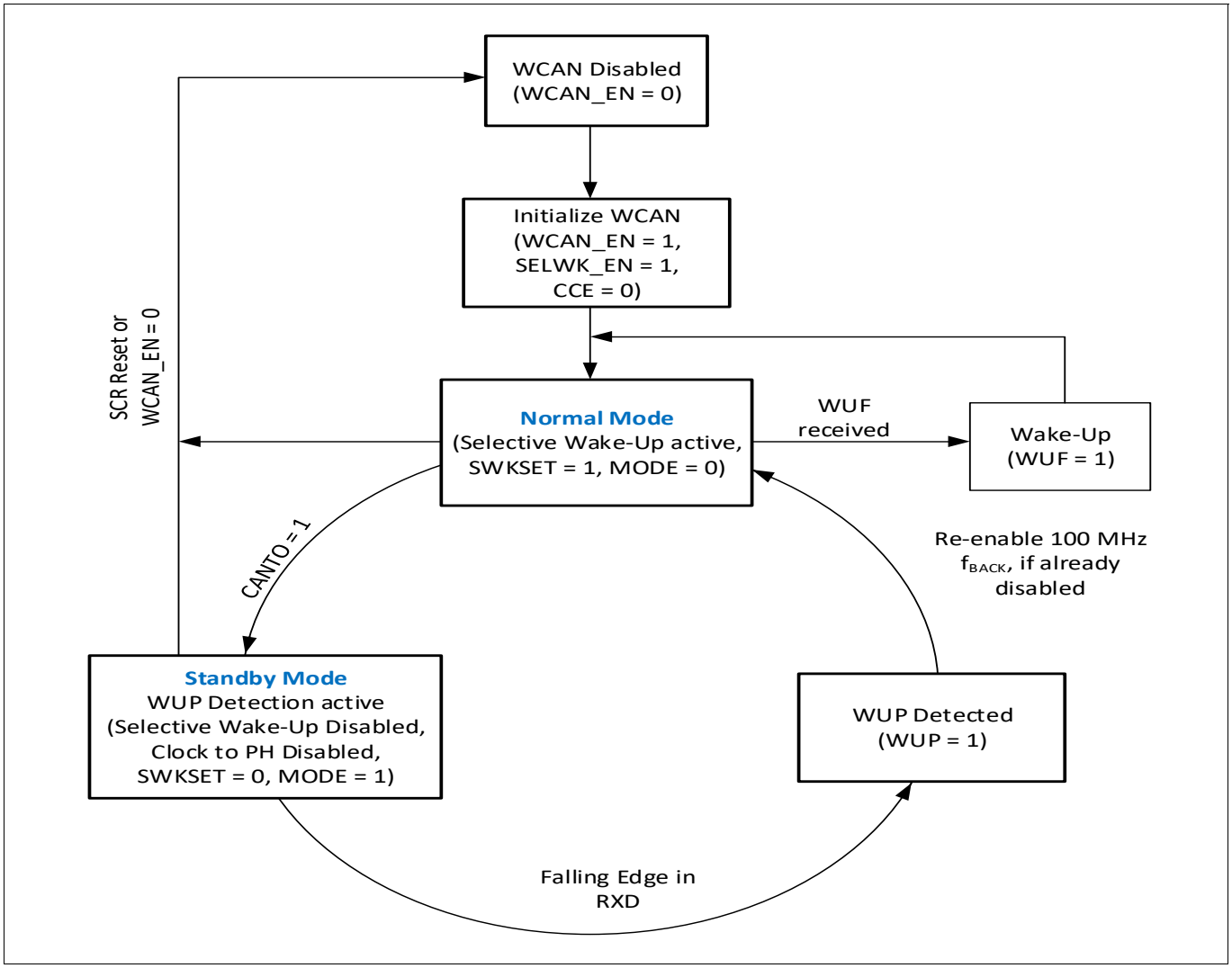


Figure 916 WCAN Modes of Operation

49.17.4 CAN Functional Description

49.17.4.1 CAN Node Control

The CAN node is equipped with an individual set of SFR registers to control and to monitor the CAN node.

49.17.4.1.1 Baud Rate Prescaler

The WCAN has an integer divider with a scaling factor defined by the BRP bit field. The f_{BACK} is the input frequency and the fCAN is the scaled output frequency of the integer divider. The BRP bit field is recommended to be configured based on the baud rates required as shown in **Table 747**, because BRP values influences the receive input filter time and CAN FD filter time. The fCAN frequency is fed to the Receive Input Filter and CDR (to measure the sample point). When 100 MHz f_{BACK} is switched off, the Selective Wake-Up filter block will be reset and then f_{CAN} is also switched off. Hence, the Selective wake-up feature cannot be used ie., **WCAN_INTESTAT1**.SWKSET = 0.

49.17.4.1.2 Receive Input Filter

The RXD signal coming from the transceiver, is filtered to eliminate the jitter effects from the CAN bus, The filter time is dependent on the recommended baud rate configuration and the BRP values (**Table 747**). The filtering is done based on fCAN frequency. The filtering time T_filt is based on the END COUNT value for different BRP as shown in **Table 747**. The filter has an asymmetrical behavior as shown in **Figure 917**.

The first recessive to dominant edge is detected asynchronously, and the event is used for the starting of the filter counter. Once the counter reaches the END COUNT value, the RXD is checked again if it is still '0', then the synchronization edge is considered valid. Hence the filter introduces a delay of T_filt.

At dominant to recessive edge, RXD is checked at each fCAN clock cycle. The RXD_FILT output is '1' only if the input signal is a stable recessive for entire filtering time (T_filt). The recessive bits shorter than t_glitch are filtered out and only recessive bits longer that t-detect are considered valid.

When CAN FD tolerance is enable, the received input signal RXD is used directly without filtering for the data phase, to avoid the additional filter delay.

The logic level of the filtered input RXD_FILT is shown in **WCAN_INTESTAT1**.RXDF.

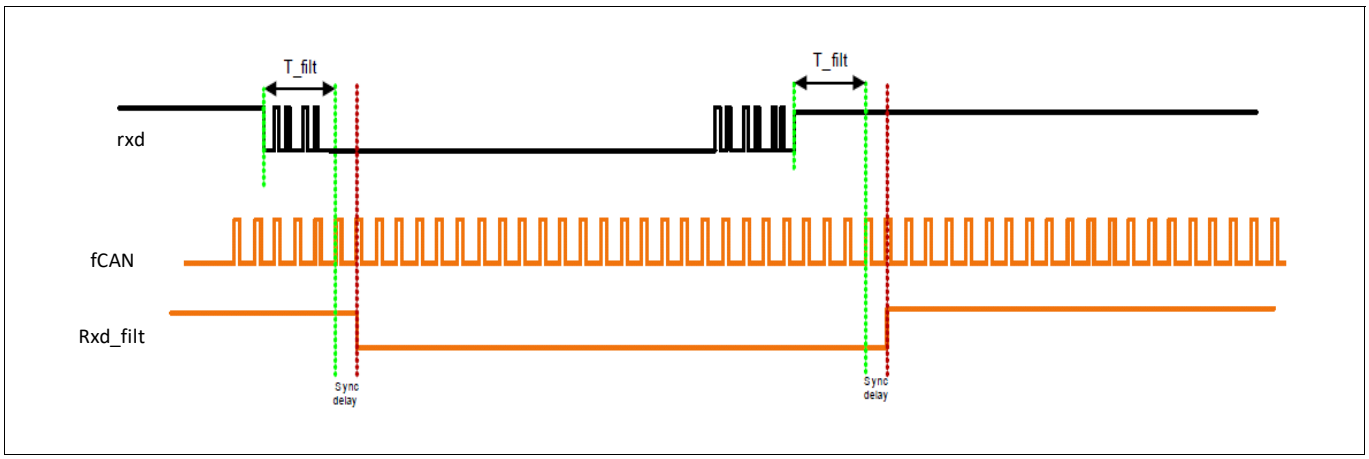
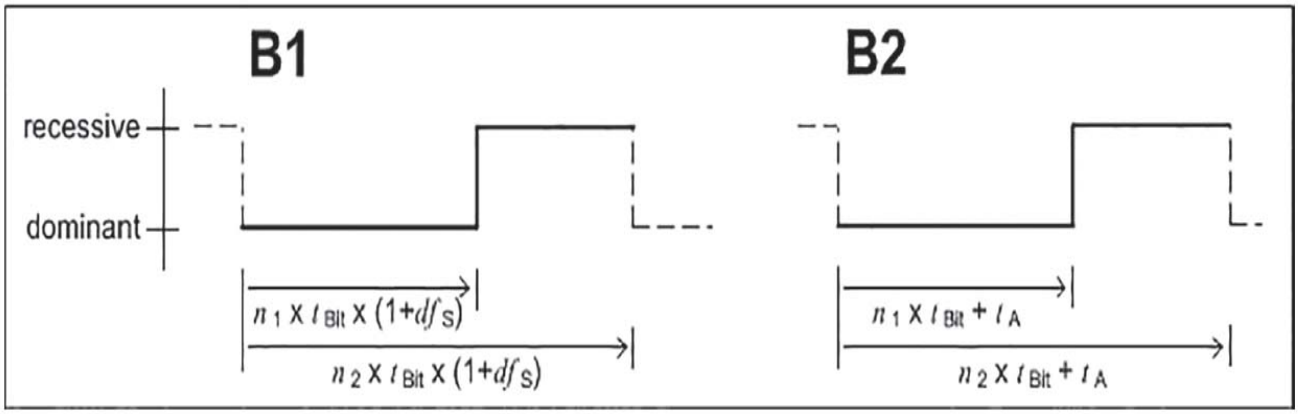


Figure 917 RXD Filtering

49.17.4.1.3 Bit Timing

According to ISO 11898-2 standard, a received classical CAN frame can be decoded correctly when the timing of the CAN bit fields complies with multiple instances of signal shape B1 and one instance of signal shape B2 (to

handle the sender clock tolerance and loss of arbitration). In order to compensate possible deviations of the CAN oscillator frequency, integrated clock and data recovery is used (CDR). The CDR can be optionally enabled or disabled using CDR_EN bit field. It is recommended to always enable the CDR feature. If CDR is disabled, the required oscillator tolerance is 1% or else it is 4%. When CDR is disabled, the sample point position is calculated directly based on BTLn_CTRL (n=1,2) values. Before enabling the CDR, the bit timings parameters have to be properly configured as shown in **Table 747**. When CAN FD tolerance is enabled **WCAN_FD_CTRL**.FDEN = 1, the CDR is stopped during CAN FD frame reception, in order to avoid calculation based on high baud-rate data phase. The CDR is kept at last calculated value before CAN FD has been detected.



Key

n_1 number of consecutive dominant bits {1, 2, 3, 4, 5}

n_2 number of bits between two falling edges {2, 3, ..., 10}; n2 > n1

t_A $0 \le t_A \le 55$ % of t_{Bit} (product specific higher maximum values for t_A are allowed)

t_B $0 \le t_B \le 5$ % of t_{Bit} (product specific higher maximum values for t_B are allowed)

t_{Bit} nominal bit time (Note: often used values for t_{Bit} are 2 µs, 4 µs and 8 µs)

df_S transceivers according to this document shall tolerate sender clock frequency deviations up to at least 0,5 %

Figure 918 CAN Bit Timing Signal Shape

Clock and Data Recovery (CDR)

The CDR consists of three building blocks as shown in **Figure 919**. The CDR uses the fCAN clock. This clock period defines the duration of one time quanta tq. For ISO 11898-6 some settings are guaranteed to be working according to the given propagation delay times (see **Section 49.17.4.1.4**). Working settings for 125k, 250k and 500k will be provided. The acquisition counter of CDR measures the CAN bit timings in terms of number of tq. The sample point calculation evaluates the new sample point based on the computed CAN bit length and the sample point position configured in **WCAN_BTL2_CTRL**. The filter block is used to compensate the error introduced during internal measurement of number of tq in one bit time.

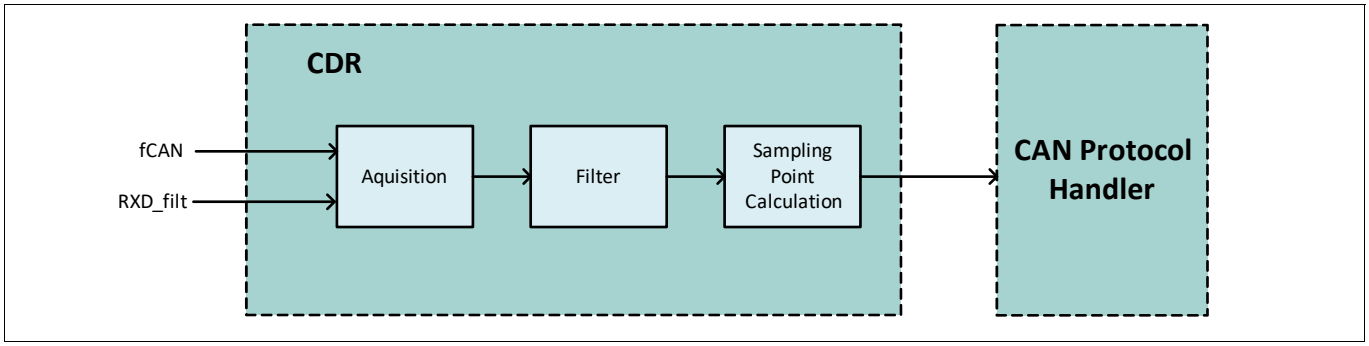


Figure 919 CDR Block Diagram

The configuration change enable, **WCAN_CFG**.CCE has to be enabled before a valid CAN bit timing can be written to the CAN Node Bit Timing Register, BTLn_CTRL (n=1-2). This is then followed by resetting **WCAN_CFG**.CCE bit before enabling the operation of the CAN node.

Note: The proposed sample point is at 80% of the bit time.

Acquisition Counter

The acquisition counter measures the bit timing based on the number of tq between two synchronization events (i.e., recessive to dominant transition of the filtered RX signal). The counter is active only if CDR_EN=1 and enabled by the protocol handler. The protocol handler enables the acquisition outside the arbitration phase, where CAN bit shortenings and extension are likely to occur. The only exception to that scenario is when an error is detected in the arbitration phase. In that case, the acquisition is enabled for preventing deadlock situation when the oscillator is outside the range and error occurs during Arbitration. The synchronization edges occurring within an error frame are cancelled by the windowing of acquisition. Based on the counter value, the number of CAN bits received between two synchronization events can be determined. This value is between 2 and 10, in accordance to the bit-stuffing rule.

Filter and Sample Point Calculation

The sample point calculation block evaluates the sampling position of the CAN bus bit based on the information from the Acquisition Counter and from the CDR configuration registers. Depending on the selected frequency and the baud rate, the default number of actual clock pulses expected during a CAN bus bit time is configured in **WCAN_BTL1_CTRL** register. The sampling position of the CAN bit is defined in BTL2_CTRL.SP bit field. It can be calculated as follows

BTL2_CTRL.SP = (64*sampling point)/100

*Note: For Example: A sampling point at 80% of bit time corresponds to **WCAN_BTL2_CTRL**.SP = 33_H.*

The number of tqs in one CAN bit time (tbit_calc) is measured internally based on the output from the acquisition counter and the expected number of tqs in one CAN bit time as provided by **WCAN_BTL1_CTRL**. This measured value can be read from the **WCAN_CDR_MEAS_HIGH** and **WCAN_CDR_MEAS_LOW**. The error introduced during the measurement is compensated using the Error scaling factor. During successive CDR measurements, the new bit timing value is evaluated by an averaging algorithm:

tbit_new = tbit_calc + error / SEL_FILT

where SEL_FILT is the error scaling factor (ie., the value over how many acquisition cycles the average will be built) as described in Table below.

Table 746 Number of acquisitions based on ESF

ESF	SEL_FILT
00_B	8
01_B	16 (recommended)
10_B	32
11_B	Adaptive averaging (based on number of bits between two consecutive synchronization edges) 2, 3 or 4 CAN bits = 32 5 or 6 CAN bits = 16 7, 8 or 9 CAN bits = 8

The sample point is then calculated based on **WCAN_BTL2_CTRL** and tbit_new. The sampling point is calculated only when tbit_new falls within the range defined by **WCAN_CDR_LOWER_CTRL**.LOWER and **WCAN_CDR_UPPER_CTRL**.UPPER. If not, the sample point is calculated directly based on the tbit value defined in **WCAN_BTL1_CTRL**.

Table 747 Bit Timing and RXD Filter configuration - Recommended

Baud Rate (kbits/s)	BRP	BTL1_CTRL	RXD Filter END COUNT	t_glitch (ns)	t-detect (ns)
1000	00_B	64_H	24	200	400
500	01_B	64_H	16	250	500
250	10_B	64_H	8	250	500
125	11_B	64_H	8	400	1400

49.17.4.1.4 Network Propagation Delays

The “worst case” network propagation delay is the network propagation delay after the 5th bit for the dominant to recessive edge on the bus. (See **Figure 920**)

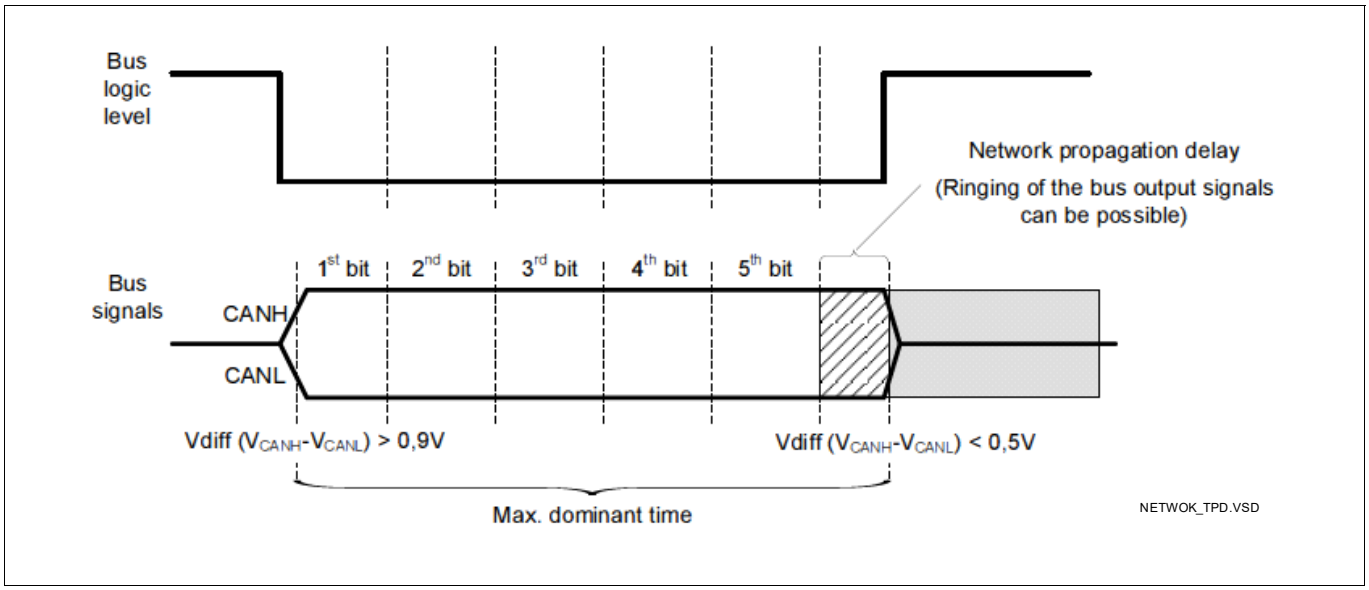


Figure 920 Network Propagation Delay after 5th Bit

The WCANmodule will be capable of sampling the recessive stuff bit after 5 consecutive dominant bits to ensure a safe wake-up, even if the dominant to recessive edge is delayed for the value of the network propagation delay shown in **Table 748**. The network propagation delay describes the maximal occurrence propagation delay which can be seen on the bus. This maximum network propagation delay does not include any EMC jitter. (i.e no HF injection)

Table 748 Network propagation delay

Baud Rate (Baud)	Network Propagation Delay (ns)	Maximum Dominant time (ns) (sender oscillator tolerance included)
1 MBaud	550	5550
500 kBaud	1350	11350
250 kBaud	2675	22675
125 kBaud	5450	45450

For HF injection the WCAN module is able to handle EMC network propagation delay as shown in **Table 749**.

Table 749 EMC Network propagation delay

Baud Rate (Baud)	Network Propagation Delay (ns)	Maximum Dominant time (ns) (sender oscillator tolerance included)
1 MBaud	450	5450
500 kBaud	1250	11250
250 kBaud	2575	22575
125 kBaud	5350	45350

Also the WCAN module is able to decode frames that contain sequences of 5 consecutive dominant bits followed by 5 consecutive recessive bits as shown in **Figure 920**.

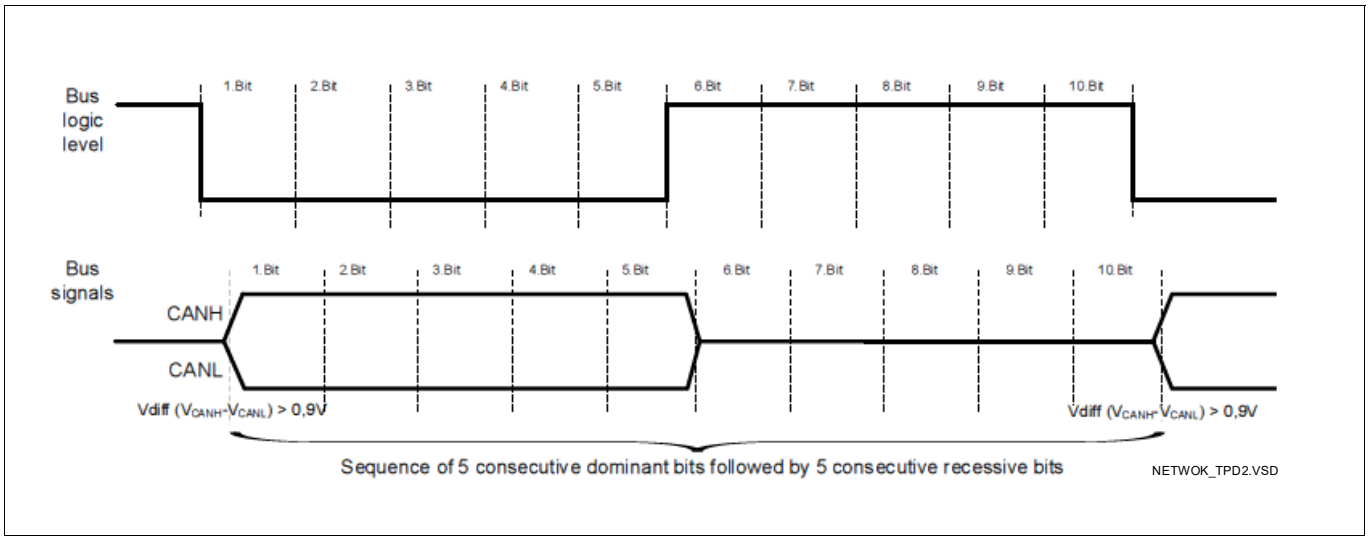


Figure 921 Network Propagation Delay after 10th Bit

The transmitting CAN node on the bus has to guarantee a maximum baud rate tolerance of +/- 0.4% for correct detection of the CAN frames (See **Table 750**)

Table 750 Worst Case length of 10 bits

Baud Rate (Baud)	Worst Case time for length of 10 bits by sender oscillator tolerance (-0.4%)	Worst Case time for length of 10 bits by sender oscillator tolerance (+0.4%)
1 MBaud	9960 ns	10040 ns
500 kBaud	19920 ns	20080 ns
250 kBaud	39840 ns	40160 ns
125 kBaud	79680 ns	80320 ns

49.17.4.1.5 CAN FD Tolerance

The WCAN can be configured as CAN FD tolerant (FDEN = 1), which means that protocol handler doesn't consider received CAN FD frames as error condition. When FDF bit is found dominant, the decoding phase is stopped and the protocol handler goes to the bus integration state waiting for 5 consecutive recessive bits. The protocol handler has an End Of Frame (EOF) bit counter to detect 5 consecutive recessive bits. The FD block generates a reset signal (active low) for the EOF bit counter, when a dominant bit in the high bit-rate phase is detected. The FD block directly uses the RXD signal and operates at f_{BACK} to avoid losing a dominant bit during the high bit-rate phase. Consequently a filter to suppress the dominant glitches is introduced in FD block. Since only dominant detection acts as the reset for EOF bit counter, the filter is an up-counter incremented when the RXD signal is dominant and reset when the RXD is recessive. So the filtered RXD is considered dominant when the counter end value (FD End Count) is reached. The glitch length (t_FD_glitch) to filter out and the minimum dominant pulse to detect (t_FD_detect) are based on the Baud rate and the FD factor as shown in **Table 751** below.

Table 751 CAN FD dominant bit filter time

Arbitration Baud Rate (kBaud)	FD Factor	FD_FILT	FD End Count	t_FD_glitch (ns)	t_FD_detect (ns)
125	4	011	90	400	1400
250	4	010	48	200	700
500	4	001	25	100	350
125	10	010	48	200	700
250	10	001	25	100	350
500	10	000	13	50	175

49.17.4.2 WCAN Kernel Registers

49.17.4.2.1 Register Address Map

The register set of the WCAN module consists of the below distinct subsets:

1. The Configuration, Miscellaneous Registers exist in one instance and apply for the entire WCAN module.
2. The Wake-Up CAN registers consist of the following:
   a) CAN Node Registers (**WCAN_BTL1_CTRL**, **WCAN_BTL2_CTRL**, **WCAN_CDR_CTRL**, **WCAN_CDR_UPPER_CTRL**, **WCAN_CDR_LOWER_CTRL**, **WCAN_CDR_MEAS_HIGH**, **WCAN_CDR_MEAS_LOW** and **WCAN_FD_CTRL**), which applies to a single CAN node.
   b) WUF configuration registers (DLC_CTRL, IDx_CTRL, MASK_IDx_CTRL, DATAx_CTRL) are defined for a single message object, and thus exist once for each message object.

Configuration and Status Registers

The configuration and status registers of a WCAN node are accessed from standard (non-mapped) SFR area. **Table 752** shows the address as seen on the Standby Controller (SCR).

Table 752 WCAN Configuration and Status Registers Address Map

Register	Addr.	Page	SCR Reset Value	Full Name of Register
WCAN_CFG	$B0_H$	0	00_H	Configuration Register
WCAN_INTMRSLT	$B1_H$	0	00_H	Interrupt Mask and Reference Select Register
WCAN_INTESTAT0	$B0_H$	1	00_H	Interrupt and Event Status 0 Register
WCAN_INTESTAT1	$B1_H$	1	60_H	Interrupt and Event Status 1 Register
WCAN_FRMERRCNT	$B2_H$	1	00_H	Frame Error Counter Register
WCAN_INTESCLR0	$B3_H$	1	00_H	Interrupt and Event Status Clear Register 0
WCAN_INTESCLR1	$B4_H$	1	00_H	Interrupt and Event Status Clear Register 1

CAN Node Registers

The registers of a WCAN node are accessed from standard (non-mapped) SFR area. **Table 753** shows the address of registers as seen on the Standby Controller (SCR).

Table 753 CAN Node Registers Address Map

Register	Addr.	Page	SCR Reset Value	Full Name of Register
WCAN_BTL1_CTRL	$B5_H$	1	64_H	Bit Timing Logic 1 Control Register
WCAN_BTL2_CTRL	$B6_H$	1	73_H	Bit Timing Logic 2 Control Register
WCAN_CDR_CTRL	$B2_H$	0	05_H	CDR Control Register
WCAN_CDR_UPPER_CTRL	$B3_H$	0	69_H	CDR Upper Limit Control Register
WCAN_CDR_LOWER_CTRL	$B4_H$	0	$5F_H$	CDR Lower Limit Control Register
WCAN_CDR_MEAS_HIGH	$B5_H$	0	64_H	CDR Measured High Register

Table 753 CAN Node Registers Address Map (cont'd)

Register	Addr.	Page	SCR Reset Value	Full Name of Register
WCAN_CDR_MEAS_LOW	$B6_H$	0	00_H	CDR Measured Low Register
WCAN_FD_CTRL	$B7_H$	0	00_H	CAN FD Control Register

WUF Configuration Registers

The registers to configure WUF are accessed from standard (non-mapped) SFR area. **Table 754** shows the address of the WUF configuration Registers as seen on the Standby Controller (SCR).

Table 754 WUF Configuration Registers Address Map

Register	Addr.	Page	SCR Reset Value	Full Name of Register
WCAN_DLC_CTRL	$B7_H$	1	00_H	Message Data Length Code Control Register
WCAN_ID0_CTRL	$B0_H$	2	00_H	Message Identifier Control Register 0
WCAN_ID1_CTRL	$B1_H$	2	00_H	Message Identifier Control Register 1
WCAN_ID2_CTRL	$B2_H$	2	00_H	Message Identifier Control Register 2
WCAN_ID3_CTRL	$B3_H$	2	00_H	Message Identifier Control Register 3
WCAN_MASK_ID0_CTRL	$B4_H$	2	00_H	Message Identifier Acceptance Mask Register 0
WCAN_MASK_ID1_CTRL	$B5_H$	2	00_H	Message Identifier Acceptance Mask Register 1
WCAN_MASK_ID2_CTRL	$B6_H$	2	00_H	Message Identifier Acceptance Mask Register 2
WCAN_MASK_ID3_CTRL	$B7_H$	2	00_H	Message Identifier Acceptance Mask Register 3
DATA0_CTRL	$B0_H$	3	00_H	Message Data Control Register 0
DATA1_CTRL	$B1_H$	3	00_H	Message Data Control Register 1
DATA2_CTRL	$B2_H$	3	00_H	Message Data Control Register 2
DATA3_CTRL	$B3_H$	3	00_H	Message Data Control Register 3
DATA4_CTRL	$B4_H$	3	00_H	Message Data Control Register 4
DATA5_CTRL	$B5_H$	3	00_H	Message Data Control Register 5
DATA6_CTRL	$B6_H$	3	00_H	Message Data Control Register 6
DATA7_CTRL	$B7_H$	3	00_H	Message Data Control Register 7

49.17.4.2.2 CAN Node Registers

The CAN node specific registers contain information that is directly related to the operation of the CAN node.The Node Control Register contains basic settings that define the bit timing for the operation of the CAN node.

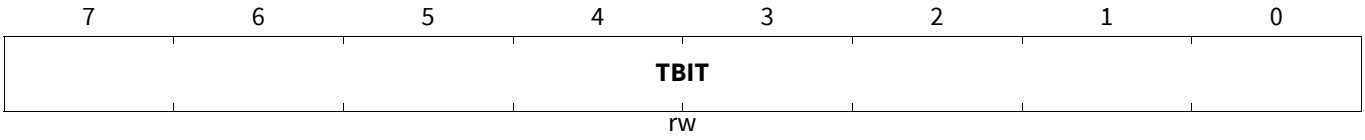
Bit Timing Logic Control Register

The Bit Timing Logic Control Registers contain all parameters to setup the bit timing for the CAN transfer.

*Note: The startup-software (SSW) overwrites the BTL1_CTRL register with 64_H and BTL2_CTRL register with 73_H after reset, to obtain a 500 Kbits/s baud rate. Please use the values specified in **Table 747** for 1 Mbits/s (BTL1_CTRL1, BTL2_CTRL), 250 Kbits/s (BTL1_CTRL, BTL2_CTRL) and 125 Kbits/s (BTL1_CTRL, BTL2_CTRL) baud rate.*
*The Configuration Change Enable, **WCAN_CFG**.CCE has to be enabled before the BTLn_CTRL register can be written.*

Bit Timing Logic 1 Control Register

WCAN_BTL1_CTRL
Bit Timing Logic 1 Control Register **($0B5_H$)** **Reset Value: Table 756**
RMAP: 0, PAGE: WCAN_PAGE=1



Field	Bits	Type	Description
TBIT	7:0	rw	**Number of Time Quanta in a Bit Time** This bitfield represents the number of time quanta in a bit time. The time quanta is based on fCAN frequency.

Table 755 Access Mode Restrictions of WCAN_BTL1_CTRL sorted by descending priority

Mode Name	Access Mode		Description
WCAN_CFG.CCE = 1	rw	TBIT	
(default)	r	TBIT	

Table 756 Reset Values of WCAN_BTL1_CTRL

Reset Type	Reset Value	Note
LVD Reset	64_H	
Generated Reset	64_H	

Bit Timing Logic 2 Control Register

WCAN_BTL2_CTRL
Bit Timing Logic 2 Control Register **(0B6$_H$)** **Reset Value: Table 758**
RMAP: 0, PAGE: WCAN_PAGE=1

7	6	5	4	3	2	1	0
BRP		**SP**					
rw		rw					

Field	Bits	Type	Description
SP	5:0	rw	**Sample Point Position** Represents the sample point position. The sample point position is calculated as given in section "Filter and Sample Point Calculation". 33_H represents the sample point position of 79.68 % (~80 %). The value shall be within $2C_H$ to 38_H (ie., 67.5 % to 87.5 %).
BRP	7:6	rw	**Baudrate Prescaler** This bitfield defines the scaling factor to derive one time-quanta. 00_B **DIV1**, Division by 1 (100 MHz) 01_B **DIV2**, Division by 2 (50 MHz) 10_B **DIV4**, Division by 4 (25 MHz) 11_B **DIV8**, Division by 8 (12.5 MHz)

Table 757 Access Mode Restrictions of WCAN_BTL2_CTRL sorted by descending priority

Mode Name	Access Mode		Description
WCAN_CFG.CCE = 1	rw	BRP, SP	
(default)	r	BRP, SP	

Table 758 Reset Values of WCAN_BTL2_CTRL

Reset Type	Reset Value	Note
LVD Reset	73_H	
Generated Reset	73_H	

CDR Control Register

WCAN_CDR_CTRL
CDR Control Register **(0B2$_H$)** **Reset Value: Table 760**
RMAP: 0, PAGE: WCAN_PAGE=0

7	6	5	4	3	2	1	0
0				**ESF**		**0**	**CDREN**
r				rw		r	rw

Field	Bits	Type	Description
CDREN	0	rw	**Enable CDR** 0_B Disable CDR internal bit time measurement 1_B Enable CDR internal bit time measurement
ESF	3:2	rw	**Error scaling factor** This bitfield defines the number of acquisition cycles on which average on error is calculated. 00_B **ERR_SCALE_FACT8**, 8 01_B **ERR_SCALE_FACT16**, 16 (recommended) 10_B **ERR_SCALE_FACT32**, 32 11_B **ADAPT_AVRGE**, Adaptive averaging
0	1, 7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 759 Access Mode Restrictions of WCAN_CDR_CTRL sorted by descending priority

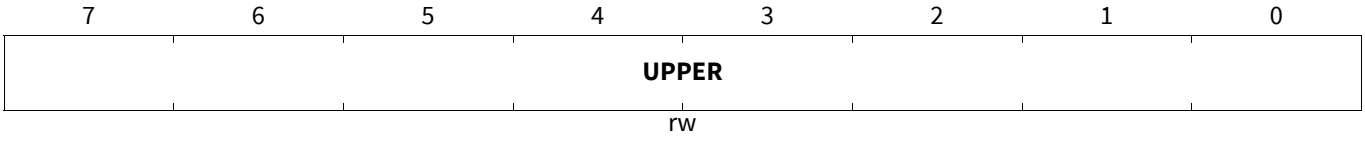
Mode Name	Access Mode		Description
WCAN_CFG.CCE = 1	rw	CDREN, ESF	
(default)	r	CDREN, ESF	

Table 760 Reset Values of WCAN_CDR_CTRL

Reset Type	Reset Value	Note
LVD Reset	XXXX $01X1_B$	
Generated Reset	---- $01-1_B$	

CDR Upper Limit Control Register

WCAN_CDR_UPPER_CTRL
CDR Upper Limit Control Register **($0B3_H$)** **Reset Value: Table 762**
RMAP: 0, PAGE: WCAN_PAGE=0



Field	Bits	Type	Description
UPPER	7:0	rw	**CDR Upper Limit** This bitfield defines the upper limit for CDR internal bit time measurement. It should be configured as +5% of BTL1_CTRL value (integer part).

Table 761 Access Mode Restrictions of WCAN_CDR_UPPER_CTRL sorted by descending priority

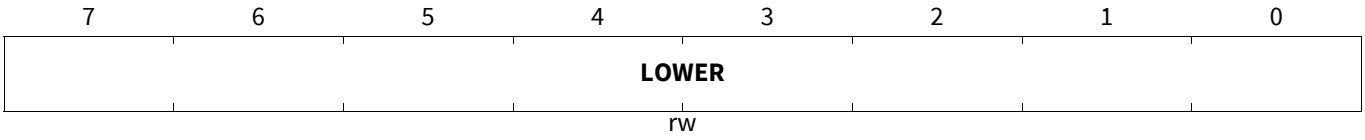
Mode Name	Access Mode		Description
WCAN_CFG.CCE = 1	rw	UPPER	
(default)	r	UPPER	

Table 762 Reset Values of WCAN_CDR_UPPER_CTRL

Reset Type	Reset Value	Note
LVD Reset	69_H	
Generated Reset	69_H	

CDR Lower Limit Control Register

WCAN_CDR_LOWER_CTRL
CDR Lower Limit Control Register ($0B4_H$) Reset Value: Table 764
RMAP: 0, PAGE: WCAN_PAGE=0



Field	Bits	Type	Description
LOWER	7:0	rw	**CDR Lower Limit** This bitfield defines the lower limit for CDR internal bit time measurement. It should be configured as -5% of BTL1_CTRL value (integer part).

Table 763 Access Mode Restrictions of WCAN_CDR_LOWER_CTRL sorted by descending priority

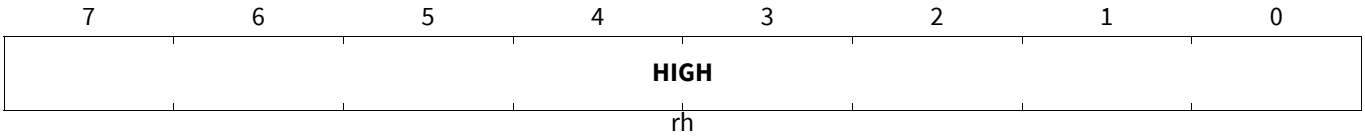
Mode Name	Access Mode		Description
WCAN_CFG.CCE = 1	rw	LOWER	
(default)	r	LOWER	

Table 764 Reset Values of WCAN_CDR_LOWER_CTRL

Reset Type	Reset Value	Note
LVD Reset	$5F_H$	
Generated Reset	$5F_H$	

CDR Measured High Register

WCAN_CDR_MEAS_HIGH
CDR Measured High Register **($0B5_H$)** **Reset Value: Table 765**
RMAP: 0, PAGE: WCAN_PAGE=0



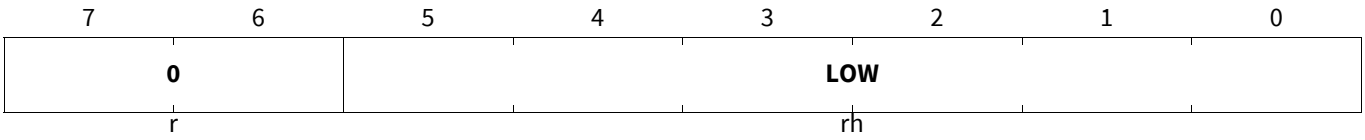
Field	Bits	Type	Description
HIGH	7:0	rh	**CDR Internally Measured High Value** This bitfield shows the internally measured duration of a CAN bus bit as number of fCAN clock cycles (integer part).

Table 765 Reset Values of WCAN_CDR_MEAS_HIGH

Reset Type	Reset Value	Note
LVD Reset	64_H	
Generated Reset	64_H	

CDR Measured Low Register

WCAN_CDR_MEAS_LOW
CDR Measured Low Register **($0B6_H$)** **Reset Value: Table 766**
RMAP: 0, PAGE: WCAN_PAGE=0



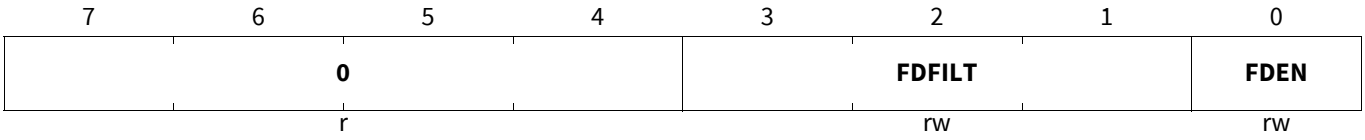
Field	Bits	Type	Description
LOW	5:0	rh	**CDR Internally Measured Low Value** This bitfield shows the internally measured duration of a CAN bus bit as number of fCAN clock cycles (fractional part).
0	7:6	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 766 Reset Values of WCAN_CDR_MEAS_LOW

Reset Type	Reset Value	Note
LVD Reset	$XX00\ 0000_B$	
Generated Reset	$--00\ 0000_B$	

CAN FD Control Register

WCAN_FD_CTRL
CAN FD Control Register **($0B7_H$)** **Reset Value: Table 768**
RMAP: 0, PAGE: WCAN_PAGE=0



Field	Bits	Type	Description
FDEN	0	rw	**CAN FD Tolerant Enable** This bitfield enables or disables the CAN FD tolerant feature. 0_B Disabled 1_B Enabled
FDFILT	3:1	rw	**CAN FD Filter Time** This bitfield configures the filter time to detect the dominant bit field in FD phase. Refer to Table "CAN FD dominant bit filter time" for the FD filter time configuration for various baudrates. 000_B 13 (130 ns) 001_B 25 (250 ns) 010_B 48 (480 ns) 011_B 90 (900 ns) **others**, Reserved
0	7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 767 Access Mode Restrictions of WCAN_FD_CTRL sorted by descending priority

Mode Name	Access Mode		Description
WCAN_CFG.CCE = 1	rw	FDEN, FDFILT	
(default)	r	FDEN, FDFILT	

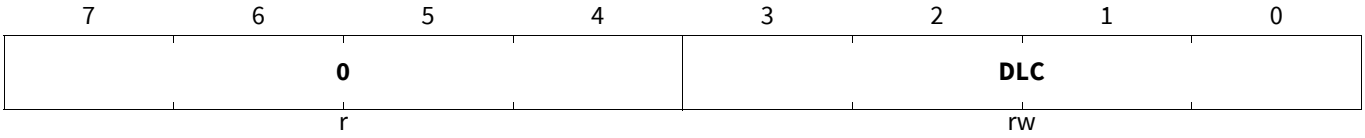
Table 768 Reset Values of WCAN_FD_CTRL

Reset Type	Reset Value	Note
LVD Reset	$X0_H$	
Generated Reset	-0_H	

49.17.4.2.3 Message Object Registers

Message Data Length Code Control Register

WCAN_DLC_CTRL
Message Data Length Code Control Register (0B7$_H$) **Reset Value: Table 769**
RMAP: 0, PAGE: WCAN_PAGE=1



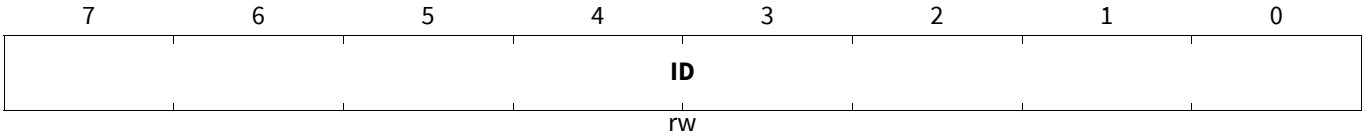
Field	Bits	Type	Description
DLC	3:0	rw	**Data Length Code** Valid values for the data length are 0 to 8. DLC > 8 will be considered as 8 data bytes, but the truncation will not be visible in this bitfield upon read.
0	7:4	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 769 Reset Values of WCAN_DLC_CTRL

Reset Type	Reset Value	Note
LVD Reset	X0$_H$	
Generated Reset	–0$_H$	

Message Identifier Control Register 0

WCAN_ID0_CTRL
Message Identifier Control Register 0 (0B0$_H$) **Reset Value: Table 770**
RMAP: 0, PAGE: WCAN_PAGE=2



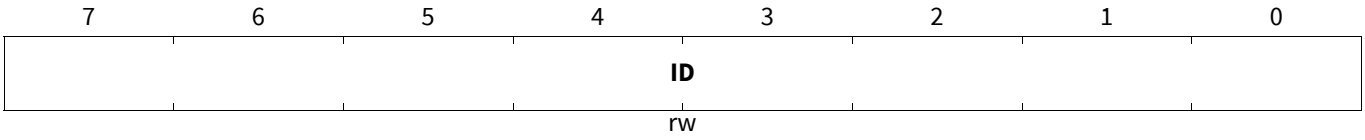
Field	Bits	Type	Description
ID	7:0	rw	**CAN Identifier** Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers, bits ID[17:0] are "don't care". This bitfield specifies ID7 to ID0.

Table 770 Reset Values of WCAN_ID0_CTRL

Reset Type	Reset Value	Note
LVD Reset	00$_H$	
Generated Reset	00$_H$	

Message Identifier Control Register 1

WCAN_ID1_CTRL
Message Identifier Control Register 1 **($0B1_H$)** **Reset Value: Table 771**
RMAP: 0, PAGE: WCAN_PAGE=2



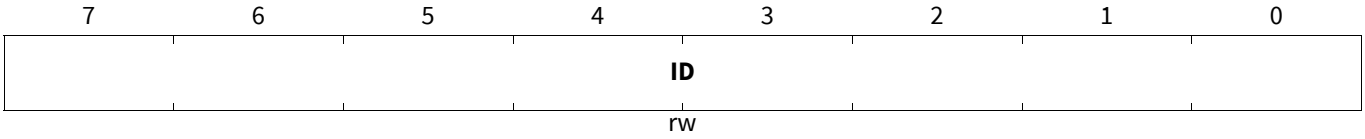
Field	Bits	Type	Description
ID	7:0	rw	**CAN Identifier** Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers, bits ID[17:0] are "don't care". This bitfield specifies ID15 to ID8.

Table 771 Reset Values of WCAN_ID1_CTRL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Message Identifier Control Register 2

WCAN_ID2_CTRL
Message Identifier Control Register 2 **($0B2_H$)** **Reset Value: Table 772**
RMAP: 0, PAGE: WCAN_PAGE=2



Field	Bits	Type	Description
ID	7:0	rw	**CAN Identifier** Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers, bits ID[17:0] are "don't care". This bitfield specifies ID23 to ID16.

Table 772 Reset Values of WCAN_ID2_CTRL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Message Identifier Control Register 3

WCAN_ID3_CTRL
Message Identifier Control Register 3 **($0B3_H$)** **Reset Value: Table 773**
RMAP: 0, PAGE: WCAN_PAGE=2

7	6	5	4	3	2	1	0
0	**ID**					**0**	**IDE**
r	rw					r	rw

Field	Bits	Type	Description
IDE	0	rw	**Identified Extension Bit** 0_B Standard frame with 11 bits identified 1_B Extended frame with 29 bits identified
ID	6:2	rw	**CAN Identifier** Identifier of a standard message (ID[28:18]) or an extended message (ID[28:0]). For standard identifiers, bits ID[17:0] are "don't care". This bitfield specifies ID28 to ID24.
0	1, 7	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 773 Reset Values of WCAN_ID3_CTRL

Reset Type	Reset Value	Note
LVD Reset	X000 00X0_B	
Generated Reset	–000 00–0_B	

Message Identifier Mask Register 0

WCAN_MASK_ID0_CTRL
Message Identifier Mask Register 0 **($0B4_H$)** **Reset Value: Table 774**
RMAP: 0, PAGE: WCAN_PAGE=2

7	6	5	4	3	2	1	0
MID							
rw							

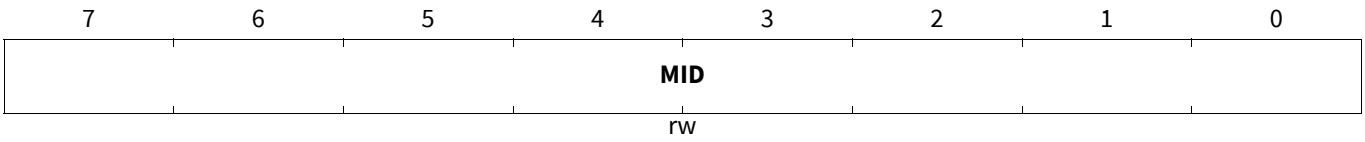
Field	Bits	Type	Description
MID	7:0	rw	**Mask for Message Identifier** Mask to filter incoming messages with standard identifiers (MID[28:18]) or extended identifiers (MID[28:0]). For standard identifiers, bits MID[17:0] are "don't care". If a specific bit in MID is set, the corresponding bit in the ID is considered for the ID matching evaluation, otherwise it is considered as "don't care". This bitfield specifies MID7 to MID0.

Table 774 Reset Values of WCAN_MASK_ID0_CTRL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Message Identifier Mask Register 1

WCAN_MASK_ID1_CTRL
Message Identifier Mask Register 1 ($0B5_H$) Reset Value: Table 775
RMAP: 0, PAGE: WCAN_PAGE=2



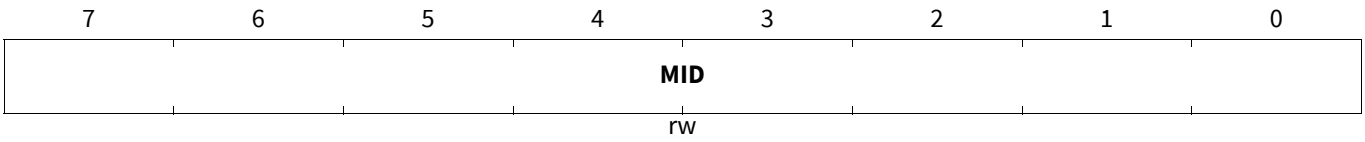
Field	Bits	Type	Description
MID	7:0	rw	**Mask for Message Identifier** Mask to filter incoming messages with standard identifiers (MID[28:18]) or extended identifiers (MID[28:0]). For standard identifiers, bits MID[17:0] are "don't care". If a specific bit in MID is set, the corresponding bit in the ID is considered for the ID matching evaluation, otherwise it is considered as "don't care". This bitfield specifies MID15 to MID8.

Table 775 Reset Values of WCAN_MASK_ID1_CTRL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Message Identifier Mask Register 2

WCAN_MASK_ID2_CTRL
Message Identifier Mask Register 2 ($0B6_H$) Reset Value: Table 776
RMAP: 0, PAGE: WCAN_PAGE=2



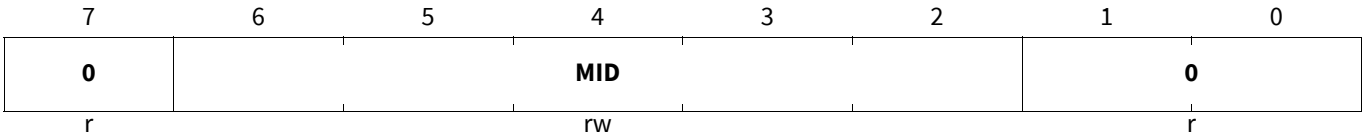
Field	Bits	Type	Description
MID	7:0	rw	**Mask for Message Identifier** Mask to filter incoming messages with standard identifiers (MID[28:18]) or extended identifiers (MID[28:0]). For standard identifiers, bits MID[17:0] are "don't care". If a specific bit in MID is set, the corresponding bit in the ID is considered for the ID matching evaluation, otherwise it is considered as "don't care". This bitfield specifies MID23 to MID16.

Table 776 Reset Values of WCAN_MASK_ID2_CTRL

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

Message Identifier Mask Register 3

WCAN_MASK_ID3_CTRL
Message Identifier Mask Register 3 ($0B7_H$) Reset Value: Table 777
RMAP: 0, PAGE: WCAN_PAGE=2



Field	Bits	Type	Description
MID	6:2	rw	**Mask for Message Identifier** Mask to filter incoming messages with standard identifiers (MID[28:18]) or extended identifiers (MID[28:0]). For standard identifiers, bits MID[17:0] are "don't care". If a specific bit in MID is set, the corresponding bit in the ID is considered for the ID matching evaluation, otherwise it is considered as "don't care". This bitfield specifies MID28 to MID24.
0	1:0, 7	r	**Reserved** Returns 0 when read; shall be written with 0.

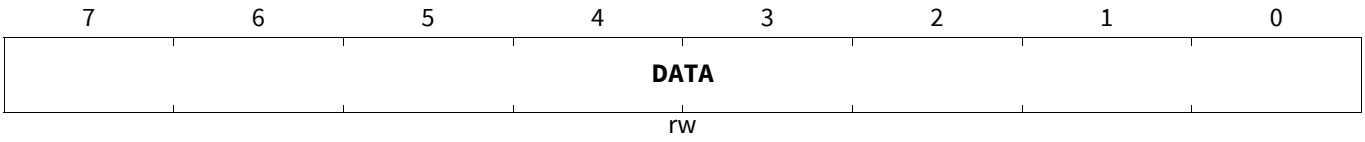
Table 777 Reset Values of WCAN_MASK_ID3_CTRL

Reset Type	Reset Value	Note
LVD Reset	$X000\ 00XX_B$	
Generated Reset	$-000\ 00--_B$	

DATA Control Register n

Data Control Registers, DATAn_CTRL (n = 0-7) contain the data filter bits. See figure “Wake-up Frame Detection” for flow of acceptance filtering which would determine if a Wake-up Frame has been detected. It is a match if a single bit is equally set at the same bit position in the incoming data and DATAn_CTRL (n = 0-7) register.

WCAN_DATAn_CTRL (n=0-7)
DATA Control Register n ($0B0_H$+n) Reset Value: Table 778
RMAP: 0, PAGE: WCAN_PAGE=3



Field	Bits	Type	Description
DATA	7:0	rw	**CAN Data Byte** These bitfields specify the data bits 7 to 0 that will be compared with corresponding bits of $(n+1)^{th}$ data byte of the received CAN frame.

Table 778 Reset Values of WCAN_DATAn_CTRL (n=0-7)

Reset Type	Reset Value	Note
LVD Reset	00_H	
Generated Reset	00_H	

49.17.4.3 Miscellaneous Register

49.17.4.3.1 Paging Register

Page Register for WCAN

WCAN_PAGE
Page Register for WCAN **($0B8_H$)** **Reset Value: Table 779**
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
OP		**STNR**		**0**	**PAGE**		
w		w		r	rw		

Field	Bits	Type	Description
PAGE	2:0	rw	**Page Bits** When written, the value indicates the new page address. When read, the value indicates the currently active page = addr[y:x+1]. 000_B **PAGE0**, Selection is PAGE0 001_B **PAGE1**, Selection is PAGE1 010_B **PAGE2**, Selection is PAGE2 011_B **PAGE3**, Selection is PAGE3 100_B **PAGE4**, Selection is PAGE4 101_B **PAGE5**, Selection is PAGE5 110_B **PAGE6**, Selection is PAGE6 111_B **PAGE7**, Selection is PAGE7
STNR	5:4	w	**Storage Number** This number indicates which storage bitfield is the target of the operation defined by bit OP. If OP = 10_B, the contents of PAGE are saved in STx before being overwritten with the new value. If OP = 11_B, the contents of PAGE are overwritten by the contents of STx. The value written to bitfield PAGE is ignored. 00_B **MOD_ST0**, ST0 is selected 01_B **MOD_ST1**, ST1 is selected 10_B **MOD_ST2**, ST2 is selected 11_B **MOD_ST3**, ST3 is selected

Field	Bits	Type	Description
OP	7:6	w	**Operation** 00_B **PAGE_MANUAL0**, Manual page mode. The value of STNR is ignored and PAGE is directly written. 01_B **PAGE_MANUAL1**, Manual page mode. The value of STNR is ignored and PAGE is directly written 10_B **PAGE_SAVE**, New page programming with automatic page saving. The value written to the bit positions of PAGE is stored. In parallel, the former contents of PAGE are saved in the storage bit field STx indicated by STNR. 11_B **PAGE_RESTORE**, Automatic restore page action. The value written to the bit positions PAGE is ignored and instead, PAGE is overwritten by the contents of the storage bit field STx indicated by STNR.
0	3	r	**Reserved** Returns 0 when read; shall be written with 0.

Table 779 Reset Values of WCAN_PAGE

Reset Type	Reset Value	Note
LVD Reset	0000 $X000_B$	
Generated Reset	0000 $–000_B$	

49.17.5 WCAN Module Implementation

This section describes the WCAN module interfaces with the clock control, port connections, interrupt control, and address decoding.

49.17.5.1 Wake-Up CAN Module Registers

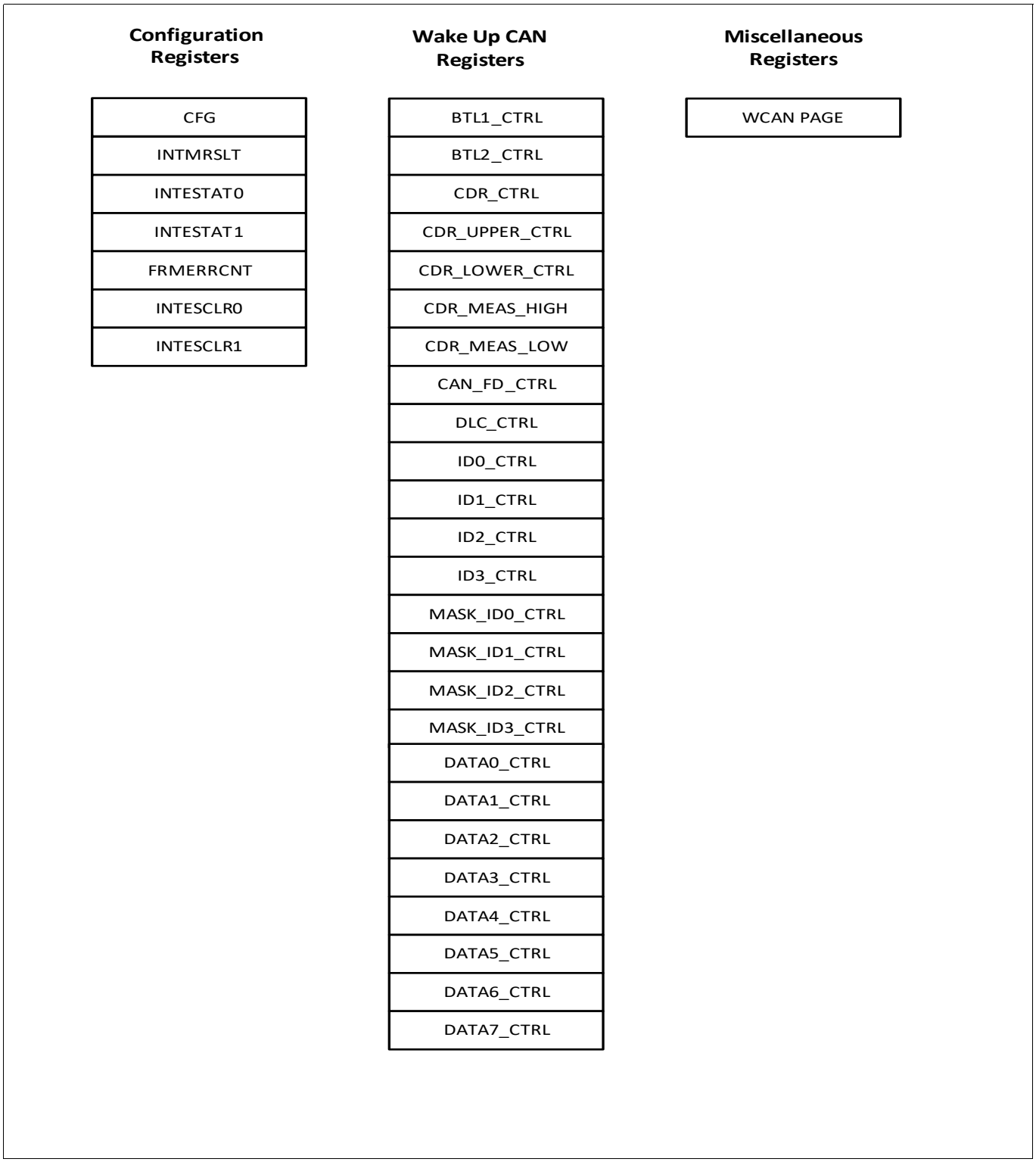


Figure 922 Wake-Up CAN Special Function Registers on Standby Controller (SCR)

The addresses of the kernel SFRs is listed in **Table 780**.

Table 780 Register Overview - WCAN (sorted by Name)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
WCAN_BTL1_CTRL	Bit Timing Logic 1 Control Register	0	WCAN_PAGE=1	$0B5_H$	342
WCAN_BTL2_CTRL	Bit Timing Logic 2 Control Register	0	WCAN_PAGE=1	$0B6_H$	343
WCAN_CDR_CTRL	CDR Control Register	0	WCAN_PAGE=0	$0B2_H$	343
WCAN_CDR_LOWER_CTRL	CDR Lower Limit Control Register	0	WCAN_PAGE=0	$0B4_H$	345
WCAN_CDR_MEAS_HIGH	CDR Measured High Register	0	WCAN_PAGE=0	$0B5_H$	346
WCAN_CDR_MEAS_LOW	CDR Measured Low Register	0	WCAN_PAGE=0	$0B6_H$	346
WCAN_CDR_UPPER_CTRL	CDR Upper Limit Control Register	0	WCAN_PAGE=0	$0B3_H$	344
WCAN_CFG	Wake-Up CAN Configuration Register	0	WCAN_PAGE=0	$0B0_H$	318
WCAN_DATAn_CTRL	DATA Control Register n	0	WCAN_PAGE=3	$0B0_H$+n	352
WCAN_DLC_CTRL	Message Data Length Code Control Register	0	WCAN_PAGE=1	$0B7_H$	348
WCAN_FD_CTRL	CAN FD Control Register	0	WCAN_PAGE=0	$0B7_H$	347
WCAN_FRMERRCNT	Wake-Up CAN Error Counter Register	0	WCAN_PAGE=1	$0B2_H$	322
WCAN_ID0_CTRL	Message Identifier Control Register 0	0	WCAN_PAGE=2	$0B0_H$	348
WCAN_ID1_CTRL	Message Identifier Control Register 1	0	WCAN_PAGE=2	$0B1_H$	349
WCAN_ID2_CTRL	Message Identifier Control Register 2	0	WCAN_PAGE=2	$0B2_H$	349
WCAN_ID3_CTRL	Message Identifier Control Register 3	0	WCAN_PAGE=2	$0B3_H$	350
WCAN_INTESCLR0	Wake-Up CAN Interrupt and Event Status Clear Register 0	0	WCAN_PAGE=1	$0B3_H$	323
WCAN_INTESCLR1	Wake-Up CAN Interrupt and Event Status Clear Register 1	0	WCAN_PAGE=1	$0B4_H$	324
WCAN_INTESTAT0	Wake-Up CAN Interrupt and Event Status Register 0	0	WCAN_PAGE=1	$0B0_H$	320
WCAN_INTESTAT1	Wake-Up CAN Interrupt and Event Status Register 1	0	WCAN_PAGE=1	$0B1_H$	321
WCAN_INTMRSLT	Wake-Up CAN Interrupt Mask Register	0	WCAN_PAGE=0	$0B1_H$	319
WCAN_MASK_ID0_CTRL	Message Identifier Mask Register 0	0	WCAN_PAGE=2	$0B4_H$	350

Table 780 Register Overview - WCAN (sorted by Name) (cont'd)

Short Name	Description	RMAP	PAGE	Offset Address	Page Number
WCAN_MASK_ID1_CTRL	Message Identifier Mask Register 1	0	WCAN_PAGE=2	$0B5_H$	351
WCAN_MASK_ID2_CTRL	Message Identifier Mask Register 2	0	WCAN_PAGE=2	$0B6_H$	351
WCAN_MASK_ID3_CTRL	Message Identifier Mask Register 3	0	WCAN_PAGE=2	$0B7_H$	352
WCAN_PAGE	Page Register for WCAN	0	X	$0B8_H$	354

49.17.5.2 Port and Interrupt Control

The interconnections between the Wake-Up CAN module and the port I/O lines are controlled in the port control logic for Port 13. Additionally to the port input selection, the following port control operations must be executed:

- Input/output function selection (SCR_IOCR registers (SCR module) and IOCR registers (GPIO module))

49.17.5.2.1 Input/Output Function Selection in Ports

The port input/output control registers contain the bit fields that selects the digital output and input driver characteristics such as pull-up/down devices, port direction (input/output), open-drain, and alternate output selections. The I/O lines for the WCAN module are controlled by the port input/output control registers shown below (see MODPISEL0 register in SCU chapter).

Table 781 shows how bits and bit fields must be programmed for the required I/O functionality of the CAN I/O lines.

Table 781 WCAN I/O Control Selection and Setup

Module	Port Lines	Input/Output Control Register Bits	I/O
CAN	CAN0		
	P33.10 / WCANRXDA	SCR_P01_IOCR2.PC2 = 0XXXXB	Input
	P33.7 / WCANRXDB	SCR_P00_IOCR7.PC7 = 0XXXXB	Input
	P33.13 / WCANRXDC	SCR_P01_IOCR5.PC5 = 0XXXXB	Input
	P33.5 / WCANRXDD	SCR_P00_IOCR5.PC5 = 0XXXXB	Input
	P33.2 / WCANRXDE	SCR_P00_IOCR2.PC2 = 0XXXXB	Input
	P14.1 / WCANRXDF [1)]	P14_IOCR1.PC1 = 0XXXXB	Input
	P33.12 / WCANRXDG	SCR_P01_IOCR4.PC4 = 0XXXXB	Input

Note: *[1)] The pad control of this pin is in the 32-bit GPIO module. This port can be used only when VDDEXT is supplied.*

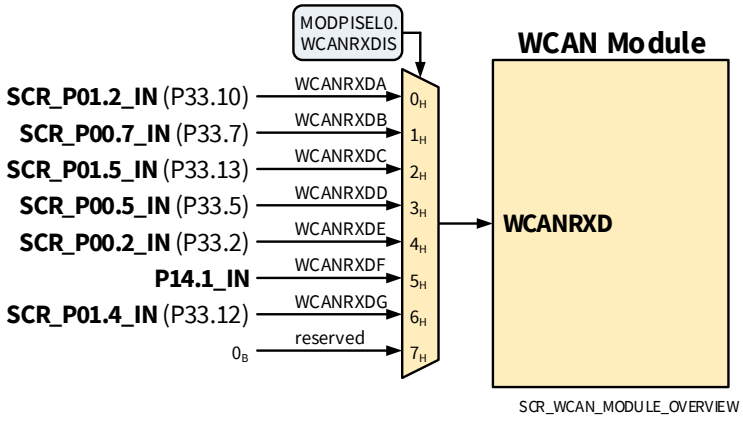


Figure 923 Wake-Up CAN Connections to GPIO Ports

49.17.5.2.2 Interrupt Control

The interrupt control logic in the Wake-Up CAN module consists of the following interrupts shown in **Table 782** and **Figure 924**.

Table 782 WCAN Interrupt Outputs

Interrupt inputs	Connected to WCAN Interrupt Output
SCR Interrupt Node 6	WUF_INT (Wake-Up Frame Interrupt)
	ERROR_INT (System Error Interrupt)
	CANTO_INT (CAN Timeout Interrupt)
	WUP_INT (WUP Detected Interrupt)

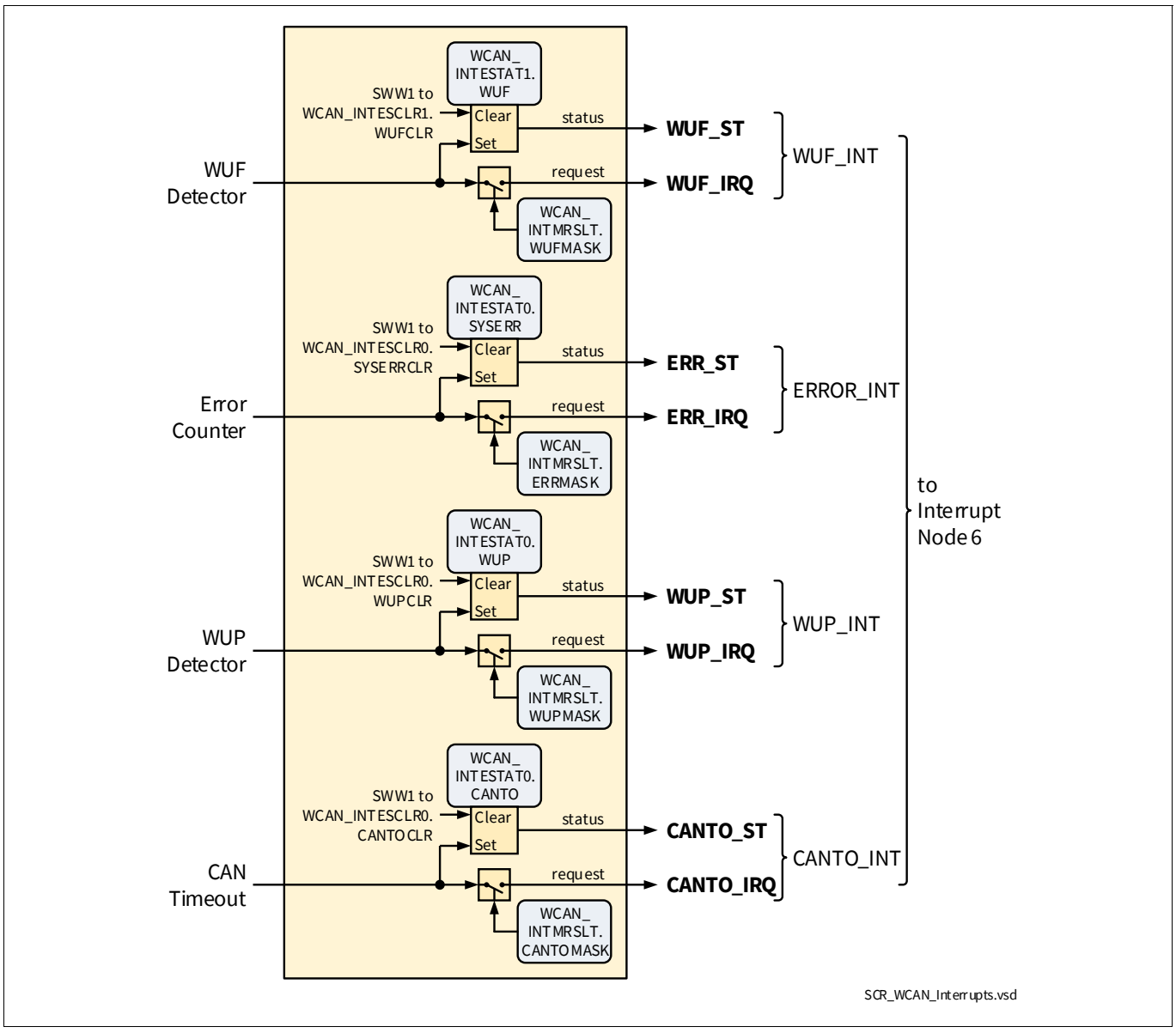


Figure 924 Wake-Up CAN Interrupt Outputs

49.17.6 Revision History

Table 783 Revision History

Reference	Change to Previous Version	
v3.2		
	First Official Release of completely reworked SCR chapter	
v3.3		
Page 326	Replaced f_{EVR_OSC} with f_{BACK}	

49.18 Debug System

This chapter describes the XC800/SCR debug system, in particular focus on the OCDS unit which provides the hardware to support debug functionality.

49.18.1 Overview

The debug system comprises of the On-Chip Debug Support (OCDS) unit and Debug-Monitor in ROM which provides basic debug functionality for XC800/SCR-based systems, controlled directly by an external tool via debug interface pin(s).

49.18.1.1 Features

The main debug functionality supported are:

- Set breakpoints on instruction address and on address range within the Program Memory
- Set breakpoints on Internal RAM address range
- Support unlimited software breakpoints in Flash/RAM code region
- Process external breaks via debug interface
- Step through the program code
- User handling of OCDS NMI in case of IRAM read/write breakpoint

49.18.1.2 Components of the Debug System

The components of the debug system are briefly described here.

49.18.1.2.1 Debug Interface

The debug interface allows to access the device, such as for data read or write. The debug interface supports:

- 2-pin Device Access Port (DAP)
- Single-pin DAP (SPD)

The DAP interface refers to the dual-pin Device Access Port standardized for the Infineon microcontrollers. It offers high noise immunity and robustness, and utilizes only two pins DAP0 (clock), DAP1 (serial data in/out).

The SPD interface refers to the single-pin Device Access Port standardized for the Infineon microcontrollers. It utilizes only one pin DAP1 for serial data in/out.

49.18.1.2.2 Monitor Program

Part of the Boot ROM is occupied by the OCDS Monitor program (or OCDS firmware). On the command from the debugger, the Monitor program executes the actual instructions for accessing the addressable locations such as memories, sfrs of the device.

49.18.1.2.3 On-Chip Debug System Unit (OCDS)

The OCDS hardware is the core of the debug system, controlling the debugging with interfaces to the XC800 CPU.

49.18.2 Product Specific Information

This section provides product specific information relevant to the OCDS.

49.18.2.1 Pinning

Table 784 describes the OCDS pin function.

Table 784 OCDS Pin Functions

Function	Description
DAP0_1	Dual-pin debug access port: clock
DAP0_0	
DAP1/SPD_1	Single-pin debug access port: data
DAP1/SPD_0	

The pin will be set up via hardware once the OCDS mode is selected. Detailed pinout is given in the GPIO chapter.

49.18.2.2 Clocking Configuration

The OCDS runs at the peripheral frequency, PCLK.

49.18.2.3 JTAG ID

The following assignment of JTAG ID are used for SCR:

VERSION = 1_H, PART_NUMBER = $01EC_H$, i.e. JTAG ID Register = $101EC083_H$

49.18.3 Revision History

Table 785 Revision History

Reference	Change to Previous Version	
v3.5		
	First Official Release of completely reworked SCR chapter	
	No change	

49.19 Device Access Port (DAP2)

In addition to the pins functionally used by the application some pins are needed to communicate with tools during development. Tools commonly used are board tester, debugger, emulator, calibration tools and performance analyzers. On the other hand the cost inferred by each non-functional pin is a strong argument to reduce the tool access port to as few pins as possible.

The standardized device access port (DAP2) of all current micro controllers offers a convenient method to get the required functionality at the least possible cost: Depending on configuration, one, two of the device's pins are reserved for tooling until the next reset. The different modes are detailed in this chapter.

DAP2 is used for debugging purposes together with the OCDS module, supported by the startup firmware for halt-after-reset on user code start address or by appropriate DAP telegram for "hot-attach" after reset (HMONITOR command to the SCR).

For SCR, the DAP kernel interfaces with the SPD unit with in-direct control of DAP port via the SPD. The addition of the SPD unit supports single-pin-DAP in addition to the standard dual-pin DAP. In this chapter, 'DAP[x]' generally refers to the 'SD_DAP[x]' which is the signal interfacing between DAP and SPD units.

This chapter also refers to the POSD reset event. The debug system of SCR is reset in case of an LVD reset. Therefore POSD in this chapter complies to LVD Reset.

TRST is inside the SCR attached to the PORST pin. The device is capable of pure JTAG only. No IO_Client is implemented.

DAP Features

Please note that the following list party represents the combination of DAP and IOClient functionality.

- Generic protocol with minimum overhead
- High-speed due to synchronous clocking
- Low pin-count
  - DAP0 clock pin
  - DAP1 data pin
- Robust interface
  - Telegrams protected by 6-bit CRC (CRC6). Optional 32-bit CRC (CRC32) checks for critical communication sequences
  - Interface enabling pattern with 50 relevant bits
  - Tool can automatically detect and recover from any kind of reset
  - Electrically robust by using standard pads with full ESD protection
- Optimized random memory accesses
  - Combined read data and set next address telegram hides internal access latency
  - Optional usage of short 16-bit offset address instead of 32 bit
  - 8/16 bit data is transferred with 8/16 bit width instead of 32 bit
  - Results in ca. 0.5 us for a random read of arbitrary data size at 160 MHz
- Effective DAP block access bandwidth for 160 MHz clocking:
  - 15 MByte/s for block read or write
- Device to tool triggering mode with minimum overhead
- On-chip JTAG infrastructure can be controlled
- Very cost effective tool hardware possible

Please note that the block access bandwidth scales linear with the DAP clock frequency. The maximum DAP frequency is product specific and can be lower than 160 MHz. It is assumed that the internal bus clock frequency is high enough.

49.19.1 Revision History

Table 786 Revision History

Reference	Change to Previous Version	Change Request
v3.3		
	First Official Release of completely reworked SCR chapter	
	No change	

49.20 Single Pin DAP (SPD)

The SPD module converts between the single pin and the two pin DAP protocol (**Figure 925**).

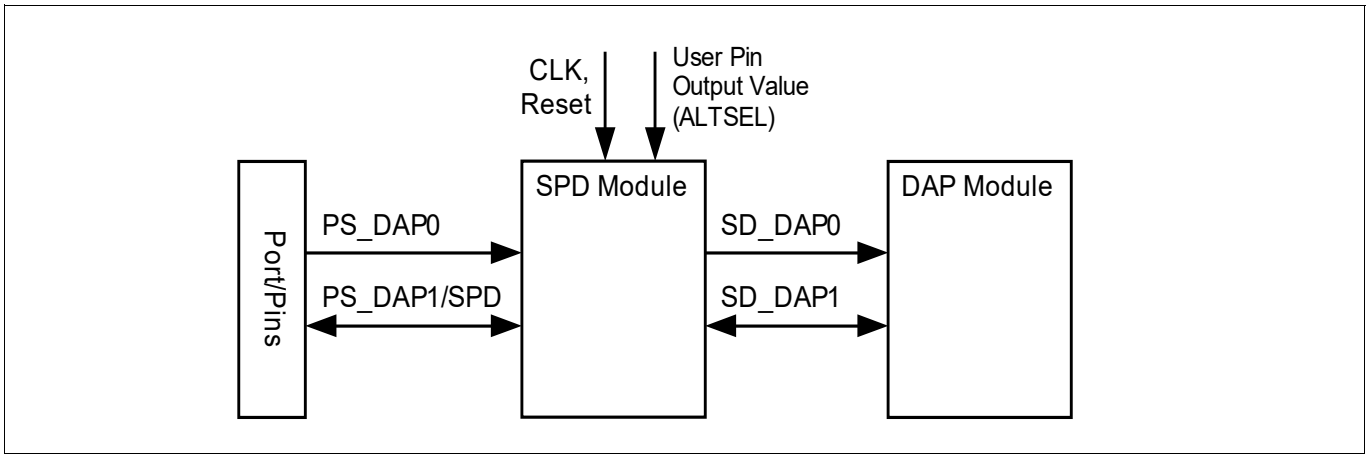


Figure 925 SPD System Integration

Features

- 2 MHz bit frequency (effective 1.3 Mbits/s for DAP telegrams)
- Digital and small implementation
- Very robust against clock deviations between tool and device

49.20.1 Revision History

Table 787 Revision History

Reference	Change to Previous Version	Change Request
v2.1		
	First Official Release of completely reworked SCR chapter	
	No change	

Revision history

Document version	Date of release	Description of changes
V2.0.0	2021-02	• Version comparison table updated. • For further changes see respective revision history of each chapter. The version comparison table below gives an overview.
V1.6.0	2020-08	• Version comparison table updated. • For further changes see respective revision history of each chapter. The version comparison table below gives an overview. • Removed device TC3Ax from set of documentation.
V1.5.0	2020-04	• Version comparison table updated. • For further changes see respective revision history of each chapter. The version comparison table below gives an overview.
V1.4.0	2019-12	• Added TC3Ax appendix as target specification. • Version comparison table updated. • For further changes see respective revision history of each chapter. The version comparison table below gives an overview.
V1.3.0	2019-09	• Added additional device TC3Ax to AURIX™ TC3xx set of documentation. • Version comparison table updated. • For further changes see respective revision history of each chapter. The version comparison table below gives an overview.
V1.2.0	2019-04	• Added additional device TC3Ex to AURIX™ TC3xx set of documentation. • Version comparison table updated. • For further changes see respective revision history of each chapter. The version comparison table below gives an overview.
V1.1.0	2019-01	• Power Management System for Low-End (PMSLE) added. • TC33x and TC33xEXT added. • Changes in connectivity tables. • Detailed Revision History contained in each chapter.
V1.0.0	2018-08	• First revision of the User's Manual. • Detailed OCDS and MiniMCDS information is not contained. Available under NDA. Overview available in Introduction chapter. • Detailed Revision History contained in each chapter.

Version comparison table for AURIX™ TC3xx Family Specification

Chapter name	UM V1.6.0 chapter version	UM V2.0.0 chapter version	Content changes
Introduction	V1.6.3	V1.6.4	**Yes, see chapter revision history**
• OCDS	V3.1.15	V3.1.15	No
• ED	V1.0.6	V1.0.6	No
• SOTA	V1.0.5	V1.0.5	No

Chapter name	UM V1.6.0 chapter version	UM V2.0.0 chapter version	Content changes
MEMMAP	V0.1.20	V0.1.21	**Yes, see chapter revision history**
FW	V1.1.0.1.17	V1.1.0.1.18	**Yes, see chapter revision history**
SRI Fabric	V1.1.16	V1.1.17	**Yes, see chapter revision history**
• FPI, BCU	V1.2.8	V1.2.9	No functional changes
CPU	V1.1.20	V1.1.21	No
NVM Subsystem	V2.0.7	V2.0.7	No
• DMU	V2.0.11	V2.0.12	**Yes, see chapter revision history**
• PFI	V2.0.1	V2.0.1	No
• NVM	V2.0.6	V2.0.6	No
• UCB	V2.0.22	V2.0.22	No
LMU	V3.1.16	V3.1.16	No
DAM	V1.3.11	V1.3.12	No
SCU	V2.1.26	V2.1.27	**Yes, see chapter revision history**
CCU	V2.0.31	V2.0.32	**Yes, see chapter revision history**
PMS	V2.2.33	V2.2.34	**Yes, see chapter revision history**
PMSLE	V1.0.6	V1.0.7	**Yes, see chapter revision history**
MTU	V7.4.12	V7.4.13	**Yes, see chapter revision history**
PORTS	V1.8.21	V1.8.21	No
SMU	V4.0.22	V4.0.23	**Yes, see chapter revision history**
INT	V1.2.11	V1.2.11	No
FCE	V4.2.9	V4.2.9	No
DMA	V0.1.18	V0.1.18	No
SPU	V1.1.24	V1.1.25	**Yes, see chapter revision history**
SPU2	n/a	n/a	**Chapter removed due to TC3Ax discontinuation**
BITMGR	n/a	n/a	**Chapter removed due to TC3Ax discontinuation**
SPULCKSTP	V1.2.5	V1.2.5	No
EMEM	V1.4.4	V1.4.4	No
RIF	V1.0.40	V1.0.43	**Yes, see chapter revision history**
HSPDM	V0.7.9	V0.7.9	No
CIF	V1.4.12	V1.4.12	No
STM	V9.2.4	V9.2.4	No
GTM	V2.2.23	V2.2.24	**Yes, see chapter revision history**
CCU6	V3.0.0	V3.0.0	No
GPT12	V3.0.2	V3.0.2	No
CONVCTRL	V3.0.1	V3.0.1	No
EVADC	V3.0.4	V3.0.5	**Yes, see chapter revision history**

Chapter name	UM V1.6.0 chapter version	UM V2.0.0 chapter version	Content changes
EDSADC	V3.0.5	V3.0.6	**Yes, see chapter revision history**
I2C	V2.3.6	V2.3.6	No
HSSL	V3.0.18	V3.0.19	**Yes, see chapter revision history**
• HSCT	V2.3.15	V2.3.15	No
ASCLIN	V3.2.8	V3.2.8	No
QSPI	V3.0.20	V3.0.20	No
MSC	V5.0.11	V5.0.11	No
SENT	V2.1.10	V2.1.10	No
MCMCAN	V1.19.13	V1.19.13	No
E-Ray	V3.2.10	V3.2.11	**Yes, see chapter revision history**
PSI5	V1.17.12	V1.17.12	No
PSI5-S	V1.12.10	V1.12.10	No
GETH	V1.3.14	V1.3.15	**Yes, see chapter revision history**
EBU	V4.0.12	V4.0.12	No
SDMMC	V1.0.18	V1.0.18	No
HSM	V2.3.9	V2.3.9	No
IOM	V2.1.15	V2.1.15	No
SCR and all subchapters	V4.0.4	V4.0.5	**Yes, see chapter revision history**

Known Issues, not considered in Chapter Revision Histories

Note: Common entries in chapter revision histories like “Connection Tables changed” or “Register and Connectivity Tables updated” mean formal changes only. No functional changes.

Chapter name	Chapter version	Family Specification content changes

Trademarks of Infineon Technologies AG

µHVIC™, µIPM™, µPFC™, AU-ConvertIR™, AURIX™, C166™, CanPAK™, CIPOS™, CIPURSE™, CoolDP™, CoolGaN™, COOLiR™, CoolMOS™, CoolSET™, CoolSiC™, DAVE™, DI-POL™, DirectFET™, DrBlade™, EasyPIM™, EconoBRIDGE™, EconoDUAL™, EconoPACK™, EconoPIM™, EiceDRIVER™, eupec™, FCOS™, GaNpowIR™, HEXFET™, HITFET™, HybridPACK™, iMOTION™, IRAM™, ISOFACE™, IsoPACK™, LEDrivIR™, LITIX™, MIPAQ™, ModSTACK™, my-d™, NovalithIC™, OPTIGA™, OptiMOS™, ORIGA™, PowIRaudio™, PowIRStage™, PrimePACK™, PrimeSTACK™, PROFET™, PRO-SIL™, RASIC™, REAL3™, SmartLEWIS™, SOLID FLASH™, SPOC™, StrongIRFET™, SupIRBuck™, TEMPFET™, TRENCHSTOP™, TriCore™, UHVIC™, XHP™, XMC™.

Trademarks updated November 2015

Other Trademarks

All referenced product or service names and trademarks are the property of their respective owners.

Edition 2021-02

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference

IMPORTANT NOTICE

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.